AD-A239 997



(2)

NAVAL POSTGRADUATE SCHOOL

Monterey, California

DTIC
ELECTE
SEP 0 4 1991
S D
D



THESIS

PART I

A DESIGN OF
COMPUTER AIDED INSTRUCTIONS (CAI)
FOR UNDIRECTED GRAPHS IN
THE DISCRETE MATH TUTORIAL (DMT)

by

Atilla Bakan & Yavuz Bas

June 1990

Thesis Advisors: Hefner & Shing

Approved for public release; distribution is unlimited.

91-09454



91

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
A DESIGN OF COMPUTER AIDED INSTRUCTIONS (CAI) FOR UNDIRECTED GRAPHS IN THE DISCRETE MATH TUTORIAL (DMT)

12. PERSONAL AUTHOR(S)
Bakan, Atilla and Bas, Yavuz

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM _____ TO _____	June 1990	1594

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Computer Aided Instructions, Tutorial, Undirected Graphs, Discrete Math Tutorial (DMT).

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The objective of this thesis research is to create a tutorial for teaching aspects of "undirected graphs" in discrete math. It is one of the submodules of the Discrete Math Tutorial (DMT), which is a Computer Aided Instructional (CAI) tool for teaching discrete math to the Naval Academy and the Westpoint Military Academy students. In order to accomplish the objective, an exploration of various conventional CAI techniques is necessary to determine which methods are readily adaptable for use with the PC s. Many of the design issues normally associated with the development of CAI packages are exasperated by the current physical limitations of the PC-based systems. With proper design and appropriate trade-offs, however, effective CAI

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: Prof. Mantak Shing and Prof. Kim Hefner	22b. TELEPHONE (Include Area Code): (408) 646-2634 / 646-2198; 22c. OFFICE SYMBOL: CS / MA

effective CAI packages for the PC s are possible. The software designed for this thesis is only an example of the possibilities made available by the PC s.

Approved for public release; distribution is unlimited.

A DESIGN OF COMPUTER AIDED INSTRUCTIONS (CAI) FOR UNDIRECTED GRAPHS IN THE DISCRETE MATH TUTORIAL (DMT)

by

Atilla Bakan
Ltjg., Turkish Navy
B.S., Turkish Naval Academy, 1984

and

Yavuz Bas
Ltjg., Turkish Navy
B.S., Turkish Naval Academy, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1990

Authors: _____
Atilla Bakan

_____
Yavuz Bas

Approved By: _____
Kim Hefner, Co-Thesis Advisor

_____
Mantak Shing, Co-Thesis Advisor

_____
Robert B. McGhee, Chairman,
Department of Computer Science

DTIC COPY INSPECTED

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

ABSTRACT

The objective of this thesis research is to create a tutorial for teaching aspects of "undirected graphs" in discrete math. It is one of the submodules of the Discrete Math Tutorial (DMT), which is a Computer Aided Instructional (CAI) tool for teaching discrete math to the Naval Academy and the Westpoint Military Academy students. In order to accomplish the objective, an exploration of various conventional CAI techniques is necessary to determine which methods are readily adaptable for use with the PCs. Many of the design issues normally associated with the development of CAI packages are exasperated by the current physical limitations of the PC-based systems. With proper design and appropriate trade-offs, however, effective CAI packages for the PCs are possible. The software designed for this thesis is only an example of the possibilities made available by the PCs.

TABLE OF CONTENTS

I. INTRODUCTION 1

II. BACKGROUND 3

A. WHAT IS A COMPUTER AIDED INSTRUCTION (CAI)? 3

B. FIVE BASIC CATEGORIES OF CAI 3

C. DMT 4

III. UNDIRECTED GRAPHS IN DMT 6

A. DESIGN STRATEGY 6

1. Front-end Analysis 6

2. Lesson Design 6

a. Graphs and Their Representations 8

b. Paths and Circuits 8

c. Coloring a Graph 8

d. Properties of Trees 9

e. Rooted Trees 9

f. Spanning Trees 9

g. Depth First Search 10

h. Minimal Spanning Trees 10

i. Binary Trees 10

j. Sorting and Searching 10

k. Language Syntax 10

IV. PRE-DEVELOPMENT CHOICES AND PROBLEMS 11

A. CHOICE OF HARDWARE AND LANGUAGE 11

B. PROBLEMS ENCOUNTERED DURING DEVELOPMENT 11

V. HOW TO USE THE PROGRAM 13

VI. CONCLUSIONS AND RECOMMENDATIONS 15

APPENDIX- LISTING OF THE SOURCE CODE 16

LIST OF REFERENCES 1582

BIBLIOGRAPHY 1583

INITIAL DISTRIBUTION LIST 1584

LIST OF FIGURES

1. An Example of Text Windows.....14

2. An Example of Graphics Windows.....14

I. INTRODUCTION

It was only two decades ago that parents were sitting at the kitchen table trying to make sense of the "new math" their children studying in school. The parents were puzzled and dismayed at their own inability to understand their ten-year-old's homework. But that proved to be only the beginning. Today we find that "yesterday's children" are facing a similar problem. Yesterday's child, the one who helped his/her parents understand "new math", is seeing his/her children using computers in their classrooms, a "magical" tool to a lot of parents.

The rate at which computers are being used in schools is rapidly increasing. Today it is not unusual to have a student going to a programming class and being taught how to use a computer by another computer, or to study their science, history, math or other subjects via the computer. Another change from "yesterday" is that this learning via the computer does not stop once the child leaves the classroom. Chances are they can take their assignments home and finish them at home.

In 1982, over two million home computers were sold and by 1986, this figure had increased to 10 million. Children can go to computer camps to either learn how to use one or to teach others. Computers are a big "hit" on the college and university campuses today. For instance, each of our target users, the cadets in the Westpoint Military Academy and the midshipmen in the Naval Academy, has his/her own computer in their dormitories. Some colleges and universities are insisting that all incoming students have computers.

With appropriate software, computers can become a valuable learning tool. The key to successful implementation of these devices in education is the development of software which has been appropriately designed to the unique characteristics of the course and the microcomputer in question, and the abilities of the learner population [Ref. 1]. The field of statistics has traditionally been on the leading edge of applying

computer technology to assist in the large amount of tedious calculations involved in statistical analysis. As such, a wide variety of statistical educational software is available for use on mainframes and minicomputers. In addition, programmable calculators have frequently been used to assist both students and statistician in completing basic mathematical calculations. However the development of similar packages for learning statistics or discrete mathematics using the PC has been lagging far behind. Current limitations of the PC, particularly in the area of memory capacity, have discouraged the implementation of full scale PC-based CAI packages. With proper design and appropriate trade-offs, however. effective CAI packages for the PC are possible. The PC is too valuable as an educational tool to be cast aside for creative software designs. The possibilities encompassed by individual use of a PC far exceed those of the personal calculators which have already revolutionized many aspects of basic statistical education. The software developed in this thesis is only an example of the possibilities made available by the PC. It is hoped that this treatment will serve as the impetus for others to develop and implement much more extensive PC-based software.

II. BACKGROUND

A. WHAT IS A COMPUTER AIDED INSTRUCTION (CAI) ?

Like many things in this rapidly changing world we live in, CAI has been defined in numerous ways, some of which include the following :

Computer Aided Instruction refers to the use of computers in an interactive manner where the computer both presents material to and receives analysis. It acts upon responses from each student on an individual basis. [Ref. 2]

or,

Computer Aided Instruction is a supplementary classroom tool that helps teachers teach more effectively. It is a system that enables many students and a single teacher to engage in a one-to-one dialogue, using a high speed digital computer as the communication medium. Because of this, instruction becomes a two-way communication system that enables the teacher to monitor the progress and tailor instruction to fit a student's needs. It accomplishes this mighty task by enabling the instructor to deliver an appropriate instructional move at the precise moment when a particular instruction action is needed. [Ref. 3]

or, very simply,

Instruction that is assisted or aided through use of the computer. [Ref. 4]

B. FIVE BASIC CATEGORIES OF CAI

CAI can be subdivided into five "teaching strategies" :

- Drill and Practice,
- Tutorial,
- Problem Solving,
- Simulation,
- Instructional Games. [Ref. 5]

The goal of the *drill and practice* strategy is to cement the retention of the material by repeated practice and exercise. The *tutorial* strategy encompasses a great deal more than the relatively simple drill and practice strategy. Where as drill and practice programs serve as an adjunct to the instructor, tutorial programs strive to "replace" the instructor as much as possible. *Problem solving* encompasses programs which are written to solve certain select problems in a specific area. *Simulation* is generally used for the generation and manipulation of data or the repetitive cycling of a model when environmental factors preclude normal methods. The distinctive feature of *instructional games* is the attempt to use a student's competitive nature to achieve certain learning goals or skills.

As for our thesis, we take the tutorial approach and develop a part of the Discrete Mathematics Tutorial (DMT) that aims at teaching discrete math to the academy students. In the following section we will introduce DMT briefly.

C. DMT

CAI systems may be generally viewed in two ways : machine-directed and student-directed. Discrete Math Tutorial (DMT) lies in between the two approaches. DMT teaches a particular lesson as it is programmed by its designers. On the other hand the student has always the capability and the opportunity of controlling the flow of the lessons via the functions provided by DMT, such as PgUp/PgDn.

The DMT provides a simple mixed initiative environment that contains two parts. The first part presents a standard CAI lesson to the student. The DMT presents each lesson as pages (or windows) on the screen. The student is allowed to move from topic to topic in any particular lesson. The second part provides the mixed initiative environment. Within the lesson, the student may access the DMT user interface. The DMT user interface contains the following functionality :

- Provides the user with the ability to print (or save to disk) any definition, algorithm or example in the DMT.

- Provides the user with the ability to run any algorithm in the DMT using his own data.
- Provides the user with a simple calculator.

These three functions allow the user to stop the lesson he/she is currently working on at any time and pursue topics of interest. For example, a user may find a term he/she does not remember during a lesson. All he/she has to do is access the DMT interface and look up the unknown definition and then return to the lesson if he/she so desires. Finally the user can access a simple calculator to help with difficult computations.

These three functions provide a noncomplex, but effective, mixed initiative environment for the user to learn discrete math.

For this thesis, we have designed the **undirected graphs** lesson, which is one of the modules of the DMT.

III. UNDIRECTED GRAPHS IN DMT

Even though graphs have been studied for a long time, the increased use of computer technology has generated a new interest in them. Not only have applications of graphs been found in computer science but in many other areas such as business and science. As a consequence, the study of graphs has become very important.

In this thesis, we give the user basic concepts of graphs, and by choosing suitable examples, we try to give them the flavor of different applications.

A. DESIGN STRATEGY

1. Front-end analysis

The target population is the U.S. Naval Academy students and the Westpoint Military Academy students. The system on which the tutorial is to be implemented is an IBM PC or compatible computer with VGA graphics cards and monitors. None of the topics takes more than one hour to complete.

2. Lesson Design

The objective of this design is to teach the following topics :

- Graphs and their representations,
- Paths and circuits,
- Coloring a graph,
- Properties of trees,
- Rooted trees,
- Spanning trees and Breadth First Search
- Depth First Search
- Minimal spanning trees,
- Binary trees,
- Sorting and searching,
- Language syntax.

Each topic starts with an introduction which is followed by definitions, theorems, examples and finally exercises.

We have used definitions to explain keywords and new concepts in each topic. We usually illustrate these definitions with suitable pictures. We believe that the best way to amplify a section of a lesson is to provide examples.

During a lesson, whenever it is appropriate, we introduce theorems to show the principles related to the topics being taught. For each theorem, a proof is provided to prevent dogmatism.

At the end of each section we provide many exercises to cover each concept in that section. We think that **practice makes perfect.** The exercises used are multiple choice, True/False, and test-yourself types of questions. When the user inputs the wrong answer in the first two types of questions, he/she receives the correct answer together with the reasoning for the answer. In the test-yourself type of questions, we let the user solve the problems by himself/herself, and when he/she is ready, he/she can see if the answer is correct. This type of question is used for algorithm applications. In addition to seeing the correct answer, if he/she likes, the user can step through the solutions and also trace through the algorithm's steps to refresh his/her memory.

The reason why we have chosen the step-by-step approach in the solutions of some of the exercises and examples is because we believe that in preparing a course for individual study, it is necessary to analyze carefully the steps that comprise the procedures to be learned and the understanding required to perform them. To ensure that nothing crucial is misunderstood, every task that is identified in the job or area breakdown is itself divided into the steps that comprise that task.

In the following paragraphs we present the contents of each topic more specifically.

a. Graphs and Their Representations

It is quite common to represent situations involving objects and their relationships by drawing a diagram of points, with segments connecting those points that are related. The general idea in this section is to represent, by a picture, a set of objects and their relations. In this section we talk about the ways to represent graphs (namely matrix representations and adjacency lists), labeled graphs, and some definitions such as vertices, edges, connectivity, adjacency, etc. At the end of the section, we touch a little bit upon matrix operations.

b. Paths and Circuits

In the previous section we have seen that graphs can be used to describe a variety of situations. In many cases we want to know whether it is possible to go from one vertex to another following a sequence of edges. In other cases it may be necessary to perform a test that involves finding a route through all of the vertices or over all of the edges. While many situations can be described by graphs as we have defined them, there are others where it may be necessary to allow an edge from a vertex to itself or to allow more than one edge between vertices. In this section we define multigraphs, simple paths, cycles, Euler circuits and paths, and Hamiltonian cycles. We also look at ways to find a shortest path between vertices in a graph. At the end of this section we introduce "Isomorphism".

c. Coloring a Graph

Suppose that a chemical manufacturer needs to ship a variety of chemical products from a refinery to a processing plant. Shipping will be by rail. But according to regulations, not all of these chemical products can be shipped together in one railroad car because of the possibility that they mix together and create a violent reaction should an accident occur. How can these products be shipped? In order to minimize the expenses the manufacturer wants to use the smallest possible number of railroad cars. What is this number ?

In this example, we see that there are objects (chemical products) and relationships (cannot travel in the same railroad car) existing among them. Since these are the basic components of a graph, it seems natural to model it using a graph where the vertices are chemical products and an edge is drawn between two vertices whenever the two chemical products cannot be in the same railroad car. For the solution, we must assign labels to the vertices of the graph so that adjacent vertices have different labels. Each label represents a different railroad car. This idea occurs frequently in graph theory. For historical reasons the labels are called colors, and the problem is known as a graph coloring problem.

In this section we examine the chromatic numbers of graphs and the Welsh and Powell algorithm for coloring graphs. In the examples and exercises we have used color graphics to represent the colors of vertices.

d. Properties of Trees

Trees are a commonly occurring type of graph in models and computations of all kinds. Computer scientists are particularly familiar with these structures through the considerations of parse trees, tree searches, game trees and so on. In this section we define trees and provide further characterizations of trees through some theorems and examples.

e. Rooted Trees

People have always been interested in learning about the descendants of historically important individuals. To assist these investigations a genealogical chart is often drawn. Here comes the idea of rooted trees. In this section we introduce the parent child relationship in addition to the ancestor, descendant and terminal vertex concepts.

f. Spanning Trees

In this section we show how spanning trees play an important role in connection with the circuit space and with the separability of a graph. We also show

how the Breadth First Search algorithm is implemented using the step-by-step instruction method.

g. Depth First Search

Another algorithm for finding a spanning tree is the Depth First Search algorithm. In this section we show how this algorithm is implemented, again using a step-by-step approach.

h. Minimal Spanning Trees

In some cases we want to find a spanning tree in which the sum of the costs of all the edges is as small as possible. Here we introduce two algorithms which find minimal spanning trees, namely Prim's Minimal Spanning Tree Algorithm and Kruskal's Minimal Spanning Tree Algorithm.

i. Binary Trees

Binary trees are used extensively in computer science to represent ways to organize data and describe algorithms. In this section, we introduce the definition of a binary tree and explain related concepts. We also introduce the Polish Notation, the reverse Polish Notation, inorder, preorder, and postorder traversals. Each concept is illustrated with appropriate examples.

j. Sorting and Searching

Since our concern is trees, in this section we cover only the method of obtaining a sorted list from a binary search tree. To do this, we show how a binary search tree is constructed from a given list, then illustrate how the sort operation is performed. We also include how to search for a particular object in a binary search tree.

k. Language Syntax

In this section parse trees are introduced as an application of trees. We talk about what a grammar is and give example grammars. Additionally, we touch a little bit on the Backus-Naur form.

IV. PRE-DEVELOPMENT CHOICES AND PROBLEMS

A. CHOICE OF HARDWARE AND LANGUAGE

While hardware selection and programming language selection can be discussed as separate issues, the interrelationships between the two (particularly in microcomputer systems) require that this selection process be a combined procedure. Since our target users are the academy students who have their own PCs we have chosen the PCs as the platform for our software.

As for the language, we need a language that is powerful enough to allow us to make windows, menus for the DMT interface and create graphics screens. We think that the C programming language with its powerful graphics capabilities is the best possible one among the existing languages that meets our needs. We have written and compiled our programs using the Turbo C Ver. 2.0*.

For the DMT User Interface, a windowing package, CXL**, has been used. This package allows us to make different types of menus and do extensive operations with windows. It has over 260 functions related to windowing, menuing, data entry etc. With its available functions, we think that this software package will be very useful for the other stages of the DMT project.

B. PROBLEMS ENCOUNTERED DURING DEVELOPMENT

During the development phase, our program reached a point where it was no longer possible for us to use the existing small memory model. We had to consider two possible solutions :

---

*Turbo C Version 2.0 Copyright (c) 1987, 1988 Borland International

**CXL is Copyright (c) 1987 - 1988 Mike Smedley

- Switch to the larger memory models of Turbo C and use the corresponding CXI libraries, or
- Decompose our program into separately compilable modules.

We preferred the second alternative, because it would be easier to maintain the program this way. Moreover, it would let us add new lessons to the DMT by simply modifying the main module. The only disadvantage is that we had to link each separately compiled module with the necessary libraries.

V. HOW TO USE THE PROGRAM

Since our program is a part of the DMT, the only way of accessing the program is by way of the DMT User Interface. When the user chooses the graphs lesson from the main menu of the DMT interface, the sections of our lesson will come up as a sub-menu. Choosing any one of the sections will start that particular lesson. Here the user has the following alternatives :

- Pressing any key to continue sequentially,
- Using PgUp/PgDn keys to move from topic to topic anytime,
- Pressing the ESC key to return to the previous menu.

The user will see the "Press a key" and "PgUp/PgDn" directives at the bottom of the active window. An example of a text screen is shown in Figure 1. A typical graphics screen is reproduced in Figure 2.

Sometimes, there may be multiple windows in the screen, and some of them may be hidden by others. At this time, if the user misses the information in one of the hidden windows, he/she can bring this window to the front by pressing Ctrl PgUp key as many times as needed. Similarly, by using Ctrl PgDn key he/she can return to the latter windows.

During the exercises, depending on the type of questions, the user is expected to enter a certain type of input, such as "a", "b", "c", "d", "y" or "n". If the user enters any other character, he/she is prompted to enter an acceptable one.

Normally, each session terminates when the user completes the last exercise of that section, and he/she returns to the previous menu where he/she can choose another section. However, he/she can terminate that section by pressing the ESC key at any time he/she chooses. If the ESC is pressed, the program will ask the user to confirm the termination.

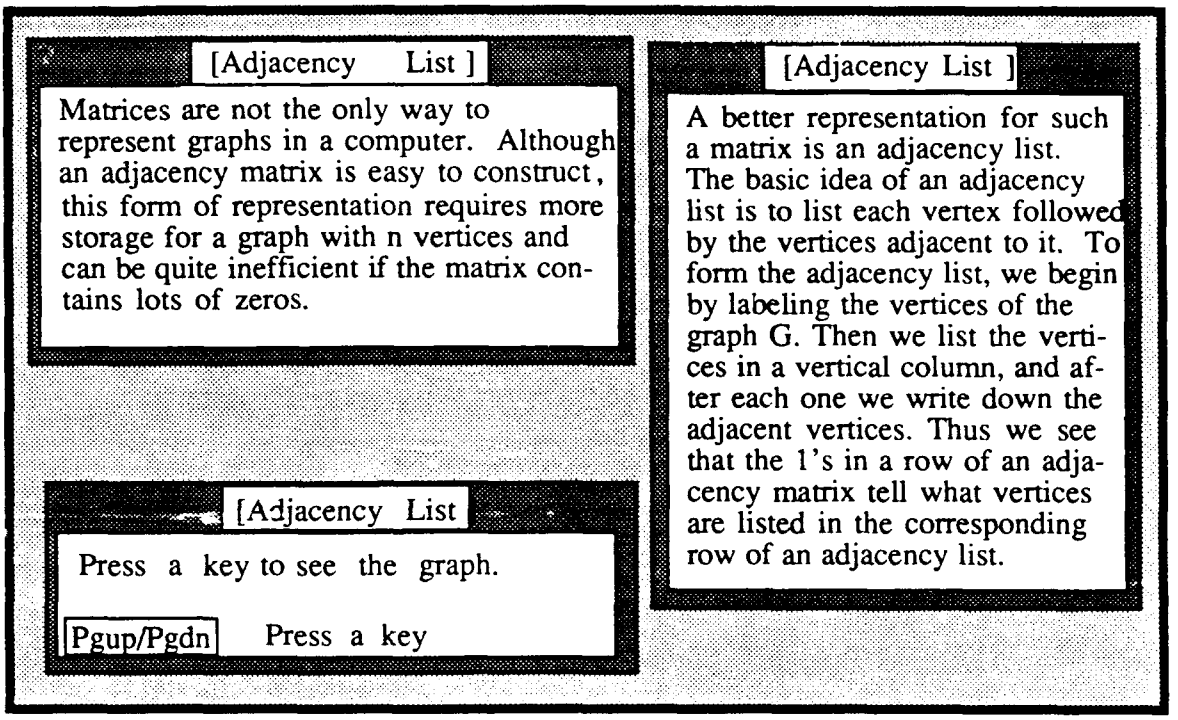


Figure 1. An Example of Text Windows.

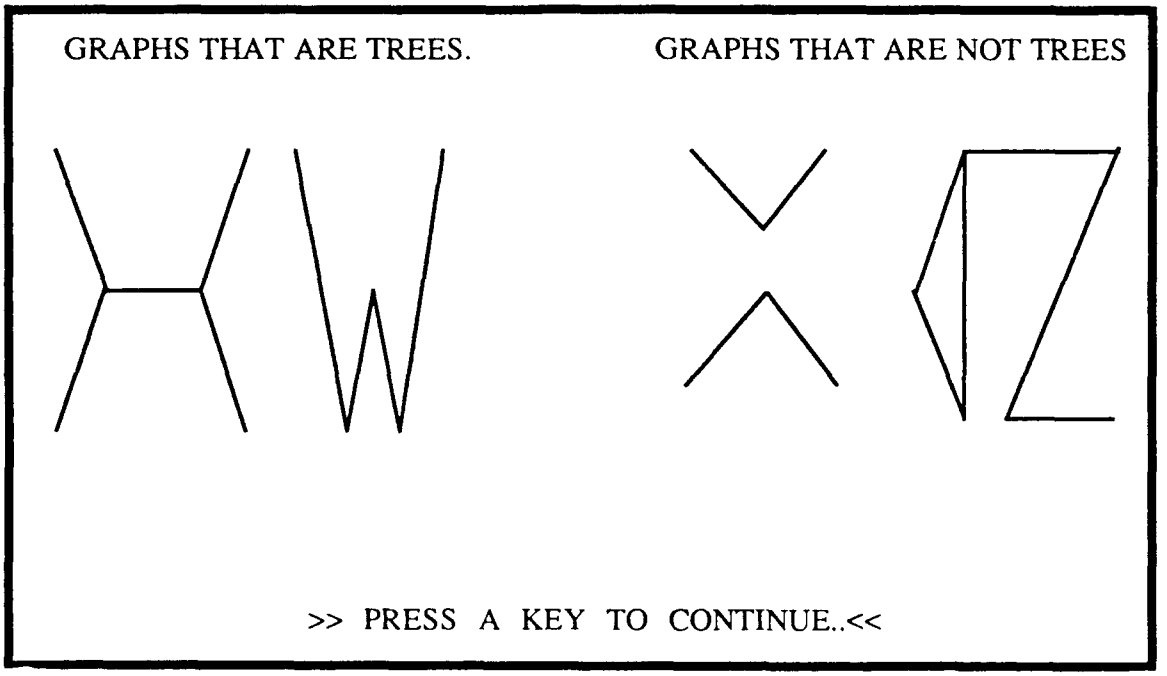


Figure 2. An Example of Graphics Windows.

VI. CONCLUSIONS AND RECOMMENDATIONS

This study achieves its primary goal of producing a tutorial for teaching "**undirected graphs**" in discrete math.

The software was designed to provide expendability and maintainability. The ease of expanding the tutorial will allow for modules to be added or deleted as the need arises. For example, the "**directed graphs**" module may be added to our "**undirected graphs**" module to complete the graphs segment.

Decomposing the entire program into separately executable modules creates an immense amount of code body and, consequently, a need for a large amount of storage is necessary. Ways to overcome this drawback should be examined.

With the existing windowing package, we are unable to present graphics within the text windows. In the future, another windowing package should be found, which allows graphics and text to be displayed on the same screen together. In this way, the programmer won't have to switch between graphics mode and text mode frequently, and won't have to include both the windowing library and the graphics libraries separately. Thus, the sizes of code bodies will be decreased significantly.

An implementation of a notepad with which the user can take notes during a session can be added as a feature to the DMT interface. This will further enhance the user friendliness of the tutorial by enabling the student to keep personally difficult items in an easily accessible place.

APPENDIX

LISTING OF THE SOURCE CODE

This section contains the program listings for the developed tutorial. The modules are listed in the sequential order in the tutorial.

```
/* PROGRAM  : grep.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains the tutorial for the graphs and their repre-
sentations.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/
/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)    /* MSC/QuickC */
```

```
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk           find_t
  #define ff_name         name
#elif defined(__ZTC__)                          /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name         name
  #define ff_attrib       attribute
#endif
#define _GRAPH_T DEFINED

/* function prototypes */
static void add_shadow   (void);
static void confirm_quit (void);
static void disp_sure_msg (void);
static void do_nothing   (void);
static void error_exit   (int errnum);
static void move_window  (int nsrow, int scol);
static void normal_exit  (void);
static void Pause        (void);
static void press_a_key  (int wrow);
static void quit_window  (void);
static void restore_cursor(void);
static void short_delay  (void);
static void size_window  (int nerow,int necol);
static void example_3_2  (void);
static void figure_3_2   (void);
static void figure_3_6   (void);
static void figure_3_7   (void);
static void figure_3_8   (void);
static void example_3_3  (void);
static void example_3_6  (void);
static void example_3_4  (void);
static void example_3_5  (void);
static void example_3_7  (void);
```

```
static void example_3_8  (void);
static void example_3_9  (void);
static void theorem_3_1  (void);
static void thm_3_1_pic  (void);
static void matrix_rep   (void);
static void theorem_3_2  (void);
static void adjacency_list(void);
static void matrix_ops   (void);
static void definition_3_1(void);
static void definition_3_2(void);
static void definition_3_3(void);
static void definition_3_4(void);
static void definition_3_5(void);
static void figure_3_2   (void);
static void figure_3_6   (void);
static void figure_3_7   (void);
static void figure_3_8   (void);
static void graph_rep (void);
static void exercise1(void);
static void exercise2(void);
static void exercise3(void);
static void exercise4(void);
static void exercise5(void);
static void Pexercise1(void);
static void Pexercise2(void);
static void Pexercise3(void);
static void Pexercise4(void);
static void Pexercise5(void);
static void initialize(void);
static void Pageup(void);
static void Pagedown(void);
static void pre_help(void);
static void Pgraph_rep(void);
static void Pdef_3_1(void);
static void Pex_3_2(void);
static void Pex_3_3(void);
```

```
static void Pex_3_4(void);
static void Pex_3_5(void);
static void Pex_3_6(void);
static void Pex_3_7(void);
static void Pex_3_8(void);
static void Pex_3_9(void);
static void following_fig1(void);
static void following_fig2(void);
static void following_fig3(void);
static void following_fig4(void);
static void following_fig5(void);
static void following_fig6(void);
static void following_fig7(void);
static void following_fig8(void);
static void following_fig9(void);
static void Pdef_3_2(void);
static void Pdef_3_3(void);
static void Pfig_3_6(void);
static void Pfig_3_7(void);
static void Pcont_grep(void);
static void Pcont2_grep(void);
static void cont_grep(void);
static void Pmatrix_rep(void);
static void cont2_grep(void);
static void cont_matrix_rep(void);
static void Pcont_matrix_rep(void);
static void Pmatrix_ops(void);
static void Padj_list(void);
static void Pdef_3_5(void);
static void Pthm_3_1(void);
static void Pthm_3_2(void);
/*************************************************************************/
/* miscellaneous global variables */
static int *savescrn,crow,ccol;
static WINDOW w[10];
static char ssan[10];
```

```
/*************************************************************************/
/* error message table */
static char *error_text[]= {
   NULL,  /* errnum = 0, no error   */
   NULL,  /* errnum == 1, windowing error */
   "Syntax:  CXLDEMO [-switches]\n\n"
      "\t -c = CGA snow elimination\n"
      "\t -b = BIOS screen writing\n"
      "\t -m = force monochrome text attributes",
   "Memory allocation error"
};
/* miscellaneous defines */
#define SHORT_DELAY 18
#define H_WINTITLE  33
/*************************************************************************/
/* this function will add a shadow to the active window                  */
/*************************************************************************/
static void add_shadow(void)
{
   wshadow(LGREY|_BLACK);
}
/*************************************************************************/
/* this function pops open a window and confirms that the user really    */
/* wants to quit the demo.  If so, it terminates the demo program.       */
/*************************************************************************/
static void confirm_quit(void)
{
   struct _onkey_t *kblist;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
   add_shadow();
   wputs("\n Quit demo, are you sure? \033A\156Y\b");
   clearkeys();
   showcur();
   if(wgetchf("YN",'Y')=='Y') normal_exit();
```

```
  wclose();
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
}



/*************************************************************************/
/* this function is called by the pull-down demo for a prompt            */
/*************************************************************************/
static void disp_sure_msg(void)
{
  wprints(0,2,WHITE|_BLUE,"Are you sure?");
}
/*************************************************************************/
/* this function is used as a dummy function for                         */
/* several menu items in the pull-down demo                              */
/*************************************************************************/
static void do_nothing(void)
{
}
/*************************************************************************/
/* this function handles abnormal termination.  If it is passed an       */
/* error code of 1, then it is a windowing system error.  Otherwise      */
/* the error message is looked up in the error message table.            */
/*************************************************************************/
static void error_exit(int errnum)
{
  if(errnum) {
    printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
    exit(errnum);
  }
}
```

```
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
/*************************************************************************/
/* this function displays a pause message then pauses for a keypress     */
/*************************************************************************/
static void press_a_key(int wrow)
{
   register int attr1;
   register int attr2;
   attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
   attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
   wcenters(wrow,attr1,"Press a key");
   wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
   hidecur();
   if(waitkey()==ESC) confirm_quit();
   wcenters(wrow,attr1,"          ");
   wprints(wrow,0,attr2,"        ");
}
/*************************************************************************/
static void short_delay(void)
{
   delay_(SHORT_DELAY);
}
```

```
/*************************************************************************/
/* this function is called by the pull-down menu demo anytime            */
/* the  selection bar moves on or off the [Q]uit menu items.             */
/*************************************************************************/
static void quit_window(void)
{
   static WINDOW handle=0;
   if(handle) {
      wactiv(handle);
      wclose();
      handle=0;
   }
   else {
      handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
      wputs(" Quit takes you back to the\n demo program's main menu.");
   }
}
/*************************************************************************/
static void restore_cursor(void)
{
   wtextattr(WHITE|_MAGENTA);
   showcur();
}
/*************************************************************************/
static void size_window(int nerow,int necol)
{
   wsize(nerow,necol);
   short_delay();
}
/*************************************************************************/
static void move_window(int nsrow,int nscol)
{
   if(wmove(nsrow,nscol)) error_exit(1);
   short_delay();
}
```

```
/*************************************************************************/

void main()
{
   register int *scrn;

   if((scrn=ssave())==NULL) error_exit(3);
   cclrscrn(LGREY|_BLUE);
   graph_rep();
   srestore(scrn);
}



/*************************************************************************/
static void initialize(void)
{
  /* initialize the CXL video system and save current screen info */
  videoinit();
  readcur(&crow,&ccol);
  if((savescrn=ssave())==NULL) error_exit(3);
  /* if mouse exists, turn on full mouse support */
  if(msinit()) {
     mssupport(MS_FULL);
     msgotoxy(12,49);
  }

  /* attach [Alt-X] to the confirm_quit() function */
  setonkey(0x2d00,confirm_quit,0);

  /* attach [Ctrl Pageup] to the Pageup() function */
  setonkey(0x8400,Pageup,0);

  /* attach [Ctrl Pagedown] to the Pagedown() function */
  setonkey(0x7600,Pagedown,0);
```

```
   /* initialize help system, help key = [F1] */
   whelpef("CXLDEMO.HLP",0x3b00,YELLOW|_RED,LRED|_RED,
            WHITE|_RED,RED|_LGREY ,pre_help);
}
/*************************************************************************/
static void pre_help(void)
{
   add_shadow();
   setonkey(0x2d00,confirm_quit,0);
}




/*************************************************************************/
/* this function is called anytime to switch back to previous window.    */
/*************************************************************************/
static void Pageup(void)
{
   static WINDOW handle;
   handle = whandle();
   wactiv(handle - 1);
}
/*************************************************************************/
/* this function is called anytime to switch back to next window.       */
/*************************************************************************/
static void Pagedown(void)
{
   static WINDOW handle;
   handle = whandle();
   wactiv(handle + 1);
}
```

```
/*************************************************************************/
static void graph_rep (void)
{
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  cclrscrn(LGREY|_BLUE);
  /*********************************************************************/
  /* attach [Pagedown] to the cont_grep function */
  setonkey(0x5100,Pcont_grep,0);
  /*********************************************************************/
   definition_3_1();
   srestore(scrn);
}

/*************************************************************************/
void definition_3_1(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the graph_rep() function */
  setonkey(0x4900,Pgraph_rep,0);
  /*********************************************************************/
  /* attach [Pagedown] to the cont_grep function */
  setonkey(0x5100,Pcont_grep,0);
  /*********************************************************************/
  if((w[1]=wopen(6,15,14,67,3,LCYAN|_GREEN,WHITE|_GREEN))==0)
     error_exit(1);
  wtitle("[Graph Representations - Definition_3_1]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" Def :  A graph is a pair G = (V, E) where:\n"
          " 1. V is a finite set whose elements are called vertices.\n"
          " 2. E is an irreflexive, symmetric relation on V, whose elements"
          " are called edges.");
  press_a_key(6);
  wcloseall();
  cont_grep();
 }
```

```
/*************************************************************************/
static void cont_grep(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the definition_3_1 function */
  setonkey(0x4900,Pdef_3_1,0);
  /*************************************************************************/
  /* attach [Pagedown] to the example_3_2 function */
  setonkey(0x5100,Pex_3_2,0);
  /*************************************************************************/
  if((w[1]=wopen(6,20,22,58,3,LCYAN|_BLUE,WHITE|_BLUE))==0) error_exit(1);
  wtitle("[Graph Theory]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" Graphs are among the most useful models within computer"
    " science, appearing in data flow models, state machines, scheduling"
    " algorithms, circuit layouts, communication networks, flowcharts and"
    " sorting and searching algorithms. Other applications outside of"
    " computer science include transportation networks, chemistry,"
    " process control systems, and the social sciences. At its core,"
    " graph theory is an extension of the theory of relations on a set.");
  press_a_key(14);
  wslide(1,1);
  if((w[2]=wopen(2,42,9,77,3,LCYAN|_RED,WHITE|_RED))==0) error_exit(1);
  whelpcat(H_WINTITLE);
  wputsw("Whenever we have an edge e={U,V), we say that the edge e connects"
         " or joins the vertices U and V and that U and V are adjacent.");
  press_a_key(5);
  wslide(17,3);
  if((w[3]=wopen(5,42,18,77,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
    error_exit(1);
  whelpcat(H_WINTITLE);
  wputsw  ("It is also said that edge i is incident"
           " on the vertex U and that the vertex U is"
           " incident on the edge e.\n"
           "With the graph in the following figure"
           " we see that vertices A and B are adjacent,"
           " whereas vertices A and C are not because"
```

```
                " there is no segment between them (that is,"
                " the set {A, C} is not an edge)");
    following_fig1();
}


/*************************************************************************/
void following_fig1(void)
{
   /*********************************************************************/
   /* attach [Pageup] to the cont_grep function */
   setonkey(0x4900,Pcont_grep,0);
   /*********************************************************************/
   /* attach [Pagedown] to the definition_3_2 function */
   setonkey(0x5100,Pdef_3_2,0);
   /*********************************************************************/
    if((w[3]=wopen(20,42,24,77,3,LCYAN|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Press a key to see the graph....");
    press_a_key(2);
    wcloseall();
    example_3_2();
}
/*************************************************************************/

  void example_3_2(void)
  {
     register int *scrn;
     if((scrn=ssave())==NULL) error_exit(3);
     spawnl(P_WAIT,"ex3_2.exe",NULL);
     srestore(scrn);
     definition_3_2();


  }
```

```
/*************************************************************************/
  void definition_3_2(void)
 {
  /*****************************************************************/
  /* attach [Pageup] to the example_3_2 function */
  setonkey(0x4900,Pex_3_2,0);
  /****** *********************************************************/
  /* attach [Pagedown] to the figure_3_6 function */
  setonkey(0x5100,Pfig_3_6,0);
  /********** ********************************************************/
  if((w[1]=wopen(2,3,8,72,3,LCYAN|_GREEN,WHITE|_GREEN))==0)
      error_exit(1);
  wtitle("[Graph Representations - Definition_3_2]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" Def :  Let G = (V, E) be a graph. Let U, T be the vertices."
            " The degree of T, denoted deg(T), is the number of edges incident"
            " to T. In the following figure we see that deg(A) = 1, deg(B) = 3,"
            " and deg(C) = 0.");
  following_fig2();
 }
/*************************************************************************/
void following_fig2(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the example_3_2 function */
  setonkey(0x4900,Pex_3_2,0);
  /*****************************************************************/
  /* attach [Pagedown] to the definition_3_3 function */
  setonkey(0x5100,Pdef_3_3,0);
  /*****************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wclose();
```

```
   figure_3_6();
}
/*************************************************************************/
 void figure_3_6(void)
 {
    register int *scrn;
    if((scrn=ssave())==NULL) error_exit(3);
    spawnl(P_WAIT,"fig3_6.exe",NULL);
    srestore(scrn);
    definition_3_3();
 }
/*************************************************************************/
 void definition_3_3(void)
 {
  /*********************************************************************/
  /* attach [Pageup] to the figure_3_6 function */
  setonkey(0x4900,Pfig_3_6,0);
  /*********************************************************************/
  /* attach [Pagedown] to the figure_3_7 function */
  setonkey(0x5100,Pfig_3_7,0);
  /*********************************************************************/
  if((w[2]=wopen(10,3,16,72,3,LCYAN|_GREEN,WHITE|_GREEN))==0)
    error_exit(1);
  wtitle("[Graph Representations - Definition_3_3]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" Def :  One special graph that is encountered frequently is the"
              " complete graph on n vertices, where every vertex is connected"
              " to every other vertex. This graph is denoted by Kn. The following"
              " figure shows K3 and K4.");
  following_fig3();
 }
```

```
/*************************************************************************/
void following_fig3(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the figure_3_6 function */
  setonkey(0x4900,Pfig_3_6,0);
  /*********************************************************************/
  /* attach [Pagedown] to the cont2_grep function */
  setonkey(0x5100,Pcont2_grep,0);
  /*********************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_GREEN,WHITE|_GREEN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wclose();
   figure_3_7();
}
/*************************************************************************/
 void figure_3_7(void)
 {
    register int *scrn;
    if((scrn=ssave())==NULL) error_exit(3);
    spawnl(P_WAIT,"fig3_7.exe",NULL);
    srestore(scrn);
    cont2_grep();


 }
```

```
/*************************************************************************/

void cont2_grep()
{
  /*********************************************************************/
  /* attach [Pageup] to the figure_3_7 function */
  setonkey(0x4900,Pfig_3_7,0);
  /*********************************************************************/
  /* attach [Pagedown] to the theorem_3_1 function */
  setonkey(0x5100,Pthm_3_1,0);
  /*********************************************************************/
   if((w[3]=wopen(18,3,24,72,3,WHITE|_MAGENTA,WHITE|_MAGENTA))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("In those figures notice that adding the degrees of the"
                  " vertices in each graph yields a number that is twice"
                  " the number of edges. This result is true in general.");
   press_a_key(4);
   wcloseall();
   theorem_3_1();
}
/*************************************************************************/
 void theorem_3_1(void)
 {
  /*********************************************************************/
  /* attach [Pageup] to the cont2_grep function */
  setonkey(0x4900,Pcont2_grep,0);
  /*********************************************************************/
  /* attach [Pagedown] to the matrix_rep function */
  setonkey(0x5100,Pmatrix_rep,0);
  /*********************************************************************/
  if((w[1]=wopen(2,4,6,71,3,LCYAN|_GREEN,WHITE|_LGREY))==0)
     error_exit(1);
  wtitle("[Graph Representations - Theorem_3_1]",
          TCENTER,_MAGENTA|WHITE);
  whelpcat(H_WINTITLE);
```

```
  wputs("\nTheorem_3_1\n");
  wputsw  ("In a graph the sum of the degrees of the vertices equals "
           "twice the number of edges.");
  delay_(20);
  if((w[2]=wopen(6,4,15,71,3,LCYAN|_RED,WHITE|_GREEN))==0) error_exit(1);
  wtitle("[Theorem_3_1 - Proof]",TCENTER,_MAGENTA|WHITE);
  whelpcat(H_WINTITLE);
  wputsw  (" The key to understanding why Theorem 3.1 is true is that "
           " each edge is incident on two vertices. When we take the"
           " sum of the degrees of the vertices, each edge is counted"
           " twice in this sum. Thus, the sum of the degrees is twice"
           " the number of the edges.");
  press_a_key(7);
  wclose();
  wclose();
  matrix_rep();
 }
/*************************************************************************/
void matrix_rep(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the theorem_3_1 function */
  setonkey(0x4900,Pthm_3_1,0);
  /*****************************************************************/
  /* attach [Pagedown] to the example_3_4 function */
  setonkey(0x5100,Pex_3_4,0);
  /*****************************************************************/
  if((w[1]=wopen(3,4,14,71,3,LCYAN|_GREEN,WHITE|_LGREY))==0)
    error_exit(1);
  wtitle("[Matrix  Representations]",TCENTER,_MAGENTA|WHITE);
  whelpcat(H_WINTITLE);
  wputsw("It is often necessary to analyze graphs and perform a variety"
         " of procedures and algorithms upon them. When they have large"
         " numbers of vertices and edges, it is often essential to use"
         " a computer to perform these algorithms. Thus, it is necessary"
         " to communicate to the computer the vertices and edges of a"
```

```
                " graph. One way to do so is to represent a graph by means of"
                " numbers, which are easy to store and manipulate.\n  An m x n"
                " matrix is rectangular array of numbers in which there are m"
                " horizontal rows and n vertical columns.");
    press_a_key(9);
    following_fig4();
}
/*************************************************************************/
void following_fig4(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the matrix_rep function */
  setonkey(0x4900,Pmatrix_rep,0);
  /*********************************************************************/
  /* attach [Pagedown] to the example_3_3 function */
  setonkey(0x5100,Pex_3_3,0);
  /*********************************************************************/
   if((w[3]=wopen(20,42,24,77,3,MAGENTA|_CYAN,WHITE|_CYAN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_4();
}
/*************************************************************************/
static void example_3_4(void)
{
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_4.exe",NULL);
  srestore(scrn);
  example_3_3();

}
```

```
/*************************************************************************/
void example_3_3(void)
{
  /*************************************************************/
  /* attach [Pageup] to the example_3_4 function */
  setonkey(0x4900,Pex_3_4,0);
  /*************************************************************/
  /* attach [Pagedown] to the example_3_5 function */
  setonkey(0x5100,Pex_3_5,0);
  /*************************************************************/
  if((w[3]=wopen(9,4,17,71,3,YELLOW|_BLUE,YELLOW|_BLUE))==0)
    error_exit(1);
  whelpcat(H_WINTITLE);
  wputsw("Two matrices A and B are said to be equal whenever A and B"
         " have the same number of rows and the same number of columns,"
         " and the i,j entry of A  is equal to the i,j entry of B  for"
         " all possible i and j. In  other words, two matrices are equal"
         " when they have the same size and all pairs of corresponding"
         " entries are equal.");
    following_fig5();
}
/*************************************************************************/
void following_fig5(void)
{
  /*************************************************************/
  /* attach [Pageup] to the example_3_4 function */
  setonkey(0x4900,Pex_3_4,0);
  /*************************************************************/
  /* attach [Pagedown] to the cont_matrix_rep function */
  setonkey(0x5100,Pcont_matrix_rep,0);
  /*************************************************************/
   if((w[3]=wopen(20,42,24,77,3,GREEN|_BROWN,YELLOW|_BROWN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
```

```
    wcloseall();
    example_3_5();
}
/*************************************************************************/
 void example_3_5(void)
{
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   spawnl(P_WAIT,"ex3_5.exe",NULL);
   srestore(scrn);
   cont_matrix_rep();
}
/*************************************************************************/
void cont_matrix_rep(void)
{
   /*********************************************************************/
   /* attach [Pageup] to the example_3_5 function */
   setonkey(0x4900,Pex_3_5,0);
   /*********************************************************************/
   /* attach [Pagedown] to the example_3_6 function */
   setonkey(0x5100,Pex_3_6,0);
   /*********************************************************************/
   if((w[2]=wopen(8,4,16,71,3,LCYAN|_GREEN,WHITE|_LGREY))==0)
      error_exit(1);
  wtitle("[Labeled Graph and Adjacency Matrix]",TCENTER,_MAGENTA|WHITE);
   whelpcat(H_WINTITLE);
   wputsw("Suppose we have a graph G with n vertices labeled V1, V2,"
     " ...,Vn. Such a graph  is called a labeled graph. To represent"
     " the labeled graph G by a  matrix, we form an n x n matrix in"
     " which the i,j element is 1  if there is an edge between the"
     " vertices Vi and Vj and 0 if there is not. This matrix is called"
     " the adjacency matrix of G (with respect to the labeling) and is"
     " denoted by A(G).");
   press_a_key(6);
   following_fig6();
}
```

```
/*************************************************************************/
void following_fig6(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the cont_matrix_rep function */
  setonkey(0x4900,Pcont_matrix_rep,0);
  /*********************************************************************/
  /* attach [Pagedown] to the theorem_3_2 function */
  setonkey(0x5100,Pthm_3_2,0);
  /*********************************************************************/
   if((w[3]=wopen(20,42,24,77,3,MAGENTA|_LGREY,WHITE|_LGREY))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_6();
}

/*************************************************************************/

void example_3_6(void)
{
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_6.exe",NULL);
  srestore(scrn);
  theorem_3_2();

}
```

```
/*************************************************************************/
 void theorem_3_2(void)
 {
  /*********************************************************************/
  /* attach [Pageup] to the example_3_6 function */
  setonkey(0x4900,Pex_3_6,0);
  /*********************************************************************/
  /* attach [Pagedown] to the adjacency_list function */
  setonkey(0x5100,Padj_list,0);
  /*********************************************************************/
  if((w[1]=wopen(3,4,8,71,3,LCYAN|_GREEN,WHITE|_LGREY))==0)
    error_exit(1);
  wtitle("[Graph Representations - Theorem_3_2]",
         TCENTER,_MAGENTA|WHITE);
  whelpcat(H_WINTITLE);
  wputs("\nTheorem_3_2\n");
  wputsw(" The sum of the entries in row i of the adjacency matrix of"
         " \n a graph is the degree of the vertex Vi in the graph. ");
  delay_(20);
  if((w[2]=wopen(10,4,17,71,3,LCYAN|_RED,WHITE|_GREEN))==0)
    error_exit(1);
  wtitle("[Theorem_3_2 - Proof]",TCENTER,_MAGENTA|WHITE);
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" We recall that each 1 in row i corresponds to an edge on the"
         "\n vertex Vi. Thus the number of 1's in row i is the number of"
         "\n edges  on Vi, which is the degree of Vi.");
  press_a_key(5);
  wclose();
  wclose();
  adjacency_list();
 }
```

```
/*************************************************************************/
static void adjacency_list(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the theorem_3_2 function */
  setonkey(0x4900,Pthm_3_2,0);
  /*********************************************************************/
  /* attach [Pagedown] to the example_3_7 function */
  setonkey(0x5100,Pex_3_7,0);
  /*********************************************************************/
  if((w[1]=wopen(3,4,15,40,3,LCYAN|_RED,WHITE|_GREEN))==0) error_exit(1);
  wtitle("[Adjacency List]",TCENTER,_MAGENTA|WHITE);
  whelpcat(H_WINTITLE);
  wputsw("Matrices are not the only way to represent graphs in a computer."
         " Although an adjacency matrix is easy to construct, this form"
         " of representation requires more storage for a graph with n"
         " vertices and can be quite inefficient if the matrix contains"
         " lots of zeros.");
  wslide(12,40);
  wslide(3,4);
  press_a_key(10);
  if((w[2]=wopen(1,42,20,74,3,LRED|_CYAN,WHITE|_CYAN))==0) error_exit(1);
  wtitle("[Adjacency List]",TCENTER,_MAGENTA|WHITE);
  whelpcat(H_WINTITLE);
  wputsw("A better representation for such a matrix is an adjacency list"
         " The basic idea of an adjacency list is to list each vertex"
         " followed by the vertices adjacent to it. To form the adjacency"
         " list, we begin by labeling the vertices of the graph G. Then we"
         " list the vertices in a vertical column, and after each one we"
         " write down the adjacent vertices. Thus we see that the 1's in"
         " a row of an adjacency matrix tell what vertices are listed in"
         " the corresponding row of an adjacency list.");
  wslide(6,2);
  wslide(2,42);
  following_fig7();
}
```

```
/*************************************************************************/
void following_fig7(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the adjacency_list function */
  setonkey(0x4900,Padj_list,0);
  /*********************************************************************/
  /* attach [Pagedown] to the matrix_ops function */
  setonkey(0x5100,Pmatrix_ops,0);
  /*********************************************************************/
   if((w[3]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_7();
}
/*************************************************************************/
static void example_3_7(void)
{
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_7.exe",NULL);
  srestore(scrn);
  matrix_ops();
}
/*************************************************************************/
static void matrix_ops(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the example_3_7 function */
  setonkey(0x4900,Pex_3_7,0);
  /*********************************************************************/
  /* attach [Pagedown] to the example_3_8 function */
  setonkey(0x5100,Pex_3_8,0);
```

```
   definition_3_4(); /* addition */
}
/*************************************************************************/
static void definition_3_4(void)
{
   /*********************************************************************/
   /* attach [Pageup] to the example_3_7 function */
   setonkey(0x4900,Pex_3_7,0);
   /*********************************************************************/
   /* attach [Pagedown] to the example_3_8 function */
   setonkey(0x5100,Pex_3_8,0);
   /*********************************************************************/
   if((w[1]=wopen(3,4,11,73,3,LCYAN|_RED,WHITE|_GREEN))==0) error_exit(1);
   wtitle("[Addition of Matrices]",TCENTER,_MAGENTA|WHITE);
   whelpcat(H_WINTITLE);
   wputsw("Suppose A and B are two m x n matrices. The sum of A and B,"
          " denoted by A + B, is the m x n matrix in which the i,j entry"
          " of A + B is the sum of the i,j entry of A and the i,j entry"
          " of B. In other words, we add two matrices of the same size"
          " by adding together the corresponding entries. Thus, only"
          " matrices of the same size can be added.");
   press_a_key(6);
   following_fig8();
}
/*************************************************************************/
void following_fig8(void)
{
   /*********************************************************************/
   /* attach [Pageup] to the matrix_ops function */
   setonkey(0x4900,Pmatrix_ops,0);
   /*********************************************************************/
   /* attach [Pagedown] to the definition_3_5 function */
   setonkey(0x5100,Pdef_3_5,0);
   /*********************************************************************/
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_MAGENTA,
       WHITE|_MAGENTA))==0)  error_exit(1);
```

```
    whelpcat(H_WINTITLE);
    wputsw("Press a key to see the graph....");
    press_a_key(2);
    wcloseall();
    example_3_8();
}
/*************************************************************************/
static void example_3_8(void)
{
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   spawnl(P_WAIT,"ex3_8.exe",NULL);
   srestore(scrn);
   definition_3_5(); /* multiplication */
}
/*************************************************************************/
static void definition_3_5(void)
{
   /*********************************************************************/
   /* attach [Pageup] to the example_3_8 function */
   setonkey(0x4900,Pex_3_8,0);
   /*********************************************************************/
   /* attach [Pagedown] to the example_3_9 function */
   setonkey(0x5100,Pex_3_9,0);
   /*********************************************************************/
   if((w[1]=wopen(11,4,20,73,3,LCYAN|_RED,WHITE|_GREEN))==0)
      error_exit(1);
   wtitle("[Multiplication of Matrices]",TCENTER,_MAGENTA|WHITE);
   whelpcat(H_WINTITLE);
   wputsw("Suppose  A is an m x k matrix with the i,j entry represented"
                  " by aij and B is a k x n matrix with the i,j entry represented"
                  " by bij. Then the product of A and B, denoted by AB, is the"
                  " m x n matrix where the i,j  entry is given by\n"
                  "     ai1b1j + a12b2j + ... + aikbkj."
                  "\nNote that the number of columns of A is required to be equal"
                  " to the number of rows of B in order to multiply A and B.");
```

```
   press_a_key(7);
   following_fig9();
}
/*************************************************************************/
void following_fig9(void)
{
   /*********************************************************************/
   /* attach [Pageup] to the definition_3_5 function */
   setonkey(0x4900,Pdef_3_5,0);
   /*********************************************************************/
   /* attach [Pagedown] to the exercise1 function */
   setonkey(0x5100,Pexercise1,0);
   /*********************************************************************/
    if((w[3]=wopen(20,42,24,77,3,MAGENTA|_GREEN,WHITE|_GREEN))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Press a key to see the graph....");
    press_a_key(2);
    wcloseall();
    example_3_9();
}
/*************************************************************************/

static void example_3_9(void)
{
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   spawnl(P_WAIT,"ex3_9.exe",NULL);
   srestore(scrn);
   exercise1();

}
```

```
/*************************************************************************/
void exercise1(void)
{
  register int *scrn;
  /*************************************************************************/
  /* attach [Pageup] to the example_3_9 function */
  setonkey(0x4900,Pex_3_9,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exercise2 function */
  setonkey(0x5100,Pexercise2,0);
  /*************************************************************************/
  if((scrn=ssave())==NULL) error_exit(3);
   if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Now let us see some exercises...");
   press_a_key(2);
   wcloseall();
   spawnl(P_WAIT,"q1.exe",NULL);
   srestore(scrn);
   exercise2();
}
/*************************************************************************/
void exercise2(void)
{
  register int *scrn;
  /*************************************************************************/
  /* attach [Pageup] to the exercise1 function */
  setonkey(0x4900,Pexercise1,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exercise3 function */
  setonkey(0x5100,Pexercise3,0);
  /*************************************************************************/
  if((scrn=ssave())==NULL) error_exit(3);
   if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
```

```
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 2..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"q2.exe",NULL);
    srestore(scrn);
   exercise3();
}
/*************************************************************************/
void exercise3(void)
{
   register int *scrn;
   /*************************************************************************/
   /* attach [Pageup] to the exercise2 function */
   setonkey(0x4900,Pexercise2,0);
   /*************************************************************************/
   /* attach [Pagedown] to the exercise4 function */
   setonkey(0x5100,Pexercise4,0);
   /*************************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 3..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT "q3.exe",NULL);
    srestore(scrn);
   exercise4();
}
/*************************************************************************/
void exercise4(void)
{
   register int *scrn;
   /*************************************************************************/
   /* attach [Pageup] to the exercise3 function */
```

```
  setonkey(0x4900,Pexercise3,0);
  /*****************************************************************/
  /* attach [Pagedown] to the exercise5 function */
  setonkey(0x5100,Pexercise5,0);
  /*****************************************************************/
  if((scrn=ssave())==NULL) error_exit(3);
   if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Now let us see exercise 4..");
   press_a_key(2);
   wcloseall();
   spawnl(P_WAIT,"q4.exe",NULL);
   srestore(scrn);
  exercise5();
}
/*****************************************************************************/
void exercise5(void)
{
  register int *scrn;
  /*****************************************************************/
  /* attach [Pageup] to the exercise4 function */
  setonkey(0x4900,Pexercise4,0);
  /*****************************************************************/
  /* attach [Pagedown] to the exercise5 function */
  setonkey(0x5100,Pexercise5,0);
  /*****************************************************************/
  if((scrn=ssave())==NULL) error_exit(3);
   if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Now let us see exercise 5..");
   press_a_key(2);
   wcloseall();
   spawnl(P_WAIT,"q5.exe",NULL);
   srestore(scrn);
```

```
  normal_exit();
}
/*************************************************************************/
/* this routine that calls definition_3_1 routine whenever Pageup or Pagedown  */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_1()
{
  wcloseall();
  definition_3_1();
}
/*************************************************************************/
/* this routine calls graph_rep routine whenever Pageup key is pressed.  */
/*************************************************************************/
void Pgraph_rep()
{
  wcloseall();
  graph_rep();
}
/*************************************************************************/
/* this routine calls example_3_2 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_2()
{
  wcloseall();
  following_fig1();
}
/*************************************************************************/
/* this routine calls definition_3_2 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_2()
{
  wcloseall();
  definition_3_2();}
```

```
/*************************************************************************/
/* this routine calls figure_3_6 routine whenever Pageup or Pagedown     */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pfig_3_6()
{
   wcloseall();
   following_fig2();
}
/*************************************************************************/
/* this routine calls figure_3_7 routine whenever Pageup or Pagedown     */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pfig_3_7()
{
   wcloseall();
   following_fig3();
}
/*************************************************************************/
/* this routine calls definition_3_3 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_3()
{
   wcloseall();
   definition_3_3();
}
/*************************************************************************/
/* this routine calls theorem_3_1 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pthm_3_1()
{
   wcloseall();
   theorem_3_1();
}
```

```
/*************************************************************************/
/* this routine calls matrix_rep routine whenever Pageup or Pagedown     */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pmatrix_rep()
{
  wcloseall();
  matrix_rep();
}
/*************************************************************************/
/* this routine calls cont_matrix_rep routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pcont_matrix_rep()
{
  wcloseall();
  cont_matrix_rep();
}
/*************************************************************************/
/* this routine calls example_3_3 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_3()
{
  wcloseall();
  example_3_3();
}
/*************************************************************************/
/* this routine calls example_3_4 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_4()
{
  wcloseall();
  following_fig4();
}
```

```
/*************************************************************************/
/* this routine calls example_3_5 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_5()
{
   wcloseall();
   following_fig5();
}
/*************************************************************************/
/* this routine calls example_3_6 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_6()
{
   wcloseall();
   following_fig6();
}
/*************************************************************************/
/* this routine calls example_3_7 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_7()
{
   wcloseall();
   following_fig7();
}
/*************************************************************************/
/* this routine calls example_3_8 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_8()
{
   wcloseall();
    following_fig8();
}
```

```
/*************************************************************************/
/* this routine calls example_3_9 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_9()
{
   wcloseall();
    following_fig9();
}
/*************************************************************************/
/* this routine calls theorem_3_2 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pthm_3_2()
{
   wcloseall();
   theorem_3_2();
}
/*************************************************************************/
/* this routine calls adjacency_list routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Padj_list()
{
   wcloseall();
   adjacency_list();
}
/*************************************************************************/
/* this routine calls matrix_ops routine whenever Pageup or Pagedown     */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pmatrix_ops()
{
   wcloseall();
   matrix_ops();
}
```

```
/*************************************************************************/
/* this routine calls definition_3_4 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_4()
{
   wcloseall();
   definition_3_4();
}
/*************************************************************************/
/* this routine calls definition_3_5 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_5()
{
   wcloseall();
   definition_3_5();
}
/*************************************************************************/
/* this routine calls exercise1 routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pexercise1()
{
   wcloseall();
   exercise1();
}
/*************************************************************************/
/* this routine calls exercise2 routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pexercise2()
{
   wcloseall();
   exercise2();
}
```

```
/*************************************************************************/
/* this routine calls exercise3 routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pexercise3()
{
   wcloseall();
   exercise3();
}
/*************************************************************************/
/* this routine calls exercise4 routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pexercise4()
{
   wcloseall();
   exercise4();
}
/*************************************************************************/
/* this routine calls exercise5 routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pexercise5()
{
   wcloseall();
   exercise5();
}
/*************************************************************************/
/* this routine calls cont_grep routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pcont_grep()
{
   wcloseall();
   cont_grep();
}
```

```
/*************************************************************************/
/* this routine calls cont2_grep routine whenever Pageup or Pagedown     */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pcont2_grep()
{
   wcloseall();
   cont2_grep();
}
```

```
/* PROGRAM  : ex3_2.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

/*************************************************************************

                GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS

 *************************************************************************/

static int *savescrn,crow,ccol;
char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
```

```
/*************************************************************************
            FUNCTION DEFINITIONS
*************************************************************************/

void wait(char title[]);
static void example_3_2 (void);
static void fig_3_3 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
          /*****************************************************
                                MAIN PROGRAM
          *****************************************************/
main()
{
         cls (1);
          adapt = getAdapter();
          if (adapt == 'V'){
              setMode(0x12);
              cls(1);
          }
          if (adapt == 'E'){
              setMode(16);
              cls(1);
          }
          if (adapt == 'C'){
              setMode(4);
              setCGAPalette(0);
               cls(1);
              }
          example_3_2();
          fig_3_3();
       setMode(3);

}
```

```
/*************************************************************************/
static void example_3_2(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    drawLine(-150,100,150,100,11);
    drawLine(150,100,-150,-100,11);
    drawLine(-150,-100,150,-100,11);
    aspect = 1.0;
    fillOval(-150,100,2,16,aspect);
    fillOval(150,100,2,16,aspect);
    fillOval(-150,-100,2,16,aspect);
    fillOval(150,-100,2,16,aspect);
    write_horz_char(-160,105,65,31);
    write_horz_char(160,105,66,31);
    write_horz_char(-160,-105,67,31);
    write_horz_char(160,-105,68,31);
    write_horz_str(-160,-145," A graph with vertices A, B, C, and D",31);
    write_horz_str(-160,-165," and edges {A, B}, {B, C}, and {C, D}",31);
    wait("");

}


/*************************************************************************/
static void fig_3_3(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    drawLine(-150,100,150,100,11);
    drawLine(150,100,-150,0,11);
    drawLine(-150,0,150,0,11);
    drawLine(150,0,-150,-100,11);
    drawLine(-150,-100,150,-100,11);
```

```
    aspect = 1.0;
    fillOval(-150,100,2,16,aspect);
    fillOval(150,100,2,16,aspect);
    fillOval(-150,0,2,16,aspect);
    fillOval(150,0,2,16,aspect);
    fillOval(-150,-100,2,16,aspect);
    fillOval(150,-100,2,16,aspect);
    write_horz_char(-160,100,65,31);    /* A */
    write_horz_char(155,100,68,31);     /* D */
    write_horz_char(-160,0,66,31);      /* B */
    write_horz_char(155,0,69,31);       /* E */
    write_horz_char(-160,-100,67,31);   /* C */
    write_horz_char(155,-100,70,31);    /* F */
    write_horz_str(-180,-145," A graph with vertices A, B, C, D,E and F",31);
    write_horz_str(-180,-165," and edges {A, D}, {B, D}, {B, E},{C, E},
                    and {C, F}.",31);
    wait("");
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
      line = 29;
    else
      line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
   write_horz_str(30,-210," Please type y or n",79);
   ch = getch ();
    write_horz_str(30,-210,"                ",31);
}
    switch (ch)         {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'Y': setMode(3);
               videoinit();
               normal_exit();
               break;

     case 'n':write_horz_str(-300,-210,"                          ",31);
               break;
     case 'N':write_horz_str(-300,-210,"                          ",31);
               break;
     default : break;
    }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_4.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

     DESCRIPTION : This program contains an example about the "Graphs and
their Representations".

    MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

/*************************************************************************
                GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
*************************************************************************/



char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void open_matrix_paren(int x1, int y1,int x2,int y2,int color);
static void example_3_4 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*********************************************************
                            MAIN PROGRAM
        *********************************************************/

main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V'){
        setMode(0x12);
        cls(1);
     }
     if (adapt == 'E'){
        setMode(16);
        cls(1);}
     if (adapt == 'C'){
        setMode(4);
        setCGAPalette(0);
        cls(1);}
     example_3_4();
     setMode(3);

}
```

```
/*************************************************************************/
static void example_3_4(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    open_matrix_paren(-163,85,-100,20,11);
    open_matrix_paren(60,85,180,0,11);

    write_horz_char(-147,76,49,31);  /* write a11 */
    write_horz_char(-124,76,50,31);  /* write a12 */
    write_horz_char(-147,57,53,31);  /* write a21 */
    write_horz_char(-124,57,48,31);  /* write a22 */
    write_horz_char(-147,38,54,31);  /* write a31 */
    write_horz_char(-124,38,55,31);  /* write a32 */

    write_horz_char(88,76,49,31);    /* write b11 */
    write_horz_char(121,76,50,31);   /* write b12 */
    write_horz_char(154,76,49,31);   /* write b13 */
    write_horz_char(88,57,51,31);    /* write b21 */
    write_horz_char(121,57,52,31);   /* write b22 */
    write_horz_char(154,57,48,31);   /* write b23 */
    write_horz_char(88,38,53,31);    /* write b31 */
    write_horz_char(121,38,56,31);   /* write b32 */
    write_horz_char(154,38,49,31);   /* write b33 */
    write_horz_char(88,19,55,31);    /* write b41 */
    write_horz_char(121,19,56,31);   /* write b42 */
    write_horz_char(154,19,51,31);   /* write b43 */

    write_horz_char(-195,57,65,31);  /* write A */
    write_horz_char(-180,57,61,31);  /* write = */
    write_horz_char(30,50,66,31);    /* write B */
    write_horz_char(40,50,61,31);    /* write = */
    write_horz_str(-300,155," Array A below is a 3 x 2 matrix"
       " because there are 3 horizontal rows and",31);
     write_horz_str(-300,140," 2 vertical columns of numbers."
                              " In the same way, B is a 4 x 3 matrix.",31);
     write_horz_str(-300,-30," The number in the ith row and"
                              " jth column is called the i,j entry of the",31);
     write_horz_str(-300,-45," matrix. Thus, in the matrix A the 2,1"
                              " entry is 5, because the entry lying",31);
```

```
    write_horz_str(-300,-60," in row 2 and column 1 is 5. For matrix B the 3,2
                    entry is 9.",31);
    wait("");
}


/*************************************************************************/


void open_matrix_paren(int x1,int y1,int x2,int y2,int color)
{

    extern unsigned long int PATTERN;
    drawLine (x1 ,y1,x1+5,y1,color);
    drawLine (x1 ,y1,x1,y2,color);
    drawLine (x1 ,y2,x1+5,y2,color);
    drawLine (x2 ,y1,x2-5,y1,color);
    drawLine (x2 ,y1,x2,y2,color);
    drawLine (x2,y2,x2-5,y2,color);
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
        line = 29;
    else
        line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
  write_horz_str(30,-210," Please type y or n",79);
  ch = getch ();
  write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
              videoinit();
              normal_exit();
              break;
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;
    case 'n':write_horz_str(-300,-210,"                          ",31);
              break;
    case 'N':write_horz_str(-300,-210,"                          ",31);
              break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : fig3_6.c
   AUTHOR       : Yavuz BAS
   DATE         : Feb. 4, 1990
   REVISED      : Mar. 5, 1990

   DESCRIPTION  : This program contains an example about the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescn,crow,ccol;
```

```
/*****************************************************
                FUNCTION DEFINITIONS
 *****************************************************/

void wait(char title[]);
static void fig_3_6 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

/*****************************************************
                MAIN PROGRAM
 *****************************************************/

main()
{
    cls (1);
    adapt = getAdapter();
    if (adapt == 'V')
       {
         setMode(0x12);
         cls(1);
       }
    if (adapt == 'E')
       {
         setMode(16);
         cls(1);
       }
   if (adapt == 'C')
       {
         setMode(4);
         setCGAPalette(0);
         cls(1);
       }
   fig_3_6();
   setMode(3);
}
```

```
/*************************************************************************/
static void fig_3_6(void)
{
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);
     drawLine(0,100,0,-100,11);
     drawLine(0,100,-150,-100,11);
     drawLine(0,100,150,-100,11);

     aspect = 1.0;
     fillOval(0,100,3,16,aspect);
     fillOval(0,-100,3,16,aspect);
     fillOval(-150,-100,3,16,aspect);
     fillOval(150,-100,3,16,aspect);
     fillOval(180,100,3,16,aspect);

     write_horz_char(165,105,67,31); /* point c*/
     write_horz_char(0,120,66,31);   /* point B */
     write_horz_char(-155,-105,65,31);  /* point A */
     write_horz_char(155,-105,68,31);   /* point D */
     write_horz_char(-5,-105,69,31);    /* point E */

     write_horz_str(-150,-130,"deg (A) =  1   deg (B) = 3   deg (C) = 0",31);
     wait("");
}
/*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
       line = 29;
     else
       line = 24;
     tab = (width - strlen(title))/2;
```

```
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'Y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'n':write_horz_str(-300,-210,"                          ",31);
           break;

    case 'N':write_horz_str(-300,-210,"                          ",31);
           break;
```

```
    default : break;
   }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : fig3_7.c
   AUTHOR     : Yavuz BAS
   DATE       : Feb. 4, 1990
   REVISED    : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
            GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
         *************************************************/

void wait(char title[]);
static void fig_3_7 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                        MAIN PROGRAM
         *************************************************/

main()
{
     adapt = getAdapter();
     if (adapt == 'V')
        {
          setMode(0x12);
          cls(1);
        }
     if (adapt == 'E')
        {
          setMode(16);
          cls(1);
        }
     if (adapt == 'C')
        {
          setMode(4);
          setCGAPalette(0);
          cls(1);
        }
    fig_3_7();
   setMode(3);

}
```

```
/*************************************************************************/
 static void fig_3_7(void)
{
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);
     drawRect(10,100,250,-100,11);
     drawLine(10,100,250,-100,11);       /*  diagonals of   */
     drawLine(250,100,10,-100,11);       /*  the rectangle  */
     drawLine(-105,120,-190,-100,11);  /******************/
     drawLine(-190,-100,-20,-100,11);  /* the triangle        */
     drawLine(-20,-100,-105,120,11);   /******************/
     aspect = 1.0;
     fillOval(10,100,3,16,aspect);
     fillOval(250,100,3,16,aspect);
     fillOval(250,-100,3,16,aspect);
     fillOval(10,-100,3,16,aspect);
     fillOval(-105,120,3,16,aspect);
     fillOval(-190,-100,3,16,aspect);
     fillOval(-20,-100,3,16,aspect);
     write_horz_char(-108,-105,75,31); /*  K */
     write_horz_char(-98,-115,51,31);  /* 3 */
     write_horz_char(127,-105,75,31);  /* K */
     write_horz_char(137,-115,52,31);  /* 4 */
     wait("");
}
/*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
        line = 29;
     else
        line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
```

```
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
/*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL); /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);    }
   switch (ch)        {
    case 'y': setMode(3);
          videoinit();
          normal_exit();
          break;
    case 'Y': setMode(3);
          videoinit();
          normal_exit();
          break;
    case 'n':write_horz_str(-300,-210,"                          ",31);
          break;
    case 'N':write_horz_str(-300,-210,"                          ",31);
          break;
    default : break;     }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
```

```
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM   : ex3_5.c
   AUTHOR    : Yavuz BAS
   DATE      : Feb. 4, 1990
   REVISED   : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Graphs and
                 their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

/*************************************************************************
                GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
 *************************************************************************/



char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
static int *savescrn,crow,ccol;
```

```
/*************************************************************************
                        FUNCTION DEFINITIONS
*************************************************************************/
void wait(char title[]);
void open_matrix_paren(int x1, int y1,int x2,int y2,int color);
static void example_3_5 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
 /************************************************************************
                            MAIN PROGRAM
 ************************************************************************/
main()
{
      cls (1);
      adapt = getAdapter();
       if (adapt == 'V')
         {
           setMode(0x12);
           cls(1);
         }
       if (adapt == 'E')
         {
           setMode(16);
           cls(1);
         }
       if (adapt == 'C')
         {
           setMode(4);
           setCGAPalette(0);
           cls(1);
         }
     example_3_5();
     setMode(3);
 }
```

```
/*************************************************************************** /

static void example_3_5(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    open_matrix_paren(-60,65,30,32,11);  /* paren for matrix D */
    open_matrix_paren(-233,65,-170,32,11); /*paren for mat C */
    open_matrix_paren(162,65,225,32,11); /*paren for matrix E*/

    write_horz_char(-213,64,49,31);  /* write c11 */
    write_horz_char(-193,64,50,31);  /* write c12 */
    write_horz_char(-213,44,51,31);  /* write c21 */
    write_horz_char(-193,44,52,31);  /* write c22 */

    write_horz_char(-40,64,49,31);  /* write d11 */
    write_horz_char(-20,64,50,31);  /* write d12 */
    write_horz_char(0,64,48,31);  /* write d13 */
    write_horz_char(-40,44,51,31);  /* write d21 */
    write_horz_char(-20,44,52,31);  /* write d22 */
    write_horz_char(0,44,48,31);  /* write d23 */

    write_horz_char(182,64,49,31);  /* write e11 */
    write_horz_char(202,64,51,31);  /* write e12 */
    write_horz_char(182,44,50,31);  /* write e21 */
    write_horz_char(202,44,52,31);  /* write e22 */

    write_horz_char(-265,55,67,31);  /* write C */
    write_horz_char(-250,55,61,31);  /* write = */
    write_horz_char(-90,55,68,31);  /* write D */
    write_horz_char(-75,55,61,31);  /* write = */
    write_horz_char(130,55,69,31);  /* write E */
    write_horz_char(145,55,61,31);  /* write = */
    write_horz_str(-300,175," Two matrices A and B are said to be "
                    "equal whenever A and B have the same",31);
    write_horz_str(-300,160," number of rows and"
                    " the same number of columns, and the i,j entry",31);
    write_horz_str(-300,145," of A is equal to the i,j entry of B"
                     " for all possible i and j. ",31);
    write_horz_str(-300,130,"    In other words, two matrices are equal"
                  "       when they have the same size",31);
```

```
    write_horz_str(-300,115," and all pairs of corresponding entries are equal.",31);
    write_horz_str(-300,-30," The matrix C is not equal to the matrix D"
                    " because C has two columns and D ",31);
    write_horz_str(-300,-45," has 3 columns. Also the matrices"
                    " C and E are not equal because their",31);
    write_horz_str(-300,-60," corresponding entries are not the same."
    "                " In particular, the 1,2 entries are",31);
    write_horz_str(-300,-75," different.",31);

     wait("");


}

/*************************************************************************/


void open_matrix_paren(int x1,int y1,int x2,int y2,int color)
{

    extern unsigned long int PATTERN;
    drawLine (x1 ,y1,x1+5,y1,color);
    drawLine (x1 ,y1,x1,y2,color);
    drawLine (x1 ,y2,x1+5,y2,color);
    drawLine (x2 ,y1,x2-5,y1,color);
    drawLine (x2 ,y1,x2,y2,color);
    drawLine (x2,y2,x2-5,y2,color);
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
       line = 29;
    else
       line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
```

```
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}

/*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  kblist=chgonkey(NULL); /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
  write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
   case 'y': setMode(3);
          videoinit();
          normal_exit();
          break;
   case 'Y': setMode(3);
          videoinit();
          normal_exit();
          break;
   case 'n':write_horz_str(-300,-210,"                          ",31);
          break;
   case 'N':write_horz_str(-300,-210,"                          ",31);
          break;
   default : break;
   }
```

```
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_6.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

/*************************************************************************
                GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
 *************************************************************************/

char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
/*************************************************************************
                        FUNCTION DEFINITIONS
*************************************************************************/
void wait(char title[]);
void open_matrix_paren(int x1, int y1,int x2,int y2,int color);
static void example_3_6 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
/*************************************************************************
                          MAIN PROGRAM
*************************************************************************/
main()
{
      cls (1);
      adapt = getAdapter();
       if (adapt == 'V'){
          setMode(0x12);
          cls(1);
      }
      if (adapt == 'E'){
         setMode(16);
         cls(1);
       }
        if (adapt == 'C'){
           setMode(4);
           setCGAPalette(0);
           cls(1);
         }
        example_3_6();
        setMode(3);
}
/*************************************************************************/
static void example_3_6(void)
{
      LINEWIDTH = 3;
      cls(1);
```

```
aspect = 1.0;
drawRect(-309,210,310,-200,11);
drawLine(-270,65,-220,65,11);   /**********************/
drawLine(-220,65,-270,25,11);   /*                    */
drawLine(-220,65,-220,20,11);   /*  Figure G1         */
write_horz_char(-245,0,71,31);  /*                    */
write_horz_char(-238,-5,49,31); /**********************/
fillOval(-270,65,3,16,aspect);
fillOval(-220,65,3,16,aspect);
fillOval(-270,25,3,16,aspect);
fillOval(-220,20,3,16,aspect);

write_horz_char(-285,75,86,31);
write_horz_char(-278,70,49,31);   /* V1 */

write_horz_char(-217,75,86,31);
write_horz_char(-210,70,50,31);   /* V2 */

write_horz_char(-280,23,86,31);
write_horz_char(-273,18,51,31);   /* V3 */

write_horz_char(-220,18,86,31);
write_horz_char(-213,13,52,31);   /* V4 */

open_matrix_paren(-130,75,-30,2,11); /* paren for matrix G1 */
write_horz_char(-110,74,48,31);  /* write 11 */
write_horz_char(-90,74,49,31);  /* write 12 */
write_horz_char(-70,74,48,31);  /* write 13 */
write_horz_char(-50,74,48,31);  /* write 14 */
write_horz_char(-110,54,49,31);  /* write 21 */
write_horz_char(-90,54,48,31);  /* write 22 */
write_horz_char(-70,54,49,31);  /* write 23 */
write_horz_char(-50,54,49,31);  /* write 24 */
write_horz_char(-110,34,48,31);  /* write 31 */
write_horz_char(-90,34,49,31);  /* write 32 */
write_horz_char(-70,34,48,31);  /* write 33 */
write_horz_char(-50,34,48,31);  /* write 34 */
write_horz_char(-110,14,48,31);  /* write 41 */
write_horz_char(-90,14,49,31);  /* write 42 */
write_horz_char(-70,14,48,31);  /* write 43 */
write_horz_char(-50,14,48,31);  /* write 44 */
```

```
write_horz_char(-145,42,61,31);  /* = */
write_horz_char(-160,42,41,31);  /* ) */
write_horz_char(-175,42,71,31);  /* G */
write_horz_char(-167,37,49,31);  /* 1 */
write_horz_char(-182,42,40,31);  /* ( */
write_horz_char(-190,42,65,31);  /* A */
write_horz_str(-163,-20,"(a)",11); /* (a) */

drawLine(10,65,70,65,11);
drawLine(70,65,40,20,11);
drawLine(40,20,10,65,11);
fillOval(10,65,3,16,aspect);
fillOval(70,65,3,16,aspect);
fillOval(40,20,3,16,aspect);
write_horz_char(-5,75,86,31);
write_horz_char(2,70,49,31);    /* V1 */
write_horz_char(72,75,86,31);
write_horz_char(79,70,50,31);    /* V2 */
write_horz_char(40,17,86,31);
write_horz_char(47,12,51,31);    /* V3 */
write_horz_char(47,0,71,31);  /*              */
write_horz_char(54,-5,50,31);  /*********************/


open_matrix_paren(170,70,250,17,11);  /* paren for matrix G1 */
write_horz_str(190,68,"0 1 1",31);
write_horz_str(190,48,"1 0 1",31);
write_horz_str(190,28,"1 1 0",31);

write_horz_char(155,48,61,31);  /* = */
write_horz_char(145,48,41,31);  /* ) */
write_horz_char(130,48,71,31);  /* G */
write_horz_char(137,43,50,31);  /* 2 */
write_horz_char(120,48,40,31);  /* ( */
write_horz_char(105,48,65,31);  /* A */
write_horz_str(125,-20,"(b)",11); /* (b) */
write_horz_str(-300,200," Figure below contains two graphs and"
   " their adjacency matrices. For (a) the",31);
write_horz_str(-300,185," 1,2 entry is 1 because there is an"
   " edge between vertices V1 and V2,",31);
write_horz_str(-300,170," and the 3,4 element is 0 because"
   " there is no edge between V3 and V4.",31);
```

```
    write_horz_str(-300,155," For (b) we see that the 1,2 entry and"
       " 1,3 entries are 1 because of the ",31);
    write_horz_str(-300,140," edges between V1 and V2 and"
        " between V1 and V3.",31);

    write_horz_str(-300,-50," Note that in Figure (a) the sum of the entries"
       " in row 1 is 1, which is",31);
    write_horz_str(-300,-65," the degree of V1, and likewise the"
    " sum of the entries in row 2 is the",31);
    write_horz_str(-300,-80," degree of V2. This illustrates a more"
    " general result.",31);

    wait("");

}

/*************************************************************************/

void open_matrix_paren(int x1,int y1,int x2,int y2,int color)
{

    extern unsigned long int PATTERN;
    drawLine (x1 ,y1,x1+5,y1,color);
    drawLine (x1 ,y1,x1,y2,color);
    drawLine (x1 ,y2,x1+5,y2,color);
    drawLine (x2 ,y1,x2-5,y1,color);
    drawLine (x2 ,y1,x2,y2,color);
    drawLine (x2,y2,x2-5,y2,color);
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
    line = 29;
    else
    line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
```

```
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}

/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'Y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'n':write_horz_str(-300,-210,"                          ",31);
           break;
    case 'N':write_horz_str(-300,-210,"                          ",31);
           break;
```

```
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}

/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : ex3_7.c
  AUTHOR   : Yavuz BAS
  DATE     : Feb. 4, 1990
  REVISED  : Mar. 5, 1990

    DESCRIPTION : This program contains an example about the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

/*************************************************************************
            GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
*************************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescm,crow,ccol;
```

```
        /*****************************************************
                              FUNCTION DEFINITIONS
        *****************************************************/

void wait(char title[]);
static void example_3_7 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
        /*****************************************************
                                MAIN PROGRAM
        *****************************************************/
main()
{
    cls (1);
    adapt = getAdapter();
    if (adapt == 'V'){
       setMode(0x12);
       cls(1);
    }
    if (adapt == 'E'){
       setMode(16);
       cls(1);
    }
    if (adapt == 'C'){
       setMode(4);
       setCGAPalette(0);
       cls(1);
    }
     example_3_7();
     setMode(3);
}
/*************************************************************************/

static void example_3_7(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    drawLine(-120,150,0,150,11);
```

```
drawLine(0,150,0,0,11);
drawLine(0,0,-120,0,11);
drawLine(-120,150,-120,75,11);
drawLine(-120,75,0,75,11);
 aspect = 1.0;
 fillOval(-120,150,3,16,aspect);
 fillOval(0,150,3,16,aspect);
 fillOval(0,0,3,16,aspect);
 fillOval(-120,0,3,16,aspect);
 fillOval(-120,75,3,16,aspect);
 fillOval(0,75,3,16,aspect);
write_horz_char(-135,160,86,31);
write_horz_char(-128,155,49,31);      /* V1 */

write_horz_char(5,160,86,31);
write_horz_char(12,155,50,31);        /* V2 */

write_horz_char(-130,75,86,31);
write_horz_char(-123,70,51,31);       /* V3 */

write_horz_char(5,75,86,31);
write_horz_char(12,70,52,31);         /* V4 */

write_horz_char(5,-5,86,31);
write_horz_char(12,-10,53,31);        /* V5 */

write_horz_char(-130,-5,86,31);
write_horz_char(-123,-10,54,31);      /* V6 */

write_horz_str(-60,-15,"(a)",11);     /* (a) */
write_horz_str(50,150,"V :V ,V",11); /****************/
write_horz_char(57,145,49,11);         /*  V1:V2,V3   */
write_horz_char(82,145,50,11);         /*             */
write_horz_char(107,145,51,11);        /****************/

write_horz_str(50,130,"V :V ,V",11); /****************/
write_horz_char(57,125,50,11);         /*  V2:V1,V4       */
write_horz_char(82,125,49,11);         / *                 */
write_horz_char(107,125,52,11);        /****************/

write_horz_str(50,110,"V :V ,V",11); /****************/
write_horz_char(57,105,51,11);         /*  V3:V1,V4       */
```

```
    write_horz_char(82,105,49,11);      /*                    */
    write_horz_char(107,105,52,11);     /****************/

    write_horz_str(50,90,"V :V ,V ,V",11); /****************/
    write_horz_char(57,85,52,11);          /* V4:V2,V3,V5  */
    write_horz_char(82,85,50,11);          /*                   */
    write_horz_char(107,85,51,11);         /****************/
    write_horz_char(132,85,53,11);

    write_horz_str(50,70,"V :V ,V",11);   /****************/
    write_horz_char(57,65,53,11);          /* V5:V4,V6      */
    write_horz_char(82,65,52,11);          /*                   */
    write_horz_char(107,65,54,11);         /****************/

    write_horz_str(50,50,"V :V ",11);     /****************/
    write_horz_char(57,45,54,11);          /* V6:V5          */
    write_horz_char(82,45,53,11);          /****************/
    write_horz_str(80,-15,"(b)",11);      /* (b)               */

    write_horz_str(-300,-50," For the graph in Figure above"
      " there are 6 labeled vertices, and we list",31);
    write_horz_str(-300,-65," them in a vertical column as in"
    " (b). Beside vertex V1 we list the adjacent",31);
    write_horz_str(-300,-80," vertices, which are V2 and V3"
    " Then proceding to the next vertex V2,",31);
    write_horz_str(-300,-95," we list the vertices adjacent to"
    " it, V1 and V4. This process is continued",31);
    write_horz_str(-300,-110," until we get the adjacency list"
    " in (b).",31);

    wait("");

}

/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
    line = 29;
```

```
    else
    line = 24;
    tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
}

/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ():
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
       write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
     case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;
```

```
    case 'n':write_horz_str(-300,-210,"                    ",31);
            break;
    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;
    default : break;
    }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_8.c
  AUTHOR   : Yavuz BAS
  DATE     : Feb. 4, 1990
  REVISED  : Mar. 5, 1990

  DESCRIPTION . This program contains an example about the "Graphs and
their Representations".

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
static int *savescm,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void open_matrix_paren(int x1, int y1,int x2,int y2,int color);
static void example_3_8 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                        MAIN PROGRAM
        *************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
      if (adapt == 'V'){
         setMode(0x12);
         cls(1);
       }
       if (adapt == 'E'){
          setMode(16);
          cls(1);
       }
       if (adapt == 'C'){
           setMode(4);
           setCGAPalette(0);
           cls(1);
       }
    example_3_8();
    setMode(3);

}
```

```
/*****************************************************************************/

static void example_3_8(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    open_matrix_paren(-200,55,-140,2,11);
    open_matrix_paren(-100,55,-40,2,11);
    open_matrix_paren(0,55,100,2,11);
    open_matrix_paren(140,55,200,2,11);

    write_horz_char(-120,34,43,31);    /* + */
    write_horz_char(-20,34,61,31);     /* = */
    write_horz_char(120,34,61,31);     /* = */

    write_horz_str(-180,54,"3 4",31);
    write_horz_str(-180,34,"2 1",31);
    write_horz_str(-180,14,"0 5",31);

    write_horz_str(-80,54,"2 6",31);
    write_horz_str(-80,34,"3 0",31);
    write_horz_str(-80,14,"2 1",31);

    write_horz_str(20,54,"3+2 4+6",31);
    write_horz_str(20,34,"2+3 1+0",31);
    write_horz_str(20,14,"0+2 5+1",31);

    write_horz_str(160,54,"5 10",31);
    write_horz_str(160,34,"5  1",31);
    write_horz_str(160,14,"2  6",31);

    write_horz_str(-300,-50," With the matrices given above,"
       " note how the sum of the two matrices is",31);
    write_horz_str(-300,-65," formed by adding together the"
       " 1,1 entries of the two matrices, then the",31);
    write_horz_str(-300,-80," 1,2 entries, then the 2,1 entries"
       " and continuing until the complete sum",31);
    write_horz_str(-300,-95," is formed.",31);
    wait("");

}
```

```
/*************************************************************************/

void open_matrix_paren(int x1,int y1,int x2,int y2,int color)
{

    extern unsigned long int PATTERN;
    drawLine (x1 ,y1,x1+5,y1,color);
    drawLine (x1 ,y1,x1,y2,color);
    drawLine (x1 ,y2,x1+5,y2,color);
    drawLine (x2  y1,x2-5,y1,color);
    drawLine (x2 ,y1,x2,y2,color);
    drawLine (x2,y2,x2-5,y2,color);
}

/*************************************************************************/

void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
    line = 29;
    else
    line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
      write_horz_str(30,-210,"                ",31);
  }
  switch (ch)        {
   case 'y': setMode(3);
          videoinit();
          normal_exit();
          break;
   case 'Y': setMode(3);
          videoinit();
          normal_exit();
          break;

  case 'n':write_horz_str(-300,-210,"                          ",31);
          break;
  case 'N':write_horz_str(-300,-210,"                          ",31);
          break;
  default : break;
  }
 if(_mouse&MS_CURS) msshowcur();
 chgonkey(kblist);     /* restore any hidden hot keys */


}
```

```
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_9.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
                GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
static int *savescm,crow,ccol;
```

```
/*****************************************************
                    FUNCTION DEFINITIONS
*****************************************************/

void wait(char title[]);
void open_matrix_paren(int x1, int y1,int x2,int y2,int color);
static void example_3_9 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

/*****************************************************
                      MAIN PROGRAM
*****************************************************/

main()
{
    cls (1);
    adapt = getAdapter();
    if (adapt == 'V')
    {
        setMode(0x12);
        cls(1);
    }
    if (adapt == 'E')
    {
        setMode(16);
        cls(1);
    }
    if (adapt == 'C')
    {
       setMode(4);
        setCGAPalette(0);
        cls(1);
    }
    example_3_9();
    setMode(3);

}
```

```
/*************************************************************************/
static void example_3_9(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,190," For the matrices A and B listed below,",31);
    open_matrix_paren(-100,170,-20,137,11);
    open_matrix_paren(20,170,120,117,11);

    write_horz_char(-130,155,65,31);    /* A */
    write_horz_char(-115,155,61,31);    /* = */
    write_horz_char(-10,155,66,31);     /* B */
    write_horz_char(5,155,61,31);       /* = */

    write_horz_str(-80,169,"1 2 3",31);
    write_horz_str(-80,149,"4 5 6",31);

    write_horz_str(40,169,"1 3 5 8",31);
    write_horz_str(40,149,"0 1 6 9",31);
    write_horz_str(40,129,"2 4 7 1",31);

    write_horz_str(-300,110," the 1,3 element in the product  AB is",31);
    write_horz_str(-290,90,"a  b  + a  b  + a  b  = 1x5 + 2x6 + 3x7 = 38." ,31);
    write_horz_char(-283,85,49,31);     /* a1 */
    write_horz_char(-275,85,49,31);     /* a11 */
    write_horz_char(-258,85,49,31);     /* b1 */
    write_horz_char(-250,85,51,31);     /* b13 */
    write_horz_char(-210,85,49,31);     /* a1 */
    write_horz_char(-202,85,50,31);     /* a12 */
    write_horz_char(-185,85,50,31);     /* b2 */
    write_horz_char(-177,85,51,31);     /* b23 */
    write_horz_char(-137,85,49,31);     /* a1 */
    write_horz_char(-129,85,51,31);     /* a13 */
    write_horz_char(-112,85,51,31);     /* b3 */
    write_horz_char(-104,85,51,31);     /* b33 */
    write_horz_str(-300,65," The 2,4 element in AB is",31);

    write_horz_str(-290,45,"a  b  + a  b  + a  b  = 4x8 + 5x9 + 6x1 = 83." ,31);
    write_horz_char(-283,40,49,31);     /* a2 */
    write_horz_char(-275,40,49,31);     /* a21 */
    write_horz_char(-258,40,49,31);     /* b1 */
    write_horz_char(-250,40,51,31);     /* b14 */
```

```
    write_horz_char(-210,40,49,31);     /* a2 */
    write_horz_char(-202,40,50,31);     /* a22 */
    write_horz_char(-185,40,50,31);     /* b2 */
    write_horz_char(-177,40,51,31);     /* b24 */
    write_horz_char(-137,40,49,31);     /* a2 */
    write_horz_char(-129,40,51,31);     /* a23 */
    write_horz_char(-112,40,51,31);     /* b3 */
    write_horz_char(-104,40,51,31);     /* b34 */
    write_horz_str(-300,20," Continuing in this fashion the product AB is found to
     be",31);
    open_matrix_paren(-250,-20,160,-53,11);
    write_horz_str(-230,-21,"1x1+2x0+3x2 1x3+2x1+3x4"
                        " 1x5+2x6+3x7 1x8+2x9+3x1",31);
    write_horz_str(-230,-41,"4x1+5x0+6x2 4x3+5x1+6x4"
                         " 4x5+5x6+6x7 4x8+5x9+6x1",31);
    open_matrix_paren(-200,-71,-80,-104,11);
    write_horz_str(-180,-72," 7 17 38 29",31);
    write_horz_str(-180,-92,"16 41 92 83",31);
    write_horz_char(-215,-87,61,31);      /* =  */
    write_horz_char(-75,-100,46,31);      /* .  */

     wait("");

}

/*************************************************************************/


void open_matrix_paren(int x1,int y1,int x2,int y2,int color)
{

    extern unsigned long int PATTERN;
    drawLine (x1 ,y1,x1+5,y1,color);
    drawLine (x1 ,y1,x1,y2,color);
    drawLine (x1 ,y2,x1+5,y2,color);
    drawLine (x2 ,y1,x2-5,y1,color);
    drawLine (x2 ,y1,x2,y2,color);
    drawLine (x2,y2,x2-5,y2,color);
}
```

```
/*****************************************************************/
void wait(char title[])
{
        int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
    line = 29;
    else
    line = 24;
    tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
}
/*****************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
  write_horz_str(30,-210," Please type y or n",79);
  ch = getch ();
  write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
          videoinit();
          normal_exit();
```

```
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

    case 'n':write_horz_str(-300,-210,"                    ",31);
            break;

    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;

    default : break;
    }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/


static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : q1.c
   AUTHOR     : Yavuz BAS
   DATE       : Feb. 4, 1990
   REVISED    : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*****************************************************
                        FUNCTION DEFINITIONS
        *****************************************************/

void wait(char title[]);
void question_1  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*****************************************************
                          MAIN PROGRAM
        *****************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
           setMode(0x12);
           cls(1);
         }
      if (adapt == 'E')
        {
           setMode(16);
           cls(1);
        }
      if (adapt == 'C')
        {
           setMode(4);
           setCGAPalette(0);
           cls(1);
         }
   question_1();
   setMode(3);
}
```

```
/*************************************************************************/
static void question_1(void)
{
  LINEWIDTH = 3;
  cls(1);
  drawRect(-309,210,310,-200,11);
  write_horz_str(-300,160," Determine if a graph is indicated in"
                          " the following Figure.",31);
  aspect = 1.0;
  fillOval(-110,130,3,16,aspect);
  write_horz_char(-125,130,65,31); /* A */
  write_horz_str(-300,40," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,40," Please type y or n",31);
     Ch = waitkey();
     if(Ch==ESC) confirm_graph_exit();
     if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
     write_horz_str(0,40,"                    ",31); }
 switch (Ch)       {
  case 'y' :write_horz_str(-30,40," y ",31);
        write_horz_str(-30,20," Yes, because this is a graph that",31);
        write_horz_str(-30,0," consists of one vertex but no edges",31);
        break;
  case 'Y' :write_horz_str(-30,40," Y ",31);
        write_horz_str(-30,20," Yes, because this is a graph that",31);
        write_horz_str(-30,0," consists of one vertex but no edges",31);
        break;

  case 'n' :write_horz_str(-30,40," n ",31);
        write_horz_str(-30,20," The answer must be yes, because",31);
        write_horz_str(-30,0," this is a graph that consists of",31);
        write_horz_str(-30,-20," one vertex but no edges.",31);
        break;
  case 'N' :write_horz_str(-30,40," N ",31);
```

```
        write_horz_str(-30,20," The answer must be yes, because",31);
        write_horz_str(-30,0," this is a graph that consists of",31);
        write_horz_str(-30,-20," one vertex but no edges.",31);
        break;
  defau!t : break;  }
  wait("Exercise 1");
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
      line = 29;
    else
      line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
    write_horz_str(30,-210," Please type y or n",79);
```

```
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)         {
     case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
     case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;
     case 'n':write_horz_str(-300,-210,"                          ",31);
            break;
     case 'N':write_horz_str(-300,-210,"                          ",31);
            break;
     default : break;
     }
    if(_mouse&MS_CURS) msshowcur();
    chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
    srestore(savescn);
    gotoxy_(crow,ccol);
    if( _mouse) mshidecur();
    showcur();
    exit(0);
}
```

```
/* PROGRAM  : q2.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Graphs and
their Representations"

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************. ************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
         *************************************************/

void wait(char title[]);
void question_2  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                          MAIN PROGRAM
         *************************************************/
main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
            setMode(0x12);
            cls(1);
         }
     if (adapt == 'E')
        {
            setMode(16);
            cls(1);
        }
    if (adapt == 'C')
        {
            setMode(4);
            setCGAPalette(0);
            cls(1);
        }
   question_2();
   setMode(3);
}
```

```
/*************************************************************************/
static void question_2(void)
{
  LINEWIDTH = 3;
  cls(1);
  drawRect(-309,210,310,-200,11);
  write_horz_str(-300,160," Determine if a graph is indicated in"
                          " the following Figure.",31);
  drawLine(-110,130,0,130,11);
  write_horz_str(-300,40," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,40," Please type y or n",31);
     Ch = waitkey();
     if(Ch==ESC) confirm_graph_exit();
     if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
     write_horz_str(0,40,"                    ",31);
  }
 switch (Ch)       {
  case 'y' :write_horz_str(-30,40," y ",31);
        write_horz_str(-30,20," No, because there is an edge",31);
        write_horz_str(-30,0," that doesn't have any vertices.",31);
        break;
  case 'Y' :write_horz_str(-30,40," Y ",31);
        write_horz_str(-30,20," No, because there is an edge",31);
        write_horz_str(-30,0," that doesn't have any vertices",31);

        break;
  case 'n' :write_horz_str(-30,40," n ",31);
        write_horz_str(-30,20," You are right, because there is an",31);
        write_horz_str(-30,0," edge that doesn't have any vertices.",31);
        break;
  case 'N' :write_horz_str(-30,40," N ",31);
        write_horz_str(-30,20," You are right, because there is an",31);
        write_horz_str(-30,0," edge that doesn't have any vertices.",31);
```

```
            break;
    default : break;    }
    wait("Exercise 2");
}
/*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
      line = 29;
      else
      line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
      cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
```

```
    switch (ch)         {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'Y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'n':write_horz_str(-300,-210,"                          ",31);
               break;
     case 'N':write_horz_str(-300,-210,"                          ",31);
               break;
     default : break;
     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : q3.c
   AUTHOR       : Yavuz BAS
   DATE         : Feb. 4, 1990
   REVISED      : Mar. 5, 1990

   DESCRIPTION  : This program contains an exercise for the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescm,crow,ccol;
```

```
        /*****************************************************
                        FUNCTION DEFINITIONS
         *****************************************************/

void wait(char title[]);
void question_1  (void);
void question_2  (void);
void question_3  (void);
static void normal_exit(void);
static void confirm_graph_exit(void);
        /*****************************************************
                        MAIN PROGRAM
         *****************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
       {
          setMode(0x12);
          cls(1);
       }
     if (adapt == 'E')
       {
          setMode(16);
          cls(1);
       }
    if (adapt == 'C')
       {
          setMode(4);
          setCGAPalette(0);
          cls(1);
       }
  question_3();
  setMode(3);
}
```

```
/*************************************************************************/
static void question_3(void)
{
  LINEWIDTH = 3;
  cls(1);
  drawRect(-309,210,310,-200,11);
  write_horz_str(-300,160," Determine if a graph is indicated in"
                          " the following Figure.",31);
  drawLine(-110,130,0,100,11);  /* AC */
  drawLine(0,100,100,100,11);   /* C? */
  drawLine(-110,70,0,100,11);   /* BC */

  aspect = 1.0;
  fillOval(-110,130,3,16,aspect);
  fillOval(0,100,3,16,aspect);
  fillOval(-110,70,3,16,aspect);
  write_horz_char(-125,130,65,31); /* A */
  write_horz_char(-125,70,66,31);  /* B */
  write_horz_char(0,120,67,31);    /* C */
  write_horz_str(-300,40," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,40," Please type y or n",31);
     Ch = waitkey();
     if(Ch==ESC) confirm_graph_exit();
     if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
     write_horz_str(0,40,"                    ",31);
  }
 switch (Ch)       {
 case 'y' :write_horz_str(-30,40," y ",31);
        write_horz_str(-30,20," No, because there is a missing",31);
        write_horz_str(-30,0," vertex for the edge starting",31);
        write_horz_str(-30,-20," at C and going to the right.",31);
        break;
 case 'Y' :write_horz_str(-30,40," Y ",31);
```

```
        write_horz_str(-30,20," No, because there is a missing",31);
        write_horz_str(-30,0," vertex for the edge starting",31);
        write_horz_str(-30,-20," at C and going to the right.",31);
        break;
  case 'n' :write_horz_str(-30,40," n ",31);
        write_horz_str(-30,20," You are right, because there is a",31);
        write_horz_str(-30,0," missing vertex for the edge starting",31);
        write_horz_str(-30,-20," at C and going to the right.",31);
        break;
  case 'N' :write_horz_str(-30,40," N ",31);
        write_horz_str(-30,20," You are right, because there is a",31);
        write_horz_str(-30,0," missing vertex for the edge starting",31);
        write_horz_str(-30,-20," at C and going to the right.",31);
        break;
  default : break;  }
  wait("Exercise 3");
}
/*****************************************************************************/

void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
      line = 29;
    else
      line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
```

```
/*************************************************************************** */
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'Y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'n':write_horz_str(-300,-210,"                          ",31);
           break;
    case 'N':write_horz_str(-300,-210,"                          ",31);
           break;
    default : break;
   }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : q4.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescm,crow,ccol;
```

```
        /*************************************************
                FUNCTION DEFINITIONS
         *************************************************/

void wait(char title[]);
void question_4  (void);
static void normal_exit(void);
static void confirm_graph_exit(void);

        /*************************************************
                        MAIN PROGRAM
         *************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
           setMode(0x12);
           cls(1);
        }
       if (adapt == 'E')
          {
           setMode(16);
           cls(1);
          }
       if (adapt == 'C')
         {
           setMode(4);
           setCGAPalette(0);
           cls(1);
         }
   question_4();
   setMode(3);
}
```

```
/*****************************************************************/
static void question_4(void)
{
   LINEWIDTH = 3;
  cls(1);
  drawRect(-309,210,310,-200,11);
  write_horz_str(-300,160," For the following Figure give the"
                          " degree of A.",31);
  drawRect(-50,120,50,20,11);
  drawLine(-50,120,50,20,11);   /* AD */
  drawLine(50,120,-50,20,11);   /* BC */
  drawLine(-50,120,-70,70,11);  /* AE */

  aspect = 1.0;
  fillOval(-50,120,3,16,aspect);
  fillOval(50,20,3,16,aspect);
  fillOval(50,120,3,16,aspect);
  fillOval(-50,20,3,16,aspect);
  fillOval(-70,70,3,16,aspect);

  write_horz_char(-65,120,65,31);  /* A */
  write_horz_char(60,120,66,31);   /* B */
  write_horz_char(-65,20,67,31);   /* C */
  write_horz_char(60,20,68,31);    /* D */
  write_horz_char(-85,70,69,31);   /* E */
  write_horz_str(-300,0," Choose one of the followings:",31);
  write_horz_str(-300,-20," a) deg (A) = 1",31);
  write_horz_str(-300,-40," b) deg (A) = 2",31);
  write_horz_str(-300,-60," c) deg (A) = 3",31);
  write_horz_str(-300,-80," d) deg (A) = 4",31);
  write_horz_str(-300,-100," e) deg (A) = 5",31);
  write_horz_str(-300,-120," Type the letter of which you believe is true--->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd') || (Ch == 'e'))) {
     write_horz_str(0,-140," Please type one of a, b, c, d, or e",31);
```

```
    Ch = waitkey();
    if(Ch==ESC) confirm_graph_exit();
    if ((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd') || (Ch == 'e'))
    write_horz_str(0,-140,"                                    ",31);}
switch (Ch)        {
 case 'a' :write_horz_str(90,-120," a ",31);
        write_horz_str(-300,-140," No, the answer must be d, because"
                                  " there are four edges that start",31);
        write_horz_str(-300,-160," from the vertex A. Namely, these"
                                  " edges are AB, AC, AD, and AE",31);
        break;
 case 'b' :write_horz_str(90,-120," b ",31);
        write_horz_str(-300,-140," No, the answer must be d, because"
                                  " there are four edges that start",31);
        write_horz_str(-300,-160," from the vertex A. Namely, these"
                                  " edges are AB, AC, AD, and AE",31);
        break;
 case 'c' :write_horz_str(90,-120," c ",31);
        write_horz_str(-300,-140," No, the answer must be d, because"
                                  " there are four edges that start",31);
        write_horz_str(-300,-160," from the vertex A. Namely, these"
                                  " edges are AB, AC, AD, and AE",31);
        break;
 case 'e' :write_horz_str(90,-120," e ",31);
        write_horz_str(-300,-140," No, the answer must be d, because"
                                  " there are four edges that start",31);
        write_horz_str(-300,-160," from the vertex A. Namely, these"
                                  " edges are AB, AC, AD, and AE",31);
        break;
 case 'd' :write_horz_str(90,-120," d ",31);
        write_horz_str(-300,-140," Yes, you are right. Deg (A) =4,"
                                  " because there are four edges that start",31);
        write_horz_str(-300,-160," from the vertex A. Namely, these"
                                  " edges are AB, AC, AD, and AE",31);
        break;
 default : break;   }
```

```
   wait("Exercise 4");
}
/*****************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
       if ((mode == 0x11) || (mode == 0x12))
          line = 29;
       else
          line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
      cls(0);
}

/*****************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
```

```
    switch (ch)         {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'Y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'n':write_horz_str(-300,-210,"                    ",31);
               break;
     case 'N':write_horz_str(-300,-210,"                    ",31);
               break;
     default : break;
     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : q5.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescm,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void question_5  (void);
static void normal_exit(void);
static void confirm_graph_exit(void);

        /*************************************************
                        MAIN PROGRAM
        *************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
         {
           setMode(0x12);
           cls(1);
         }
      if (adapt == 'E')
         {
           setMode(16);
           cls(1);
         }
     if (adapt == 'C')
       {
           setMode(4);
           setCGAPalette(0);
           cls(1);
       }
   question_5();
   setMode(3);
}
```

```
/*************************************************************************/
static void question_5(void)
{
  LINEWIDTH = 3;
  cls(1);
  drawRect(-309,210,310,-200,11);
  write_horz_str(-300,160," For the following Figure give the"
                          " degree of A and C.",31);
  drawLine(-50,130,50,130,11);   /* CB */
  aspect = 1.0;
  fillOval(-50,130,3,16,aspect);
  fillOval(0,80,3,16,aspect);
  fillOval(50,130,3,16,aspect);
  write_horz_char(-65,130,67,31); /* C */
  write_horz_char(60,130,66,31);  /* B */
  write_horz_char(-15,80,65,31);   /* A */
  write_horz_str(-300,50," Choose one of the followings:",31);
  write_horz_str(-300,30," a) deg (A) = 1, deg (C) = 2",31);
  write_horz_str(-300,10," b) deg (A) = 0, deg (C) = 1",31);
  write_horz_str(-300,-10," c) deg (A) = 1, deg (C) = 3",31);
  write_horz_str(-300,-30," d) deg (A) = 0, deg (C) = 3",31);
  write_horz_str(-300,-60," Type the letter of which you believe"
                          " is true--->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd') )) {
     write_horz_str(0,-80," Please type one of a, b, c, or d",31);
     Ch = getch ();
     if ((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'N'))
     write_horz_str(0,-80,"                              ",31);
  }
 switch (Ch)        {
  case 'a' :write_horz_str(90,-60," a ",31);
        write_horz_str(-300,-120," No, the answer must be b, because"
                                 " there is only one edge that starts",31);
        write_horz_str(-300,-140," from the vertex C, wheras there"
```

```
                    " is none from the vertex A.",31);
            break;
    casc 'b' :write_horz_str(90,-60," b ",31);
            write_horz_str(-300,-120," Yes you are right, because"
                                     " there is only one edge that starts",31);
            write_horz_str(-300,-140," from the vertex C, wheras there"
                                     " is none from the vertex A.",31);
            break;
    case 'c' :write_horz_str(90,-60," c ",31);
            write_horz_str(-300,-120," No, the answer must be b, because"
                                     " there is only one edge that starts",31);
            write_horz_str(-300,-140," from the vertex C, wheras there"
                                     " is none from the vertex A.",31);
            break;
    case 'd' :write_horz_str(90,-60," d ",31);
            write_horz_str(-300,-120," No, the answer must be b, because"
                                     " there is only one edge that starts",31);
            write_horz_str(-300,-140," from the vertex C, wheras there"
                                     " is none from the vertex A.",31);
            break;
    default : break;   }
    wait("Exercise 4");
}
/*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
       line = 29;
     else
       line = 24;
      tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
```

```
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
}
/*****************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       write_horz_str(30,-210," Please type y or n",79);
       ch = getch ();
       write_horz_str(30,-210,"                ",31);
    }
    switch (ch)         {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'Y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'n':write_horz_str(-300,-210,"                          ",31);
               break;
     case 'N':write_horz_str(-300,-210,"                          ",31);
               break;
     default : break;   }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);      /* restore any hidden hot keys */
}
```

```
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : q6.c
   AUTHOR     : Yavuz BAS
   DATE       : Feb. 4, 1990
   REVISED    : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescm,crow,ccol;
```

```
        /*************************************************
                    FUNCTION DEFINITIONS
         *************************************************/

void wait(char title[]);
void question_6  (void);
static void normal_exit(void);
static void confirm_graph_exit(void);

        /*************************************************
                      MAIN PROGRAM
         *************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
           setMode(0x12);
           cls(1);
        }
      if (adapt == 'E')
        {
           setMode(16);
           cls(1);
         }
      if (adapt == 'C')
        {
           setMode(4);
           setCGAPalette(0);
           cls(1);
        }
   question_6();
   setMode(3);
}
```

```
/*************************************************************************/
static void question_6(void)
{
   LINEWIDTH = 3;
   cls(1);
   drawRect(-309,210,310,-200,11);
   write_horz_str(-300,160," Are the graphs in the following Figure"
                           " isomorphic or not?",31);
   LINEWIDTH = 1;
   aspect = 1.0;
   drawArc(-50,100,30,0,2700,11,aspect);
   LINEWIDTH = 3;
   drawLine(-50,100,-20,100,11);
   drawLine(-50,100,-50,70,11);

   fillOval(-50,100,3,16,aspect);
   fillOval(-20,100,3,16,aspect);
   fillOval(-50,70,3,16,aspect);

   write_horz_char(-65,100,65,31);   /* A */
   write_horz_char(-50,60,66,31);    /* B */
   write_horz_char(-10,100,67,31);   /* C */

   drawLine(50,130,50,70,11);
   drawLine(50,70,110,70,11);
   drawLine(110,70,50,130,11);

   write_horz_char(35,130,80,31);    /* P */
   write_horz_char(35,70,81,31);     /* Q */
   write_horz_char(115,70,82,31);    /* R */

   fillOval(50,130,3,16,aspect);
   fillOval(50,70,3,16,aspect);
   fillOval(110,70,3,16,aspect);

   write_horz_str(-300,30," Enter your choice as y or n --->",31);
```

```
 Ch = waitkey();
 if(Ch==ESC) confirm_graph_exit();
 while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
    write_horz_str(0,30," Please type y or n",31);
    Ch = getch ();
    if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
    write_horz_str(0,30,"                    ",31);
 }
switch (Ch)        {
 case 'y' :write_horz_str(-30,30," y ",31);
        write_horz_str(-30,10," Yes, you are right, because we",31);
        write_horz_str(-30,-10," can find a bijection f from",31);
        write_horz_str(-30,-30," the set {A,B,C} to the set",31);
        write_horz_str(-30,-50," {Q,P,R}. The following table",31);
        write_horz_str(-30,-70," shows the correspondence between",31);
        write_horz_str(-30,-90," edges.",31);
        write_horz_str(-250,-55,"{U,V}      {f(U),f(V)}",31);
        write_horz_str(-250,-85,"{A,B}      {P,Q}",31);
        write_horz_str(-250,-105,"{B,C}      {Q,R}",31);
        write_horz_str(-250,-125,"{A,C}      {P,R}",31);
        drawLine(-260,-40,-40,-40,11);
        drawLine(-260,-70,-40,-70,11);
        drawLine(-260,-140,-40,-140,11);
        drawLine(-260,-40,-260,-140,11);
        drawLine(-40,-40,-40,-140,11);
        break;
 case 'Y' :write_horz_str(-30,30," y ",31);
        write_horz_str(-30,10," Yes, you are right, because we",31);
        write_horz_str(-30,-10," can find a bijection f from",31);
        write_horz_str(-30,-30," the set {A,B,C} to the set",31);
        write_horz_str(-30,-50," {Q,P,R}. The following table",31);
        write_horz_str(-30,-70," shows the correspondence between",31);
        write_horz_str(-30,-90," edges.",31);
        write_horz_str(-250,-55,"{U,V}      {f(U),f(V)}",31);
        write_horz_str(-250,-85,"{A,B}      {P,Q}",31);
        write_horz_str(-250,-105,"{B,C}      {Q,R}",31);
```

```
        write_horz_str(-250,-125,"{A,C}       {P,R}",31);
        drawLine(-260,-40,-40,-40,11);
        drawLine(-260,-70,-40,-70,11);
        drawLine(-260,-140,-40,-140,11);
        drawLine(-260,-40,-260,-140,11);
        drawLine(-40,-40,-40,-140,11);
        break;
case 'n' :write_horz_str(-30,30," n ",31);
        write_horz_str(-30,10," No, the answer must be yes, because",31);
        write_horz_str(-30,-10," we can find a bijection f from",31);
        write_horz_str(-30,-30," the set {A,B,C} to the set",31);
        write_horz_str(-30,-50," {Q,P,R}. The following table",31);
        write_horz_str(-30,-70," shows the correspondence between",31);
        write_horz_str(-30,-90," edges.",31);
        write_horz_str(-250,-55,"{U,V}       {f(U),f(V)}",31);
        write_horz_str(-250,-85,"{A,B}       {P,Q}",31);
        write_horz_str(-250,-105,"{B,C}       {Q,R}",31);
        write_horz_str(-250,-125,"{A,C}       {P,R}",31);
        drawLine(-260,-40,-40,-40,11);
        drawLine(-260,-70,-40,-70,11);
        drawLine(-260,-140,-40,-140,11);
        drawLine(-260,-40,-260,-140,11);
        drawLine(-40,-40,-40,-140,11);
        break;
case 'N' :write_horz_str(-30,30," N ",31);
        write_horz_str(-30,10," No, the answer must be yes, because",31);
        write_horz_str(-30,-10," we can find a bijection f from",31);
        write_horz_str(-30,-30," the set {A    C} to the set",31);
        write_horz_str(-30,-50," {Q,P,R}. The following table",31);
        write_horz_str(-30,-70," shows the correspondence between",31);
        write_horz_str(-30,-90," edges.",31);
        write_horz_str(-250,-55,"{U,V}       {f(U),f(V)}",31);
        write_horz_str(-250,-85,"{A,B}       {P,Q}",31);
        write_horz_str(-250,-105,"{B,C}       {Q,R}",31);
        write_horz_str(-250,-125,"{A,C}       {P,R}",31);
        drawLine(-260,-40,-40,-40,11);
```

```
        drawLine(-260,-70,-40,-70,11);
        drawLine(-260,-140,-40,-140,11);
        drawLine(-260,-40,-260,-140,11);
        drawLine(-40,-40,-40,-140,11);
        break;
  default : break;   }
   wait("Exercise 6");
}
/*****************************************************************************/
void wait(ch, * title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
        line = 29;
      else
        line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
      cls(0);
}
/*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,' Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
```

```
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
     case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;
     case 'n':write_horz_str(-300,-210,"                          ",31);
            break;
     case 'N':write_horz_str(-300,-210,"                          ",31);
            break;
     default : break;
     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : q7.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Graphs and
their Representations".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
         *************************************************/

void wait(char title[]);
void question_7  (void);
static void normal_exit(void);
static void confirm_graph_exit(void);

        /*************************************************
                        MAIN PROGRAM
         *************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
           setMode(0x12);
          cls(1);
        }
      if (adapt == 'E')
        {
           setMode(16);
           cls(1);
         }
     if (adapt == 'C')
       {
           setMode(4);
           setCGAPalette(0);
          cls(1);
       }
   question_7();
   setMode(3);
}
```

```
/*************************************************************************/
static void question_7(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,160," Are the graphs in the following Figure"
                            " isomorphic or not?",31);
   LINEWIDTH = 1;
   aspect = 1.0;
   drawOval(-50,100,30,11,aspect);
   LINEWIDTH = 3;
   drawLine(-50,100,-20,100,11);
   drawLine(-50,100,-50,70,11);
   drawLine(-50,100,-78,105,11);
   fillOval(-50,100,3,16,aspect);
   fillOval(-20,100,3,16,aspect);
   fillOval(-50,70,3,16,aspect);
   fillOval(-78,105,3,16,aspect);

   write_horz_char(-65,100,65,31);   /* A */
   write_horz_char(-50,60,66,31);    /* B */
   write_horz_char(-10,100,67,31);   /* C */
   write_horz_char(-93,105,68,31);   /* D */
   drawRect(50,130,100,70,11);
   drawLine(50,130,100,70,11);
   drawLine(50,70,100,130,11);
   write_horz_char(35,130,80,31);    /* P */
   write_horz_char(35,70,81,31);     /* Q */
   write_horz_char(115,70,82,31);    /* R */
   write_horz_char(115,130,83,31);   /* S */
   fillOval(50,130,3,16,aspect);
   fillOval(50,70,3,16,aspect);
   fillOval(100,70,3,16,aspect);
   fillOval(100,130,3,16,aspect);
   fillOval(75,100,3,16,aspect);
```

```
  write_horz_str(-300,30," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,30," Please type y or n",31);
     Ch = getch ();
     if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
     write_horz_str(0,30,"                    ",31); }
switch (Ch)        {
  case 'y' :write_horz_str(-30,30," y ",31);
         write_horz_str(-30,10," No, they are not isomorphic,",31);
         write_horz_str(-30,-10," because one of the graphs",31);
         write_horz_str(-30,-30," has five vertices while the",31);
         write_horz_str(-30,-50," other one has four.",31);
         break;
  case 'Y' :write_horz_str(-30,30," Y ",31);
         write_horz_str(-30,10," No, they are not isomorphic,",31);
         write_horz_str(-30,-10," because one of the graphs",31);
         write_horz_str(-30,-30," has five vertices while the",31);
         write_horz_str(-30,-50," other one has four.",31);
         break;
  case 'n' :write_horz_str(-30,30," n ",31);
         write_horz_str(-30,10," No, they are not isomorphic,",31);
         write_horz_str(-30,-10," because one of the graphs",31);
         write_horz_str(-30,-30," has five vertices while the",31);
         write_horz_str(-30,-50," other one has four.",31);
         break;
  case 'N' :write_horz_str(-30,30," N ",31);
         write_horz_str(-30,10," No, they are not isomorphic,",31);
         write_horz_str(-30,-10," because one of the graphs",31);
         write_horz_str(-30,-30," has five vertices while the",31);
         write_horz_str(-30,-50," other one has four.",31);
         break;
  default : break;   }
  wait("Exercise 7");
}
```

```
/*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
         line = 29;
      else
         line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
      cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': setMode(3);
           videoinit();
           normal_exit();
```

```
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;
    case 'n':write_horz_str(-300,-210,"                    ",31);
            break;
    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescm);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : paths.c
  AUTHOR     : Yavuz BAS
  DATE       : Feb. 4, 1990
  REVISED    : Mar. 5, 1990

  DESCRIPTION : This program contains the tutorial for the paths and circuits.
  MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/
/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                          /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                            /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk           find_t
  #define ff_name         name
#elif defined(__ZTC__)                           /* Zortech C/C++ */
  #define ffblk           FIND
  #define ff_name         name
  #define ff_attrib       attribute
#endif
#define _GRAPH_T_DEFINED
```

```
/* function prototypes */
static void add_shadow    (void);
static void confirm_quit  (void);
static void disp_sure_msg (void);
static void do_nothing    (void);
static void error_exit    (int errnum);
static void move_window   (int nsrow, int scol);
static void normal_exit   (void);
static void press_a_key   (int wrow);
static void quit_window   (void);
static void restore_cursor(void);
static void short_delay   (void);
static void size_window   (int nerow,int necol);
static void example_3_10  (void);
static void example_3_11  (void);
static void example_3_12  (void);
static void example_3_13  (void);
static void example_3_14  (void);
static void example_3_15  (void);
static void example_3_16  (void);
static void example_3_17  (void);
static void example_3_18  (void);
static void example_3_19  (void);
static void example_3_30  (void);
static void example_3_31  (void);
static void example_3_32  (void);
static void example_3_39  (void);
static void example_3_40  (void);
static void example_3_41  (void);
static void example_3_42  (void);
static void Pex_3_10  (void);
static void Pex_3_11  (void);
static void Pex_3_12  (void);
static void Pex_3_13  (void);
static void Pex_3_14  (void);
static void Pex_3_15  (void);
```

```
static void Pex_3_16  (void);
static void Pex_3_17  (void);
static void Pex_3_18  (void);
static void Pex_3_19  (void);
static void Pex_3_30  (void);
static void Pex_3_31  (void);
static void Pex_3_32  (void);
static void Pex_3_39  (void);
static void Pex_3_42  (void);
static void Pex_3_40  (void);
static void Pex_3_41  (void);

static void definition_3_6(void);
static void definition_3_7(void);
static void definition_3_8(void);
static void definition_3_9(void);
static void definition_3_10(void);
static void definition_3_11(void);
static void definition_3_12(void);
static void definition_3_13(void);
static void definition_3_14(void);
static void definition_3_15(void);
static void definition_3_16(void);
static void Pdef_3_6(void);
static void Pdef_3_7(void);
static void Pdef_3_8(void);
static void Pdef_3_9(void);
static void Pdef_3_10(void);
static void Pdef_3_11(void);
static void Pdef_3_12(void);
static void Pdef_3_15(void);
static void Pdef_3_16(void);
static void Pthm_3_3(void);
static void Pthm_3_4(void);
static void Pthm_3_5(void);
static void paths_circuits (void);
```

```
static void theorem_3_3  (void);
static void theorem_3_4  (void);
static void theorem_3_5  (void);
static void euler_circuits_and_paths(void);
static void euler_circuit_algorithm(void);
static void BFS_algorithm(void);
static void dijkstra_algorithm(void);
static void introduction  (void);
static void hamilton  (void);
static void isomorphism(void);
static void exercises(void);
static void Peuler_circuits_and_paths(void);
static void Peuler_circuit_algorithm(void);
static void PBFS_algorithm(void);
static void Pdijkstra_algorithm(void);
static void Pintroduction  (void);
static void Phamilton  (void);
static void Pisomorphism(void);
static void Pexercise1(void);
static void Pexercise2(void);
static void Pexercise3(void);
static void Pexercise4(void);
static void Pexercise5(void);
static void Pexercise6(void);
static void Pexercise7(void);
static void Pexercise8(void);
static void Pexercise9(void);
static void Pexercise10(void);
static void exercise1(void);
static void exercise2(void);
static void exercise3(void);
static void exercise4(void);
static void exercise5(void);
static void exercise6(void);
static void exercise7(void);
static void exercise8(void);
```

```
static void exercise9(void);
static void exercise10(void);
static void following10(void);
static void following11(void);
static void following12(void);
static void following13(void);
static void following14(void);
static void following15(void);
static void following16(void);
static void following17(void);
static void following18(void);
static void following19(void);
static void following39(void);
static void following40(void);
static void following41(void);
static void following42(void);
static void following30(void);
static void following31(void);
static void following32(void);
static void followingthm_3_3(void);
static void ex_3_18cont(void);
static void Pex_3_18cont(void);
/*************************************************************************/
/* miscellaneous global variables */
static int *savescrn,crow,ccol;
static WINDOW w[10];
static char ssan[10];
/*************************************************************************/
/* error message table */
static char *error_text[]= {
   NULL,  /* ermum =  0, no error   */
   NULL,  /* ermum == 1, windowing error */
   "Syntax:  CXLDEMO [-switches]\n\n"
      "\t -c = CGA snow elimination\n"
      "\t -b = BIOS screen writing\n"
      "\t -m = force monochrome text attributes",
```

```
    "Memory allocation error"
};
/* miscellaneous defines */
#define SHORT_DELAY 18
#define H_WINTITLE  33
/*************************************************************************/
/* this function will add a shadow to the active window                  */
/*************************************************************************/
static void add_shadow(void)
{
   wshadow(LGREY|_BLACK);
}
/*************************************************************************/
/* this function pops open a window and confirms that the user really    */
/* wants to quit the demo.  If so, it terminates the demo program.       */
/*************************************************************************/
static void confirm_quit(void)
{
   struct _onkey_t *kblist;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
   add_shadow();
   wputs("\n Quit demo, are you sure? \033A\156Y\b");
   clearkeys();
   showcur();
   if(wgetchf("YN",'Y')=='Y') normal_exit();
   wclose();
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);      /* restore any hidden hot keys */
}
```

```
/*************************************************************************/
/* this function is called by the pull-down demo for a prompt            */
/*************************************************************************/
static void disp_sure_msg(void)
{
   wprints(0,2,WHITE|_BLUE,"Are you sure?");
}
/*************************************************************************/
/* this function is used as a dummy function for several menu items in the */
/* pull-down demo                                                         */
/*************************************************************************/
static void do_nothing(void)
{
}
/*************************************************************************/
/* this function handles abnormal termination. If it is passed an error code of 1, */
/* then it is a windowing system error. Otherwise the error message is looked up */
/* in the error message table.                                            */
/*************************************************************************/
static void error_exit(int errnum)
{
   if(errnum) { printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
                exit(errnum);   }
}
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/*************************************************************************/
/* this function displays a pause message then pauses for a keypress     */
/*************************************************************************/
static void press_a_key(int wrow)
{
   register int attr1;
   register int attr2;
   attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
   attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
   wcenters(wrow,attr1,"Press a key");
   wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
   hidecur();
   if(waitkey()==ESC) confirm_quit();
   wcenters(wrow,attr1,"          ");
   wprints(wrow,0,attr2,"         ");
}
/*************************************************************************/
static void short_delay(void)
{
   delay_(SHORT_DELAY);
}
/*************************************************************************/
/* this function is called by the pull-down menu demo anytime            */
/* the  selection bar moves on or off the [Q]uit menu items.             */
/*************************************************************************/
static void quit_window(void)
{
   static WINDOW handle=0;
   if(handle) {
      wactiv(handle);
      wclose();
      handle=0;   }
   else {
      handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
      wputs(" Quit takes you back to the\n demo program's main menu.");   }
}
```

```
/*************************************************************************/
static void restore_cursor(void)
{
   wtextattr(WHITE|_MAGENTA);
   showcur();
}
/*************************************************************************/
static void size_window(int nerow,int necol)
{
   wsize(nerow,necol);
   short_delay();
}
/*************************************************************************/
static void move_window(int nsrow,int nscol)
{
   if(wmove(nsrow,nscol)) error_exit(1);
   short_delay();
}
/*************************************************************************/
void main()
{
  paths_circuits();
}
/*************************************************************************/
static void paths_circuits (void)
{
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   cclrscrn(LGREY|_GREEN);
   introduction();
   srestore(scrn);
}
```

```
/*************************************************************************/
static void introduction (void)
{
  /*************************************************************/
  /* attach [Pageup] to the introduction function */
  setonkey(0x4900,Pintroduction,0);
  /*************************************************************/
  /* attach [Pagedown] to the definition_3_6 function */
  setonkey(0x5100,Pdef_3_6,0);
  /*************************************************************/
  if((w[1]=wopen(3,3,17,74,3,LCYAN|_BLUE,WHITE|_BLUE))==0) error_exit(1);
  wtitle("[Paths and Circuits]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" As we have seen, graphs can be used to describe a variety"
       " of situations. In many cases we want to know whether it is"
       " possible to go from one vertex to another by following"
       " a route using the edges. In other cases, it may be necessary"
       " to perform a test that involves finding a route through all"
       " the vertices or over all the edges. While many situations"
       " can be described by graphs as we have defined them, there"
       " are others where it may be necessary to allow an edge from"
       " a vertex to itself or to allow more than one edge between"
       " vertices. For example, when a road system is being described"
       ", there can be two roads, an interstate highway and an older"
       " two-lane road, between the same two towns. There could even"
       " be a scenic route starting and ending at the same town.");
  press_a_key(12);
  wclose();
  definition_3_6();
}
```

```
/*************************************************************************/
 void definition_3_6(void)
 {
  /*********************************************************************/
  /* attach [Pageup] to the introduction function */
  setonkey(0x4900,Pintroduction,0);
  /*********************************************************************/
  /* attach [Pagedown] to the example_3_10 function */
  setonkey(0x5100,Pex_3_10,0);
  /*********************************************************************/

  if((w[1]=wopen(6,10,18,70,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
     error_exit(1);
  wtitle("[Graph Representations - Definition_3_1]",
          TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" A multigraph is a generalization of a graph that consists of"
       " a finite set of vertices and a finite set of edges. Unlike a"
       " graph, a multigraph can contain edges from a vertex to itself"
       " and several edges between the same two vertices. An edge from"
       " a vertex to itself is called a loop. When there is more than"
       " one edge between two vertices, these edges are called parallel"
       " edges. It is important to note that a graph is a special kind"
       " of multigraph. Thus, all the definitions given for multigraphs"
       " apply to graphs as well.");
  press_a_key(10);
  wclose();
  following10();

 }
```

```
/*************************************************************************/
void following10(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the definition_3_6 function */
  setonkey(0x4900,Pdef_3_6,0);
  /*********************************************************************/
  /* attach [Pagedown] to the definition_3_7 function */
  setonkey(0x5100,Pdef_3_7,0);
  /*********************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_10();
}
/*************************************************************************/
 void example_3_10(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_10.exe",NULL);
  srestore(scrn);
  definition_3_7();
 }
```

```
/*************************************************************************/
void definition_3_7(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the example_3_10 function */
  setonkey(0x4900,Pex_3_10,0);
  /*********************************************************************/
  /* attach [Pagedown] to the example_3_11 function */
  setonkey(0x5100,Pex_3_11,0);
  /*********************************************************************/
  if((w[1]=wopen(4,3,12,74,3,LCYAN|_BLUE,WHITE|_BLUE))==0) error_exit(1);
  wtitle("[A path in a graph]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" A path of length k in a graph is a sequence of vertices"
       " V0, V1, V2,...,Vk such that for i = 1, 2,...,k,{Vi-1, Vi}"
       " is an element of E. The length of this path is n, the number"
       " of edges listed. We note that U is a path to itself of length"
       " 0.");
  press_a_key(5);
  if((w[2]=wopen(14,3,20,74,3,LCYAN|_CYAN,WHITE|_CYAN))==0)
    error_exit(1);
  wtitle("[Paths]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" In a path the vertices need not be distinct from eachother"
       " and likewise some of the edges can be the same. When there"
       " can be no chance of confusion a path can be represented by the"
       " vertices V1, V2,..., Vn+1 only or by the edges e1, e2,..."
       ", en only.");
  press_a_key(4);
  wclose();
  wclose();
  following11();
}
```

```
/*************************************************************************/
void following11(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the definition_3_7 function */
  setonkey(0x4900,Pdef_3_7,0);
  /*********************************************************************/
  /* attach [Pagedown] to the definition_3_8 function */
  setonkey(0x5100,Pdef_3_8,0);
  /*********************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_GREEN,WHITE|_GREEN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_11();
}
/*************************************************************************/
 void example_3_11(void)
 {
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   spawnl(P_WAIT,"ex3_11.exe",NULL);
   srestore(scrn);
   definition_3_8();
 }
/*************************************************************************/
 void definition_3_8(void)
 {
   /*********************************************************************/
   /* attach [Pageup] to the example_3_11 function */
   setonkey(0x4900,Pex_3_11,0);
   /*********************************************************************/
   /* attach [Pagedown] to the example_3_12 function */
   setonkey(0x5100,Pex_3_12,0);
```

```
  if((w[1]=wopen(2,3,8,74,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
    error_exit(1);
  wtitle("[Simple Path]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" A path provides a way of describing how to go from one vertex"
       " to another by following edges. A U-V path need not be an"
       " efficient route; that is, it may repeat vertices or edges."
       " However, a U-V simple path is a path from U to V in which"
       " no vertex is repeated.");
  press_a_key(4);
  if((w[2]=wopen(11,3,18,74,3,LCYAN|_BLUE,WHITE|_BLUE))==0) error_exit(1);
  wtitle("[Simple Path]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" There are no simple paths of length 1 or more from a vertex"
       " to itself. Furthermore, a simple path does not have loops"
       " or pairs of parallel edges in it. In some sense, a simple path"
       " is an efficient route between vertices, whereas a path allows"
       " wandering back and forth, repeating vertices and edges.");
  press_a_key(5);
  wcloseall();
  following12();
 }
/*************************************************************************/
void following12(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the definition_3_8 function */
  setonkey(0x4900,Pdef_3_8,0);
  /*****************************************************************/
  /* attach [Pagedown] to the theorem_3_3 function */
  setonkey(0x5100,Pthm_3_3,0);
  /*****************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_CYAN,WHITE|_CYAN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
```

```
    press_a_key(2);
    wcloseall();
    example_3_12();
}
/*************************************************************************/
 void example_3_12(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_12.exe",NULL);
  srestore(scrn);
  followingthm_3_3();
 }
/*************************************************************************/
void followingthm_3_3(void)
{
  /*************************************************************/
  /* attach [Pageup] to the example_3_12 function */
  setonkey(0x4900,Pex_3_12,0);
  /*************************************************************/
  /* attach [Pagedown] to the definition_3_9 function */
  setonkey(0x5100,Pdef_3_9,0);
  /*************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   theorem_3_3();
}
```

```
/*************************************************************************/
 void theorem_3_3(void)
 {
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   spawnl(P_WAIT,"thm3_3.exe",NULL);
   srestore(scrn);
   definition_3_9();
 }
/*************************************************************************/
static void definition_3_9(void)
{
   /*************************************************************/
   /* attach [Pageup] to the theorem_3_3 function */
   setonkey(0x4900,Pthm_3_3,0);
   /*************************************************************/
   /* attach [Pagedown] to the example_3_13 function */
   setonkey(0x5100,Pex_3_13,0);
   /*************************************************************/
   if((w[1]=wopen(13,3,20,74,3,LCYAN|_BLUE,WHITE|_BLUE))==0) error_exit(1);
   wtitle("[Connected Graph]",TCENTER,_LGREY|BROWN);
   whelpcat(H_WINTITLE);
   wputsw(" Vertices U and V in a multigraph are called connected if"
        " there is a path between them. A multigraph G is connected if"
        " every two vertices of G are connected. Thus, in a connected "
        " multigraph we can go from any one vertex to another by following"
        " some route along the edges.");
   press_a_key(5);
   wclose();
   following13();
}
```

```
/*************************************************************************/
void following13(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the definition_3_9 function */
  setonkey(0x4900,Pdef_3_9,0);
  /*****************************************************************/
  /* attach [Pagedown] to the definition_3_10 function */
  setonkey(0x5100,Pdef_3_10,0);
  /*****************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_RED,WHITE|_RED))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_13();
}
/*************************************************************************/
 void example_3_13(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_13.exe",NULL);
  srestore(scrn);
  definition_3_10();

 }
/*************************************************************************/
static void definition_3_10(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the example_3_13 function */
  setonkey(0x4900,Pex_3_13,0);
  /*****************************************************************/
  /* attach [Pagedown] to the example_3_14 function */
```

```
  setonkey(0x5100,Pex_3_14,0);
  /*****************************************************************/
  if((w[1]=wopen(8,3,14,74,3,LCYAN|_BLUE,WHITE|_MAGENTA))==0)
    error_exit(1);
  wtitle("[A Cycle]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" A path in a graph is called a cycle (or circuit) if the first"
       " and last vertices in the path are the same vertex and no edges"
       " in the path are repeated. Thus a cycle in a graph must have"
       " at least three edges.");
  press_a_key(4);
  wclose();
  following14();
}
/*************************************************************************/
void following14(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the definition_3_10 function */
  setonkey(0x4900,Pdef_3_10,0);
  /*****************************************************************/
  /* attach [Pagedown] to the euler_circuits_and_paths  function */
  setonkey(0x5100,Peuler_circuits_and_paths,0);
  /*****************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_14();
}
```

```
/*************************************************************************/
 void example_3_14(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_14.exe",NULL);
  srestore(scrn);
  euler_circuits_and_paths();
 }
/*************************************************************************/
static void euler_circuits_and_paths(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the example_3_14 function */
  setonkey(0x4900,Pex_3_14,0);
  /*********************************************************************/
  /* attach [Pagedown] to the definition_3_11  function */
  setonkey(0x5100,Pdef_3_11,0);
  /*********************************************************************/
  if((w[1]=wopen(4,3,14,74,3,LCYAN|_BLUE,WHITE|_MAGENTA))==0)
    error_exit(1);
  wtitle("[Euler Circuits and Paths]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" A postman delivers mail everyday in a network of streets."
       " In order to minimize his journey he wishes to know whether or"
       " not he can traverse this network and return to his depot without"
       " walking the length of any street more than once. This problem'
       " concerns the existence or otherwise of an Eulerian circuit of"
       " the corresponding graph. If one exists then he may wish to know"
       " how many others do in order to vary the otherwise tedious"
       " routine.");
  press_a_key(8);
  definition_3_11();
}
```

```
/*************************************************************************/
static void definition_3_11(void)
{
  /*************************************************************/
  /* attach [Pageup] to the euler_circuits_and_paths function */
  setonkey(0x4900,Peuler_circuits_and_paths,0);
  /*************************************************************/
  /* attach [Pagedown] to the example_3_15 function */
  setonkey(0x5100,Pex_3_15,0);
  /*************************************************************/
  if((w[1]=wopen(15,3,22,74,3,LCYAN|_MAGENTA,WHITE|_BLUE))==0)
   error_exit(1);
  wtitle("[Euler Path and Euler Circuit]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" A path in a multigraph G that includes exactly once all"
       " the edges of G and has different first and last vertices is"
       " called an Euler path. A path that includes exactly once all"
       " the edges of G and has the same initial and terminal"
       " vertices is called an Euler circuit.");
  press_a_key(5);
  wcloseall();
  following15();
}
/*************************************************************************/
void following15(void)
{
  /*************************************************************/
  /* attach [Pageup] to the definition_3_11 function */
  setonkey(0x4900,Pdef_3_10,0);
  /*************************************************************/
  /* attach [Pagedown] to the euler_circuit_algorithm function */
  setonkey(0x5100,Peuler_circuit_algorithm,0);
  /*************************************************************/
  if((w[1]=wopen(20,12,24,77,3,LCYAN|_BROWN,WHITE|_BROWN))==0)
    error_exit(1);
  whelpcat(H_WINTITLE);
```

```
    wputsw("Press a key to see the graph....");
    press_a_key(2);
    wcloseall();
    example_3_15();
}
/*****************************************************************************/
 void example_3_15(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_15.exe",NULL);
  srestore(scrn);
  euler_circuit_algorithm();
 }
/*****************************************************************************/
static void euler_circuit_algorithm(void)
{
  /*************************************************************/
  /* attach [Pageup] to the example_3_15 function */
  setonkey(0x4900,Pex_3_15,0);
  /*************************************************************/
  /* attach [Pagedown] to the example_3_16  function */
  setonkey(0x5100,Pex_3_16,0);
  /*************************************************************/
  if((w[1]=wopen(0,3,4,74,3,LCYAN|_BLUE,YELLOW|_BROWN))==0)
    error_exit(1);
  wtitle("[Euler Circuit algorithm]",TCENTER,_LGREY|BROWN);
  wputsw(" This algorithm finds an Euler Circuit for a connected"
         " multigraph G such that every vertex of G has even degree.");
  press_a_key(2);
  if((w[2]=wopen(3,3,6,74,3,LCYAN|_BLUE,WHITE|_BLUE))==0) error_exit(1);
  wtitle("[Step 1]",TCENTER,_LGREY|BROWN);
  wputsw(" (start path). Select a vertex V and an edge on V.");
  press_a_key(1);
  if((w[3]=wopen(6,3,10,74,3,LCYAN|_BLUE,WHITE|_MAGENTA))==0)
    error_exit(1);
```

```
  wtitle("[Step 2]",TCENTER,_LGREY|BROWN);
  wputsw("(get V-V path). If the other vertex on the last chosen edge"
       " is not V, then choose an unused edge on this other vertex."
       " Repeat step 2.");
  press_a_key(2);
  if((w[4]=wopen(10,3,16,74,3,LCYAN|_BLUE,WHITE|_LGREY))==0)
    error_exit(1);
  wtitle("[Step 3]",TCENTER,_LGREY|BROWN);
  wputsw("(start new path at A). If all of the edges nave been used, then"
       " stop (an Euler circuit has been constructed). Otherwise choose"
       " an unused edge on a vertex already visited, and give this"
       " previously visited vertex a temporary name A.");
  press_a_key(4);
  if((w[5]=wopen(16,3,20,74,3,LCYAN|_BLUE,WHITE|_RED))==0) error_exit(1);
  wtitle("[Step 4]",TCENTER,_LGREY|BROWN);
  wputsw("(get A-A path). If the other vertex on the last chosen edge"
       " is not A, then choose an unused edge on this other vertex."
       " Repeat step 4.");
  press_a_key(2);
  if((w[6]=wopen(20,3,24,74,3,LCYAN|_BLUE,WHITE|_CYAN))==0)
    error_exit(1);
  wtitle("[Step 5]",TCENTER,_LGREY|BROWN);
  wputsw("(join paths together). Insert these newly chosen edges at the"
       " vertex A. Go to step 3.");
  press_a_key(2);
  wcloseall();
  following16();
}
```

```
/*************************************************************************/
void following16(void)
{
  /*************************************************************/
  /* attach [Pageup] to the euler_circuit_algorithm function */
  setonkey(0x4900,Peuler_circuit_algorithm,0);
  /*************************************************************/
  /* attach [Pagedown] to the theorem_3_4  function */
  setonkey(0x5100,Pthm_3_4,0);
  /*************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_LGREY,WHITE|_LGREY))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_16();
}
/*************************************************************************/
 void example_3_16(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_16.exe",NULL);
  srestore(scrn);
  theorem_3_4();
 }
```

```
/*************************************************************************/
void theorem_3_4(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the example_3_16 function */
  setonkey(0x4900,Pex_3_16,0);
  /*********************************************************************/
  /* attach [Pagedown] to the example_3_17 function */
  setonkey(0x5100,Pex_3_17,0);
  /*********************************************************************/
 if((w[1]=wopen(3,3,7,74,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
 error_exit(1);
 wtitle("[Theorem 3.4]",TCENTER,_LGREY|BROWN);
 whelpcat(H_WINTITLE);
 wputsw(" A multigraph G has an Eulerian circuit (or path if and only if"
    " it is connected and the number of vertices with odd degree"
    " is 0 (or 2).");
 delay(20);
 if((w[2]=wopen(7,3,13,74,3,LCYAN|_MAGENTA,WHITE|_CYAN))==0)
 error_exit(1);
 wtitle("[PROOF OF THEOREM 3.4]",TCENTER,_LGREY|BROWN);
 wputsw(" The conditions are clearly necessary because if an Eulerian"
    " circuit (or path) exists then G must be connected and only"
    " the vertices at the ends of an Eulerian path can be of odd"
    " degree.\n To show sufficiency we use induction on the number"
    " of edges |E|.");
 delay(20);
 if((w[3]=wopen(15,3,18,74,3,LCYAN|_MAGENTA,WHITE|_LGREY))==0)
 error_exit(1);
 wtitle("[BASE CASE]",TCENTER,_LGREY|BROWN);
 wputsw(" The theorem is trivially true for |e| = 2.");
 press_a_key(1);
 wclose();
 if((w[4]=wopen(13,3,24,74,3,LCYAN|_MAGENTA,WHITE|_LGREY))==0)
 error_exit(1);
 wtitle("[INDUCTIVE STEP]",TCENTER,_LGREY|BROWN);
```

```
wputsw(" Let G have |E| > 2 edges, and let it satisfy the conditions"
    " of the  theorem. If G contains two vertices of odd-degree,"
    " we denote them by V1 and V2. Consider tracing a circuit or"
    " a path T from a vertex Vi ( = V1 if there are vertices of"
    " odd-degree). We trace T leaving each new vertex encountered"
    " by an unused edge until a vertex Vj is encountered for which"
    " every incident edge hes been used. If G contains no vertices"
    " of odd-degree then it must be the case that Vi = Vj, otherwise"
    " it must be the case that Vj = V2.");
press_a_key(9);
wclose();
if((w[5]=wopen(13,3,24,74,3,LCYAN|_MAGENTA,WHITE|_LGREY))==0)
error_exit(1);
wtitle("[INDUCTIVE STEP CONT'D]",TCENTER,_LGREY|BROWN);
wputsw(" Suppose that T does not use every edge of G. If we remove from"
    " G all those edges that have been used, then we are left with a,"
    " not necessarily connected, subgraph G'. G' only contains"
    " vertices of even-degree. By the induction hypothesis each"
    " component of G' contains an Eulerian circuit. Since G is"
    " connected, T must pass through at least one vertex in each"
    " component of G'. An Eulerian circuit or path can then be"
    " constructed for G by inserting into T, at one such vertex"
    " for each component of G', an Eulerian circuit for that"
    " component.");
press_a_key(9);
wclose();
wclose();
wclose();
following17();

}
```

```
/*************************************************************************/
void following17(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the theorem_3_4 function */
  setonkey(0x4900,Pthm_3_4,0);
  /*********************************************************************/
  /* attach [Pagedown] to the definition_3_12  function */
  setonkey(0x5100,Pdef_3_12,0);
  /*********************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_17();
}
/*************************************************************************/
 void example_3_17(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_17.exe",NULL);
  srestore(scrn);
  definition_3_12();
 }
/*************************************************************************/
void hamilton (void)
{
  /*********************************************************************/
  /* attach [Pageup] to the definition_3_12 function */
  setonkey(0x4900,Pdef_3_12,0);
  /*********************************************************************/
  /* attach [Pagedown] to the example_3_18  function */
  setonkey(0x5100,Pex_3_18,0);
```

```
 if((w[1]=wopen(8,3,14,74,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
 error_exit(1);
 wtitle("[Travelling Salesman]",TCENTER,_LGREY|BROWN);
 whelpcat(H_WINTITLE);
 wputsw(" A salesman, starting in his own city, has to visit each of"
     " (n-1) other cities and return home by the shortest route. "
     " A solution to the salesman's problem can be provided by"
     " finding a Hamiltonian circuit.");
 press_a_key(4);
 following18();
}

/*************************************************************************/
void following18(void)
{
   /*********************************************************************/
   /* attach [Pageup] to the hamilton function */
   setonkey(0x4900,Phamilton,0);
   /*********************************************************************/
   /* attach [Pagedown] to the ex_3_18cont  function */
   setonkey(0x5100,Pex_3_18cont,0);
   /*********************************************************************/
    if((w[1]=wopen(20,42,24,77,3,LCYAN|_GREEN,WHITE|_GREEN))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Press a key to see the graph....");
    press_a_key(2);
    wcloseall();
    example_3_18();
}
```

```
/*************************************************************************/
void definition_3_12 (void)
{
  /*****************************************************************/
  /* attach [Pageup] to the example_3_17 function */
  setonkey(0x4900,Pex_3_17,0);
  /*****************************************************************/
  /* attach [Pagedown] to the hamilton  function */
  setonkey(0x5100,Phamilton,0);
  /*****************************************************************/
 if((w[1]=wopen(10,3,13,74,3,LCYAN|_MAGENTA,YELLOW|_BLUE))==0)
 error_exit(1);
 wtitle("[Hamiltonian Cycle]",TCENTER,_LGREY|BROWN);
 whelpcat(H_WINTITLE);
 wputsw(" A Hamiltonian cycle is a cycle that visits every vertex exactly once.");
 press_a_key(1);
 wclose();
 hamilton();
}
/*************************************************************************/
 void example_3_18(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_18.exe",NULL);
  srestore(scrn);
  ex_3_18cont();
 }
/*************************************************************************/
 void ex_3_18cont(void)
 {
  /*****************************************************************/
  /* attach [Pageup] to the example_3_18 function */
  setonkey(0x4900,Pex_3_18,0);
  /*****************************************************************/
  /* attach [Pagedown] to the theorem_3_5  function */
```

```
 setonkey(0x5100,Pthm_3_5,0);
 /*****************************************************************/
wslide(2,3);
if((w[2]=wopen(8,3,21,74,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
error_exit(1);
wtitle("[The criteria]",TCENTER,_LGREY|BROWN);
whelpcat(H_WINTITLE);
wputsw(" Relatively easy criteria exist to determine if there is"
     " an Euler circuit or an Euler path. All that has to be done"
     " is to check the degree of each vertex. Forthermore, there"
     " is a straightforward algorithm to use in constructing an"
     " Euler circuit or path. Unfortunately the same situation"
     " does not hold for Hamiltonian cycles. It is a major unsolved"
     " problem to determine necessary and sufficient conditions"
     " for a graph to have a Hamiltonian cycle. In general it is very"
     " difficult to find a Hamiltonian cycle for a graph. There are,"
     " however, some conditions that guarantee the existence of a "
     " Hamiltonian cycle in a graph. Here is an example of one of"
     " these theorems");
press_a_key(11);
theorem_3_5();
 }
/*************************************************************************/
void theorem_3_5(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the ex_3_18cont function */
  setonkey(0x4900,Pex_3_18cont,0);
  /*****************************************************************/
  /* attach [Pagedown] to the example_3_19  function */
  setonkey(0x5100,Pex_3_19,0);
  /*****************************************************************/
if((w[1]=wopen(20,3,24,74,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
error_exit(1);
wtitle("[Theorem 3.5]",TCENTER,_LGREY|BROWN);
whelpcat(H_WINTITLE);
```

```
  wputsw(" Suppose G is a graph with n vertices, where n > 2. If each"
       " vertex has degree at least n/2, then G has a Hamiltonian"
       " cycle.");
  press_a_key(2);
  wclose();
  wclose();
  wclose();
  following19();
}
/*************************************************************************/
void following19(void)
{
   /*****************************************************************/
   /* attach [Pageup] to the theorem_3_5 function */
   setonkey(0x4900,Pthm_3_5,0);
   /*****************************************************************/
   /* attach [Pagedown] to the definition_3_15  function */
   setonkey(0x5100,Pdef_3_15,0);
   /*****************************************************************/
    if((w[1]=wopen(20,42,24,77,3,LCYAN|_CYAN,WHITE|_CYAN))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Press a key to see the graph....");
    press_a_key(2);
    wcloseall();
    example_3_19();
}
/*************************************************************************/
  void example_3_19(void)
  {
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   spawnl(P_WAIT,"ex3_19.exe",NULL);
   srestore(scrn);
   definition_3_15();
  }
```

```
/*************************************************************************/
 void definition_3_15(void)
 {
   /*****************************************************************/
   /* attach [Pageup] to the example_3_19 function */
   setonkey(0x4900,Pex_3_19,0);
   /*****************************************************************/
   /* attach [Pagedown] to the BFSalgorithm  function */
   setonkey(0x5100,?BFS_algorithm,0);
   /*****************************************************************/
   if((w[1]=wopen(6,10,13,70,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
   error_exit(1);
   wtitle("[The distance]",TCENTER,_LGREY|BROWN);
   whelpcat(H_WINTITLE);
   wputsw(" Sometimes we want to find a path of minimal length between"
        " two vertices S, and T, that is, a path from S to T that has"
        " the smallest possible number of edges. This smallest possible"
        " number of edges in a path from S to T is called the distance"
        " from S to T.");
   press_a_key(5);
   wclose();
   BFS_algorithm();
  }
/*************************************************************************/
static void BFS_algorithm(void)
{
   /*****************************************************************/
   /* attach [Pageup] to the definition_3_15 function */
   setonkey(0x4900,Pdef_3_15,0);
   /*****************************************************************/
   /* attach [Pagedown] to the example_3_39  function */
   setonkey(0x5100,Pex_3_39,0);
   /*****************************************************************/
   if((w[1]=wopen(1,3,7,74,3,LCYAN|_BLUE,YELLOW|_BROWN))==0)
   error_exit(1);
   wtitle("[Breadth-First Search Algorithm]",TCENTER,_LGREY|BROWN);
```

```
 wputsw(" This algorithm will determine the distance and a shortest"
      " path from vertex X to vertex T in a graph G. In the algorithm"
      ", L denotes the set of labeled vertices and the predecessor of"
      " vertex A is a vertex in L that is used in labeling A.");
 press_a_key(4);
 if((w[2]=wopen(6,3,10,74,3,LCYAN|_BLUE,WHITE|_BLUE))==0) error_exit(1);
 wtitle("[Step 1]",TCENTER,_LGREY|BROWN);
 wputsw(" (start with S). Assign S the label 0, let L = {s},"
      " and let S have no predecessor.");
 press_a_key(2);
 if((w[3]=wopen(9,3,16,74,3,LCYAN|_BLUE,WHITE|_MAGENTA))==0)
error_exit(1);
 wtitle("[Step 2]",TCENTER,_LGREY|BROWN);
 wputsw("(check for completion). If T is not in L, go to step 3. If"
      " T is in L, then stop. The label for T is the distance from"
      " S to T. A shortest path from S to T is formed by using in"
      " reverse order the vertices T, the predecessor T1 of T, the"
      " predecessor T2 of T1, and so forth until S is reached.");
 press_a_key(5);
 if((w[4]=wopen(15,3,25,74,3,LCYAN|_BLUE,WHITE|_LGREY))==0)
 error_exit(1);
 wtitle("[Step 3]",TCENTER,_LGREY|BROWN);
 wputsw("(find next vertex). If T is not in L, find the unlabeled"
      " vertices in G that are adjacent to vertices in L with the"
      " largest label number k. If there are no such vertices, then"
      " there is no path from S to T. Otherwise, assign these newly"
      " found vertices the label k+1 and put them in L. If B is one"
      " of these newly found vertices and B is adjacent to a vertex"
      " C in L with label k, let C be the predecessor of B. (If "
      " there is more than one choice for a predecessor, choose one"
      " at random.) Return to step 2.");
 press_a_key(8);
 wclose();
 wclose();
 wclose();
 wclose();
```

```
  following39();
}
/*************************************************************************/
void following39(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the BFS_algorithm function */
  setonkey(0x4900,PBFS_algorithm,0);
  /*****************************************************************/
  /* attach [Pagedown] to the definition_3_16  function */
  setonkey(0x5100,Pdef_3_16,0);
  /*****************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_RED,WHITE|_RED))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_39();
}
/*************************************************************************/
 void definition_3_16(void)
 {
  /*****************************************************************/
  /* attach [Pageup] to the example_3_39 function */
  setonkey(0x4900,Pex_3_39,0);
  /*****************************************************************/
  /* attach [Pagedown] to the example_3_40  function */
  setonkey(0x5100,Pex_3_40,0);
  /*****************************************************************/
  if((w[1]=wopen(6,10,11,70,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
   error_exit(1);
  wtitle("[The weighted graph and the weight]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" A weighted graph is a graph in which a number called the"
         " weight is assigned to each edge. The weight of a path is the"
```

```
        " sum of weights of the edges in the path.");
   press_a_key(3);
   wclose();
   following40();
  }
/*****************************************************************************/
void following40(void)
{
   /*****************************************************************/
   /* attach [Pageup] to the definition  3_16 function */
   setonkey(0x4900,Pdef_3_16,0);
   /*****************************************************************/
   /* attach [Pagedown] to the example_3_41  function */
   setonkey(0x5100,Pex_3_41,0);
   /*****************************************************************/
    if((w[1]=wopen(20,42,24,77,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Press a key to see the graph....");
    press_a_key(2);
    wcloseall();
    example_3_40();
}

/*****************************************************************************/

 void example_3_39(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_39.exe",NULL);
  srestore(scrn);
  definition_3_16();

 }
```

```
/*************************************************************************/
 void example_3_40(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_40.exe",NULL);
  srestore(scrn);
  following41();
 }
/*************************************************************************/
void following41(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the example_3_40 function */
  setonkey(0x4900,Pex_3_40,0);
  /*********************************************************************/
  /* attach [Pagedown] to the dijkstra_algorithm  function */
  setonkey(0x5100,Pdijkstra_algorithm,0);
  /*********************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYANl_BROWN,WHITEl_BROWN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_41();
}
/*************************************************************************/
 void example_3_41(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_41.exe",NULL);
  srestore(scrn);
  dijkstra_algorithm();
 }
```

```
/*************************************************************************/
static void dijkstra_algorithm(void)
{
 /*****************************************************************/
  /* attach [Pageup] to the example_3_41 function */
  setonkey(0x4900,Pex_3_41,0);
  /*****************************************************************/
  /* attach [Pagedown] to the example_3_42  function */
  setonkey(0x5100,Pex_3_42,0);
  /*****************************************************************/
  if((w[1]=wopen(1,3,11,74,3,LCYAN|_BLUE,YELLOW|_BROWN))==0)
  error_exit(1);
  wtitle("[Dijkstra's Algorithm]",TCENTER,_LGREY|BROWN);
  wputsw(" Let G be a weighted graph in which there is more than one"
       " vertex and there are nonnegative weights on the edges. This"
       " algorithm will determine the distance and a shortest path"
       " from vertex S to every other vertex in G. In the algorithm"
       " P denotes the set of vertices with permanent labels. The"
       " predecessor of a vertex A is a vertex in P used to label A"
       ". The weight of the edge on vertices U and V will be denoted"
       " by W(U,V), and if there is no edge on U and V, we will write"
       " W(U,V) = inf.");
  press_a_key(8);
  if((w[2]=wopen(10,3,14,74,3,LCYAN|_BLUE,WHITE|_BLUE))==0)
   error_exit(1);
  wtitle("[Step 1]",TCENTER,_LGREY|BROWN);
  wputsw(" (start with S). Assign S the label 0, let P = {s},"
            " and let S have no predecessor.");
  press_a_key(2);
  if((w[3]=wopen(13,3,17,74,3,LCYAN|_CYAN,WHITE|_CYAN))==0)
  error_exit(1);
  wtitle("[Step 2]",TCENTER,_LGREY|BROWN);
  wputsw("(assign labels to all the other vertices). To each vertex V"
            " not in P, assign the label W(S,V), and let V have"
            " predecessor S.");
  press_a_key(2);
```

```
if((w[4]=wopen(16,3,24,74,3,LCYAN|_BLUE,WHITE|_MAGENTA))==0)
error_exit(1);
wtitle("[Step 3]",TCENTER,_LGREY|BROWN);
wputsw("(find nearest vertex to P and revise labels). Include in P"
    " a vertex U having the smallest label of the vertices not"
    " in P. (If there is more than one such vertex, arbitrarily"
    " choose any one of them.) For each vertex X not in P and"
    " adjacent to U, replace the label on X by the smaller of the"
    " old label on X and (label on U) + W(U,X). If the label on X"
    " was changed, let U be the new predecessor of X.");
press_a_key(6);
wactiv(w[2]);
wslide(1,3);
wactiv(w[3]);
wslide(4,3);
wactiv(w[4]);
wslide(7,3);
if((w[5]=wopen(15,3,24,74,3,LCYAN|_BLUE,WHITE|_LGREY))==0)
error_exit(1);
wtitle("[Step 4]",TCENTER,_LGREY|BROWN);
wputsw("(check for completion). If P does not contain all the"
    " vertices of G, then return to step 3. Otherwise the label"
    " on a vertex Y is its distance from S. If the label on Y is"
    " inf, then there is no path, hence, no shortest path, from"
    " S to Y. Otherwise, a shortest path from S to Y is formed by"
    " using in reverse order the vertices Y, the predecessor Y1"
    " of Y, the predecessor Y2 of Y1, and so forth until S is"
    " reached.");
press_a_key(7);
wclose();
wclose();
wclose();
wclose();
wclose();
following42();
}
```

```
/*************************************************************************/
void following42(void)
{
  /*************************************************************/
  /* attach [Pageup] to the dijkstra_algorithm function */
  setonkey(0x4900,Pdijkstra_algorithm,0);
  /*************************************************************/
  /* attach [Pagedown] to the example_3_30  function */
  setonkey(0x5100,Pex_3_30,0);
  /*************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_LGREY,WHITE|_LGREY))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_42();
}
/*************************************************************************/
 void example_3_42(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_42.exe",NULL);
  srestore(scrn);
  following30();
 }
/*************************************************************************/
void following30(void)
{
  /*************************************************************/
  /* attach [Pageup] to the example_3_42 function */
  setonkey(0x4900,Pex_3_42,0);
  /*************************************************************/
  /* attach [Pagedown] to the example_3_31  function */
  setonkey(0x5100,Pex_3_31,0);
```

```
    if((w[1]=wopen(20,42,24,77,3,LCYAN|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Press a key to see the graph....");
    press_a_key(2);
    wcloseall();
    example_3_30();
}
/*************************************************************************/
void isomorphism (void)
{
   /*****************************************************************/
   /* attach [Pageup] to the example_3_31 function */
   setonkey(0x4900,Pex_3_31,0);
   /*****************************************************************/
   /* attach [Pagedown] to the example_3_32  function */
   setonkey(0x5100,Pex_3_32,0);
   /*****************************************************************/
 if((w[1]=wopen(8,3,14,74,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
 error_exit(1);
 wtitle("[ Isomorphism ]",TCENTER,_LGREY|BROWN);
 whelpcat(H_WINTITLE);
 wputsw(" When we believe that two graphs are isomorphic, we can justify"
     " that belief by constructing an isomorphism. When we suspect that"
     " two graphs are not isomorphic, we can substantiate that suspicion"
     " only by showing that no isomorphism exists.");
 press_a_key(4);
 wslide(2,3);
 if((w[2]=wopen(8,3,15,74,3,LCYAN|_BLUE,WHITE|_BLUE))==0)
 error_exit(1);
 wtitle("[ Isomorphism ]",TCENTER,_LGREY|BROWN);
 whelpcat(H_WINTITLE);
 wputsw(" In some cases this is easy. For instance, if the graphs do not"
       " have the same number of vertices, then there can be no bijection"
       " between the sets of vertices, and hence no isomorphism."
       " Similarly, if the graphs do not have the same number of"
```

```
      " edges, the graphs cannot be isomorphic.");
 press_a_key(5);
 if((w[3]=wopen(15,3,24,74,3,LCYAN|_LGREY,WHITE|_LGREY))==0)
 error_exit(1);
 wtitle("[ Isomorphism ]",TCENTER,_LGREY|BROWN);
 whelpcat(H_WINTITLE);
 wputsw(" There are other properties that must be common between"
      " isomorphic graphs that we can check (connectivity for example)."
      " Otherwise, verifying that two graphs are nonisomorphic"
      " can be difficult. Suppose G1 = (V1,E1) and G2 = (V2,E2)."
      " If |V1| = |V2| = n, than there are n! possible bijections"
      " between V1 and V2. To show that G1 is not isomorphic to G2,"
      " we must show that everyone of the n! bijections fails to map"
      " the edges of G1 onto the edges of G2.");
 press_a_key(7);
 wclose();
 wclose();
 wclose();
 following32();
}
/*************************************************************************/
void following32(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the isomorphism function */
  setonkey(0x4900,Pisomorphism,0);
  /*****************************************************************/
  /* attach [Pagedown] to the exercise1  function */
  setonkey(0x5100,Pexercise1,0);
  /*****************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
```

```
    example_3_32();
}
/*************************************************************************/
 void example_3_30(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_30.exe",NULL);
  srestore(scrn);
  following31();
 }
/*************************************************************************/
void following31(void)
{
  /*************************************************************/
  /* attach [Pageup] to the example_3_30 function */
  setonkey(0x4900,Pex_3_30,0);
  /*************************************************************/
  /* attach [Pagedown] to the isomorphism  function */
  setonkey(0x5100,Pisomorphism,0);
  /*************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_GREEN,WHITE|_GREEN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see the graph....");
   press_a_key(2);
   wcloseall();
   example_3_31();
}
/*************************************************************************/
 void example_3_31(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_31.exe",NULL);
  srestore(scrn);
```

```
   isomorphism();
  }
/*************************************************************************/
  void example_3_32(void)
  {
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   spawnl(P_WAIT,"ex3_32.exe",NULL);
   srestore(scrn);
   exercise1();
  }
/*************************************************************************/
void exercise1(void)
{
   register int *scrn;
   /*********************************************************************/
   /* attach [Pageup] to the example_3_32 function */
   setonkey(0x4900,Pex_3_32,0);
   /*********************************************************************/
   /* attach [Pagedown] to the exercise2 function */
   setonkey(0x5100,Pexercise2,0);
   /*********************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see some exercises...");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"pq1.exe",NULL);
    srestore(scrn);
    exercise2();
}
```

```
/*************************************************************************/
void exercise2(void)
{
   register int *scrn;
   /*********************************************************************/
   /* attach [Pageup] to the exercise1 function */
   setonkey(0x4900,Pexercise1,0);
   /*********************************************************************/
   /* attach [Pagedown] to the exercise3 function */
   setonkey(0x5100,Pexercise3,0);
   /*********************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 2..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"pq2.exe",NULL);
    srestore(scrn);
    exercise3();
}
/*************************************************************************/
void exercise3(void)
{
   register int *scrn;
   /*********************************************************************/
   /* attach [Pageup] to the exercise2 function */
   setonkey(0x4900,Pexercise2,0);
   /*********************************************************************/
   /* attach [Pagedown] to the exercise4 function */
   setonkey(0x5100,Pexercise4,0);
   /*********************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
```

```
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 3..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"pq3.exe",NULL);
    srestore(scrn);
    exercise4();
}
/*************************************************************************/
void exercise4(void)
{
   register int *scrn;
   /*************************************************************/
   /* attach [Pageup] to the exercise3 function */
   setonkey(0x4900,Pexercise3,0);
   /*************************************************************/
   /* attach [Pagedown] to the exercise5 function */
   setonkey(0x5100,Pexercise5,0);
   /*************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 4..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"pq4.exe",NULL);
    srestore(scrn);
    exercise5();
}
```

```
/*************************************************************************/
void exercise5(void)
{
  register int *scrn;
  /*****************************************************************/
  /* attach [Pageup] to the exercise4 function */
  setonkey(0x4900,Pexercise4,0);
  /*****************************************************************/
  /* attach [Pagedown] to the exercise6 function */
  setonkey(0x5100,Pexercise6,0);
  /*****************************************************************/
  if((scrn=ssave())==NULL) error_exit(3);
   if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Now let us see exercise 5..");
   press_a_key(2);
   wcloseall();
   spawnl(P_WAIT,"pq5.exe",NULL);
   srestore(scrn);
   exercise6();
}
/*************************************************************************/
void exercise6(void)
{
  register int *scrn;
  /*****************************************************************/
  /* attach [Pageup] to the exercise5 function */
  setonkey(0x4900,Pexercise5,0);
  /*****************************************************************/
  /* attach [Pagedown] to the exercise7 function */
  setonkey(0x5100,Pexercise7,0);
  /*****************************************************************/
  if((scrn=ssave())==NULL) error_exit(3);
   if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
```

```
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 6..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"pq6.exe",NULL);
    srestore(scrn);
    exercise7();
}
/*************************************************************************/
void exercise7(void)
{
   register int *scrn;
   /*****************************************************************/
   /* attach [Pageup] to the exercise6 function */
   setonkey(0x4900,Pexercise6,0);
   /*****************************************************************/
   /* attach [Pagedown] to the exercise8 function */
   setonkey(0x5100,Pexercise8,0);
   /*****************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 7..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"pq7.exe",NULL);
    srestore(scrn);
    exercise8();
}
```

```
/*************************************************************************/
void exercise8(void)
{
  register int *scrn;
  /*************************************************************/
  /* attach [Pageup] to the exercise7 function */
  setonkey(0x4900,Pexercise7,0);
  /*************************************************************/
  /* attach [Pagedown] to the exercise9 function */
  setonkey(0x5100,Pexercise9,0);
  /*************************************************************/
  if((scrn=ssave())==NULL) error_exit(3);
   if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Now let us see exercise 8..");
   press_a_key(2);
   wcloseall();
   spawnl(P_WAIT,"pq8.exe",NULL);
   srestore(scrn);
   exercise9();
}
/*************************************************************************/
void exercise9(void)
{
  register int *scrn;
  /*************************************************************/
  /* attach [Pageup] to the exercise8 function */
  setonkey(0x4900,Pexercise8,0);
  /*************************************************************/
  /* attach [Pagedown] to the exercise10 function */
  setonkey(0x5100,Pexercise10,0);
  /*************************************************************/
  if((scrn=ssave())==NULL) error_exit(3);
   if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
```

```
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 9..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"pq9.exe",NULL);
    srestore(scrn);
    exercise10();
}
/*****************************************************************************/
void exercise10(void)
{
   register int *scrn;
   /*************************************************************/
   /* attach [Pageup] to the exercise9 function */
   setonkey(0x4900,Pexercise9,0);
   /*************************************************************/
   /* attach [Pagedown] to the exercise10 function */
   setonkey(0x5100,Pexercise10,0);
   /*************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 10..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"pq10.exe",NULL);
    srestore(scrn);
    normal_exit();
}
```

```
/*************************************************************************/
/* this routine calls introduction routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pintroduction()
{
  wcloseall();
  introduction();
}
/*************************************************************************/
/* this routine calls definition_3_6 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_6()
{
  wcloseall();
  definition_3_6();
}
/*************************************************************************/
/* this routine calls example_3_10 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_10()
{
  wcloseall();
  following10();
}
/*************************************************************************/
/* this routine calls definition_3_7 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_7()
{
  wcloseall();
  definition_3_7();
}
```

```
/*************************************************************************/
/* this routine calls example_3_11 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_11()
{
   wcloseall();
   following11();
}
/*************************************************************************/
/* this routine calls example_3_12 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_12()
{
   wcloseall();
   following12();
}
/*************************************************************************/
/* this routine calls example_3_13 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_13()
{
   wcloseall();
   following13();
}
/*************************************************************************/
/* this routine calls example_3_14 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_14()
{
   wcloseall();
   following14();
}
```

```
/*************************************************************************/
/* this routine calls example_3_15 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_15()
{
   wcloseall();
   following15();
}
/*************************************************************************/
/* this routine calls example_3_16 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_16()
{
   wcloseall();
   following16();
}
/*************************************************************************/
/* this routine calls example_3_17 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_17()
{
   wcloseall();
   following17();
}
/*************************************************************************/
/* this routine calls example_3_18 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_18()
{
   wcloseall();
   following18();
}
```

```
/*************************************************************************/
/* this routine calls example_3_19 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_19()
{
  wcloseall();
  following19();
}
/*************************************************************************/
/* this routine calls example_3_30 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_30()
{
  wcloseall();
  following30();
}
/*************************************************************************/
/* this routine calls example_3_31 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_31()
{
  wcloseall();
  following31();
}
/*************************************************************************/
/* this routine calls example_3_32 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_32()
{
  wcloseall();
  following32();
}
```

```
/*************************************************************************/
/* this routine calls example_3_39 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_39()
{
   wcloseall();
   following39();
}
/*************************************************************************/
/* this routine calls example_3_40 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_40()
{
   wcloseall();
   following40();
}
/*************************************************************************/
/* this routine calls example_3_41 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_41()
{
   wcloseall();
   following41();
}
/*************************************************************************/
/* this routine calls example_3_42 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pex_3_42()
{
   wcloseall();
   following42();
}
```

```
/*************************************************************************/
/* this routine calls definition_3_8 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_8()
{
  wcloseall();
  definition_3_8();
}
/*************************************************************************/
/* this routine calls definition_3_9 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_9()
{
  wcloseall();
  definition_3_9();
}
/*************************************************************************/
/* this routine calls definition_3_10 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                      */
/*************************************************************************/
void Pdef_3_10()
{
  wcloseall();
  definition_3_10();
}
/*************************************************************************/
/* this routine calls definition_3_11 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                      */
/*************************************************************************/
void Pdef_3_11()
{
  wcloseall();
  definition_3_11();
}
```

```
/*************************************************************************/
/* this routine calls definition_3_12 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                      */
/*************************************************************************/
void Pdef_3_12()
{
  wcloseall();
  definition_3_12();
}
/*************************************************************************/
/* this routine calls definition_3_16 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                      */
/*************************************************************************/
void Pdef_3_16()
{
  wcloseall();
  definition_3_16();
}
/*************************************************************************/
/* this routine calls theorem_3_3 routine whenever Pageup or Pagedown     */
/* keys are pressed.                                                      */
/*************************************************************************/
void Pthm_3_3()
{
  wcloseall();
  followingthm_3_3();
}
/*************************************************************************/
/* this routine calls theorem_3_4 routine whenever Pageup or Pagedown     */
/* keys are pressed.                                                      */
/*************************************************************************/
void Pthm_3_4()
{
  wcloseall();
  theorem_3_4();
}
```

```
/*************************************************************************/
/* this routine calls theorem_3_5 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pthm_3_5()
{
  wcloseall();
  theorem_3_5();
}
/*************************************************************************/
/* this routine calls euler_circuits_and_paths routine whenever Pageup or*/
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Peuler_circuits_and_paths()
{
  wcloseall();
  euler_circuits_and_paths();
}
/*************************************************************************/
/* this routine calls euler_circuit_algorithm routine whenever Pageup or */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Peuler_circuit_algorithm()
{
  wcloseall();
  euler_circuit_algorithm();
}
/*************************************************************************/
/* this routine calls hamilton routine whenever Pageup or                */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Phamilton()
{
  wcloseall();
  hamilton();
}
```

```
/*************************************************************************/
/* this routine calls BFS_algorithm routine whenever Pageup or           */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void PBFS_algorithm()
{
  wcloseall();
  BFS_algorithm();
}
/*************************************************************************/
/* this routine calls dijkstra_algorithm routine whenever Pageup or      */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pdijkstra_algorithm()
{
  wcloseall();
  dijkstra_algorithm();
}
/*************************************************************************/
/* this routine calls isomorphism routine whenever Pageup or             */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pisomorphism()
{
  wcloseall();
  isomorphism();
}
/*************************************************************************/
/* this routine calls exercise1 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise1()
{
  wcloseall();
  exercise1();
}
```

```
/*************************************************************************/
/* this routine calls exercise2 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise2()
{
  wcloseall();
  exercise2();
}
/*************************************************************************/
/* this routine calls exercise3 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise3()
{
  wcloseall();
  exercise3();
}
/*************************************************************************/
/* this routine calls exercise4 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise4()
{
  wcloseall();
  exercise4();
}
/*************************************************************************/
/* this routine calls exercise5 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise5()
{
  wcloseall();
  exercise5();
}
```

```
/*************************************************************************/
/* this routine calls exercise6 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise6()
{
  wcloseall();
  exercise6();
}
/*************************************************************************/
/* this routine calls exercise7 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise7()
{
  wcloseall();
  exercise7();
}
/*************************************************************************/
/* this routine calls exercise8 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise8()
{
  wcloseall();
  exercise8();
}
/*************************************************************************/
/* this routine calls exercise9 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise9()
{
  wcloseall();
  exercise9();
}
```

```
/*************************************************************************/
/* this routine calls exercise10 routine whenever Pageup or              */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pexercise10()
{
  wcloseall();
  exercise10();
}
/*************************************************************************/
/* this routine calls ex_3_18cont routine whenever Pageup or             */
/* Page down keys are pressed.                                           */
/*************************************************************************/
void Pex_3_18cont()
{
  wcloseall();
  ex_3_18cont();
}
/*************************************************************************/
/* this routine calls definition_3_15 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void Pdef_3_15()
{
  wcloseall();
  definition_3_15();
}
```

```
/* PROGRAM  : ex3_10.c
  AUTHOR   : Yavuz BAS
  DATE     : Feb. 4, 1990
  REVISED  : Mar. 5, 1990

  DESCRIPTION : This program contains an example about the "Paths and
Circuits".

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void example_3_10  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);


        /*************************************************
                        MAIN PROGRAM
        *************************************************/


main()
{
      cls (1);
      adapt = getAdapter();
     if (adapt == 'V')
     {
        setMode(0x12);
        cls(1);
     }
     if (adapt == 'E')
     {
         setMode(16);
         cls(1);
     }
      if (adapt == 'C')
     {
         setMode(4);
          setCGAPalette(0);
          cls(1);
     }
     example_3_10();
     setMode(3);
}
```

```
/*************************************************************************/
static void example_3_10(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,160," The diagram in the following figure"
        " represents a multigraph but not a",31);
    write_horz_str(-300,140," graph because there are two parallel edges"
        " k and m between the",31);
    write_horz_str(-300,120," vertices Y and Z and a loop h at vertex"
        " X",31);
    drawLine(-100,-50,100,-50,7);
    drawLine(100,-50,120,50,11);
    drawLine(120,50,-120,50,11);
    aspect = .20;
    LINEWIDTH = 1;
    drawArc(0,-50,20,0,1760,7,aspect);
    aspect = 1.0;
    drawOval(140,70,25,7,aspect);
    fillOval(-100,-50,3,16,aspect);
    fillOval(100,-50,3,16,aspect);
    fillOval(120,50,3,16,aspect);
    fillOval(-120,50,3,16,aspect);
    write_horz_char(-110,-60,90,31); /* Z */
    write_horz_char(110,-60,89,31);  /* Y */
    write_horz_char(130,65,88,31);   /* X */
    write_horz_char(-130,60,85,31);  /* U */
    write_horz_char(0,65,103,31);    /* g */
    write_horz_char(165,95,104,31);  /* h */
    write_horz_char(120,0,106,31);   /* j */
    write_horz_char(0,-12,107,31);   /* k */
    write_horz_char(0,-57,109,31);   /* m */
    write_horz_str(-300,-90," In a multigraph the number of edges"
        " incident with a vertex V is called the",31);
    write_horz_str(-300,-110," degree of V and is denoted as deg(V)."
        " A loop on a vertex V is counted",31);
    write_horz_str(-300,-130," twice in deg(V). Thus, in the Figure"
        " deg(Y) = 3 and deg(X) = 4.",31);
    wait("");
}
```

```
/*************************************************************************/

void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
     line = 29;
     else
     line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': setMode(3);
           videoinit();
           normal_exit();
```

```
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

    case 'n':write_horz_str(-300,-210,"                    ",31);
            break;

    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/

static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : ex3_11.c
  AUTHOR   : Yavuz BAS
  DATE     : Feb. 4, 1990
  REVISED  : Mar. 5, 1990

  DESCRIPTION : This program contains an example about the "Paths and
Circuits".

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void example_3_11  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                              MAIN PROGRAM
        *************************************************/


main()
{
     cls (1);
     adapt = getAdapter();
    if (adapt == 'V')
    {
       setMode(0x12);
       cls(1);
    }
    if (adapt == 'E')
    {
       setMode(16);
       cls(1);
    }
    if (adapt == 'C')
    {
       setMode(4);
       setCGAPalette(0);
       cls(1);
    }
    example_3_11();
    setMode(3);
}
```

```
/*************************************************************************/
static void example_3_11(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,160," In the following figure U, g, X "
                            " is a path of length 1 from U to X. This",31);
    write_horz_str(-300,140," path can also be written as  g."
                            " Likewise  g, h is a path of length 2",31);
    write_horz_str(-300,120," from U to X, and U, g, X, g, U is a path"
                            " of length 2 from U to U.",31);
    write_horz_str(-300,100," The path Z, m, Y cannot be described by just"
                            " listing Z, Y since it would",31);
    write_horz_str(-300,80, " not be clear which edge between Z and Y,"
                           " k or m, is part of the path.",31);


    drawLine(-100,-100,100,-100,7);
    drawLine(100,-100,120,0,11);
    drawLine(120,0,-120,0,11);
    aspect = .20;
    LINEWIDTH = 1;
    drawArc(0,-100,20,0,1760,7,aspect);
    aspect = 1.0;
    drawOval(140,20,25,7,aspect);
    fillOval(-100,-100,3,16,aspect);
    fillOval(100,-100,3,16,aspect);
    fillOval(120,0,3,16,aspect);
    fillOval(-120,0,3,16,aspect);
    write_horz_char(-110,-110,90,31); /* Z */
    write_horz_char(110,-110,89,31);  /* Y */
    write_horz_char(130,15,88,31);    /* X */
    write_horz_char(-130,10,85,31);   /* U */
    write_horz_char(0,15,103,31);     /* g */
    write_horz_char(165,45,104,31);   /* h */
    write_horz_char(120,-50,106,31);  /* j */
    write_horz_char(0,-62,107,31);    /* k */
    write_horz_char(0,-107,109,31);   /* m */

    wait("");
}
```

```
/*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width)
     if ((mode == 0x11) || (mode == 0x12))
     line = 29;
     else
     line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
```

```
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;

    case 'n':write_horz_str(-300,-210,"                    ",31);
              break;

    case 'N':write_horz_str(-300,-210,"                    ",31);
              break;

    default : break;
    }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/

static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_12.c
   AUTHOR       : Yavuz BAS
   DATE         : Feb. 4, 1990
   REVISED      : Mar. 5, 1990

   DESCRIPTION  : This program contains an example about the "Paths and
Circuits".

 MACHINE/COMPILER : This program is written with IBM pc by using Turbo      C
compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
/************************************************
                FUNCTION DEFINITIONS
************************************************/

void wait(char title[]);
void example_3_12  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);


/************************************************
                    MAIN PROGRAM
************************************************/

main()
{
    cls (1);
    adapt = getAdapter();
    if (adapt == 'V')
    {
      setMode(0x12);
      cls(1);
    }
    if (adapt == 'E')
    {
       setMode(16);
       cls(1);
    }
    if (adapt == 'C')
    {
       setMode(4);
       setCGAPalette(0);
       cls(1);
    }
    example_3_12();
    setMode(3);
}
```

```
/*************************************************************************/
static void example_3_12(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,160," For the graph in the following figure"
                            " the edges a, c, d, j form a simple",31);
    write_horz_str(-300,140," path from U to Z, whereas a, c, m, d, j is a"
                            " path from U to Z that is not",31);
    write_horz_str(-300,120," a simple path because the vertex W is"
                            " repeated. Similarly, e, i is a",31);
     writ e_horz_str(-300,100," simple path from X to Z, but f, i, j is a"
                            " path from X to Z that is not a",31);
    write_horz_str(-300,80," simple path. Note also that c, p, f, i, e,"
                           " n is a path from V to U that is",31);
    write_horz_str(-300,60," not simple, but deleting f, i, e produces"
                           " a simple path c, p, n from V to U.",31);
    write_horz_str(-300,40," This illustrates the following result.",31);


    drawLine(-110,-10,40,-10,11);   /* c */
    drawLine(-110,-10,-140,-50,11); /* b */
    drawLine(-140,-50,30,-60,11);   /* n */
    drawLine(30,-60,40,-10,11);     /* p */
    drawLine(30,-60,30,-150,11);    /* f */
    drawLine(30,-150,100,-80,11);   /* i */
    drawLine(30,-60,100,-80,11);    /* e */
    drawLine(40,-10,100,-80,11);    /* d */
    drawLine(30,-150,-140,-85,11);  /* h */
    drawLine(-140,-85,-140,-50,11); /* g */

    aspect = .30;
    LINEWIDTH = 1;
    drawArc(65,-115,37,2500,730,11,aspect);
    aspect = 2.0;
    drawArc(-125,-34,38,400,2000,11,aspect);
    aspect = 1.0;
    drawOval(58,8,25,11,aspect);
    fillOval(-110,-10,3,16,aspect);
    fillOval(40,-10,3,16,aspect);
    fillOval(-140,-50,3,16,aspect);
    fillOval(30,-60,3,16,aspect);
```

```
    fillOval(30,-150,3,16,aspect);
    fillOval(100,-80,3,16,aspect);
    fillOval(-140,-85,3,16,aspect);
    write_horz_char(30,-160,90,31); /* Z */
    write_horz_char(105,-85,89,31); /* Y */
    write_horz_char(10,-40,88,31);  /* X */
    write_horz_char(50,0,87,31);    /* W */
    write_horz_char(-100,5,86,31);  /* V */
    write_horz_char(-160,-50,85,31);  /* U */
    write_horz_char(-150,-85,82,31);  /* R */
    write_horz_char(-160,-20,97,31);  /* a */
    write_horz_char(-120,-30,98,31);  /* b */
    write_horz_char(-60,5,99,31);  /* c */
    write_horz_char(80,-35,100,31);  /* d */
    write_horz_char(60,-75,101,31);  /* e */

    write_horz_char(40,-105,102,31);  /* f */
    write_horz_char(-135,-62,103,31);  /* g */
    write_horz_char(-45,-128,104,31);  /* h */
    write_horz_char(60,-100,105,31);  /* i */
    write_horz_char(160,-100,106,31);  /* j */
    write_horz_char(75,40,109,31);  /* m */
    write_horz_char(-55,-65,110,31);  /* n */
    write_horz_char(40,-35,112,31);  /* p */
    wait("");
}



/*****************************************************************************/

void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
     line = 29;
     else
     line = 24;
     tab = (width - strlen(title))/2;
```

```
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
/*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
      write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;

    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

    case 'n':write_horz_str(-300,-210,"                          ",31);
            break;
```

```
    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;

    default : break;
    }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : thm3_3.c
   AUTHOR     : Yavuz BAS
   DATE       : Feb. 4, 1990
   REVISED    : Mar. 5, 1990

   DESCRIPTION : This program contains a theorem about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/



char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void theorem_3_3  (void);
void confirm_graph_exit(void);
void normal_exit(void);
        /*************************************************
                        MAIN PROGRAM
        *************************************************/
main()
{
    cls (1);
    adapt = getAdapter();
     if (adapt == 'V')
       {
          setMode(0x12);
          cls(1);
       }
    if (adapt == 'E')
       {
          setMode(16);
          cls(1);
       }
    if (adapt == 'C')
       {
          setMode(4);
          setCGAPalette(0);
          cls(1);
       }
   theorem_3_3();
   setMode(3);
}
```

```
/*************************************************************************/
static void theorem_3_3(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,200," THEOREM 3.3 ",11);
    write_horz_str(-190,200,"Every U-V path contains a U_V simple path",31);
    write_horz_str(-300,180," Proof: ",11);
    write_horz_str(-235,180," Let's suppose that U = V1,e1,V2,...,en,Vn+1"
                            " = V is a U-V path. In",31);
    write_horz_str(-300,165," the special case that U = V we can choose"
                            " our U-V simple path to be",31);
    write_horz_str(-300,150," just the vertex U. Now suppose U and V are"
                            " different. If all the vertices",31);
    write_horz_str(-300,135," V1,...,Vn+1 are different initially, then"
                            " our path is already a U-V",31);
    write_horz_str(-300,120," simple path. Thus,let's suppose that"
                            " at least two of the vertices are",31);
    write_horz_str(-300,105," the same, say that Vi = Vj where i < j. See"
                            " the Figure for an illustration",31);
    write_horz_str(-300,90," of how a path from Vi to Vj is found.",31);

    drawLine(-60,70,0,20,31);
    drawLine(60,70,0,20,31);
    drawLine(0,20,-100,-30,31);
    drawLine(0,20,100,-30,31);

    aspect = 1.0;
    fillOval(0,70,3,16,aspect);
    fillOval(-20,70,3,16,aspect);
    fillOval(20,70,3,16,aspect);
    fillOval(-60,70,3,16,aspect);
    fillOval(60,70,3,16,aspect);
    fillOval(0,20,3,16,aspect);
    fillOval(-100,-30,3,16,aspect);
    fillOval(100,-30,3,16,aspect);

    write_horz_char(-30,60,101,31);   /* e */
    write_horz_str(-23,55,"j-1",31);  /* e sub J-1 */
    write_horz_char(45,50,101,31);    /* e */
```

```
    write_horz_char(52,45,105,31);      /* e sub i */
    write_horz_char(60,10,101,31);      /* e */
    write_horz_char(67,5,106,31);       /* e sub j */
    write_horz_char(-50,-10,101,31);    /* e */
    write_horz_str(-43,-15,"i-1",31);   /* e sub i-1 */
    write_horz_str(-20,5,"V = V",31);   /* V = V */
    write_horz_char(-13,0,105,31);      /* V sub i*/
    write_horz_char(20,0,106,31);       /* V sub j */
    write_horz_char(-100,-35,86,31);    /* V */
    write_horz_char(100,-35,86,31);     /* V */
    write_horz_str(-93,-40,"i-1",31);   /* V sub i-1 */
    write_horz_str(107,-40,"j+1",31);   /* V sub j+1 */

    write_horz_str(-300,-60," We delete ei, Vi+1,...,ej-1,Vj from"
                            " the original path. What has been",31);
    write_horz_str(-300,-75," deleted is the part that is between"
                            " vertex Vi and edge ej. This still",31);
    write_horz_str(-300,-90," leaves a path from U to V. If there are"
                            " only distinct vertices left after",31);
    write_horz_str(-300,-105," this deletion, then we are done. If there"
                             " are still repetitions among the",31);
    write_horz_str(-300,-120," remaining vertices, the above process"
                             " is repeated. Because the number of",31);
    write_horz_str(-300,-135," vertices is finite, this process will"
                             " eventually end and give a U-V simple",31);
    write_horz_str(-300,-150," path from U to V.",31);
    wait("Theorem 3.3");
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
      line = 29;
    else
      line = 24;
     tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
```

```
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    getch();
    cls(0);
}
/*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
  ch = getch ():
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(-80,190," Please type y or n",31);
     ch = getch ();
   }
   switch (ch)        {
    case 'y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'Y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'n':
           break;
    case 'N':
           break;
    default : break;     }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
```

```
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_13.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
            GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
/*************************************************
                FUNCTION DEFINITIONS
 *************************************************/


void wait(char title[]);
void example_3_13  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);


/*************************************************
                  MAIN PROGRAM
 *************************************************/


main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
     {
        setMode(0x12);
        cls(1);
     }
     if (adapt == 'E')
     {
       setMode(16);
       cls(1);
     }
     if (adapt == 'C')
     {
       setMode(4);
        setCGAPalette(0);
        cls(1);
     }
     example_3_13();
     setMode(3);
}
```

```
/*****************************************************************************/
static void example_3_13(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,180," The multigraph in the following figure"
        " is connected since a path can be",31);
    write_horz_str(-300,165," found between any two vertices.",31);
    drawLine(-110,100,40,100,11);   /* c */
    drawLine(-110,100,-140,60,11);  /* b */
    drawLine(-140,60,30,50,11);     /* n */
    drawLine(30,50,40,100,11);      /* p */
    drawLine(30,50,30,-40,11);      /* f */
    drawLine(30,-40,100,30,11);     /* i */
    drawLine(30,50,100,30,11);      /* e */
    drawLine(40,100,100,30,11);     /* d */
    drawLine(30,-40,-140,25,11);    /* h */
    drawLine(-140,25,-140,60,11);   /* g */

    aspect = .30;
    LINEWIDTH = 1;
    drawArc(65,-5,37,2500,730,11,aspect);
    aspect = 2.0;
    drawArc(-125,76,38,400,2000,11,aspect);
    aspect = 1.0;
    drawOval(58,118,25,11,aspect);
    fillOval(-110,100,3,16,aspect);
    fillOval(40,100,3,16,aspect);
    fillOval(-140,60,3,16,aspect);
    fillOval(30,50,3,16,aspect);
    fillOval(30,-40,3,16,aspect);
    fillOval(100,30,3,16,aspect);
    fillOval(-140,25,3,16,aspect);
    write_horz_char(30,-50,90,31);  /* Z */
    write_horz_char(105,25,89,31);  /* Y */
    write_horz_char(10,70,88,31);   /* X */
    write_horz_char(50,110,87,31);  /* W */
```

```
    write_horz_char(-100,115,86,31);   /* V */
    write_horz_char(-160,60,85,31);   /* U */
    write_horz_char(-150,25,82,31);   /* R */
    write_horz_char(-160,90,97,31);   /* a */
    write_horz_char(-120,80,98,31);   /* b */
    write_horz_char(-60,115,99,31);   /* c */
    write_horz_char(80,75,100,31);   /* d */
    write_horz_char(60,35,101,31);   /* e */

    write_horz_char(40,5,102,31);   /* f */
    write_horz_char(-135,48,103,31);   /* g */
    write_horz_char(-45,-18,104,31);   /* h */
    write_horz_char(60,10,105,31);   /* i */
    write_horz_char(160,10,106,31);   /* j */
    write_horz_char(75,150,109,31);   /* m */
    write_horz_char(-55,45,110,31);   /* n */
    write_horz_char(40,75,112,31);   /* p */
    write_horz_str(-300,-65," However, the graph in the following "
                            " Figure is not connected since there is",31);
    write_horz_str(-300,-80," no path from vertex U to vertex W.",31);
    drawLine(-100,-100,100,-165,11);
    drawLine(-100,-130,100,-130,11);
    fillOval(-100,-100,3,16,aspect);
    fillOval(100,-165,3,16,aspect);
    fillOval(-100,-130,3,16,aspect);
    fillOval(100,-130,3,16,aspect);
    write_horz_char(-115,-100,85,11);  /* U */
    write_horz_char(105,-165,84,11);  /* V */
    write_horz_char(-115,-130,86,11);  /* W */
    write_horz_char(105,-130,87,11);  /* X */

    wait("");
}
```

```
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
    line = 29;
    else
    line = 24;
    tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
```

```
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;

    case 'n':write_horz_str(-300,-210,"                    ",31);
              break;

    case 'N':write_horz_str(-300,-210,"                    ",31);
              break;

    default : break;
    }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/


static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_14.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
/*************************************************
                FUNCTION DEFINITIONS
*************************************************/

void wait(char title[]);
void example_3_14  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

/*************************************************
                  MAIN PROGRAM
*************************************************/

main()
{
     cls(1);
     adapt = getAdapter();
     if (adapt == 'V')
     {
         setMode(0x12);
         cls(1);
     }
     if (adapt == 'E')
     {
        setMode(16);
        cls(1);
     }
     if (adapt == 'C')
     {
        setMode(4);
        setCGAPalette(0);
        cls(1);
     }
     example_3_14();
     setMode(3);
}
```

```
/*************************************************************************/
static void example_3_14(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,160," For the multigraph in the following figure"
                            " the edges a, c, p, n form",31);
    write_horz_str(-300,140," a cycle. Likewise, the edges g, b, c, p, f,"
                            " h form a cycle. Furthermore the",31);
    write_horz_str(-300,120," edges f, p, d, e, n, g, h do not form"
                            " a cycle because the vertex X",31);
    write_horz_str(-300,100," is used twice.",31);

    drawLine(-110,-10,40,-10,11);     /* c */
    drawLine(-110,-10,-140,-50,11);   /* b */
    drawLine(-140,-50,30,-60,11);     /* n */
    drawLine(30,-60,40,-10,11);       /* p */
    drawLine(30,-60,30,-150,11);      /* f */
    drawLine(30,-150,100,-80,11);     /* i */
    drawLine(30,-60,100,-80,11);      /* e */
    drawLine(40,-10,100,-80,11);      /* d */
    drawLine(30,-150,-140,-85,11);    /* h */
    drawLine(-140,-85,-140,-50,11);   /* g */

    aspect = .30;
    LINEWIDTH = 1;
    drawArc(65,-115,37,2500,730,11,aspect);
    aspect = 2.0;
    drawArc(-125,-34,38,400,2000,11,aspect);
    aspect = 1.0;
    drawOval(58,8,25,11,aspect);
    fillOval(-110,-10,3,16,aspect);
    fillOval(40,-10,3,16,aspect);
    fillOval(-140,-50,3,16,aspect);
    fillOval(30,-60,3,16,aspect);
    fillOval(30,-150,3,16,aspect);
    fillOval(100,-80,3,16,aspect);
    fillOval(-140,-85,3,16,aspect);
    write_horz_char(30,-160,90,31);  /* Z */
```

```
    write_horz_char(105,-85,89,31);  /* Y */
    write_horz_char(10,-40,88,31);    /* X */
    write_horz_char(50,0,87,31);      /* W */
    write_horz_char(-100,5,86,31);    /* V */
    write_horz_char(-160,-50,85,31);  /* U */
    write_horz_char(-150,-85,82,31);  /* R */
    write_horz_char(-160,-20,97,31);  /* a */
    write_horz_char(-120,-30,98,31);  /* b */
    write_horz_char(-60,5,99,31);     /* c */
    write_horz_char(80,-35,100,31);   /* d */
    write_horz_char(60,-75,101,31);   /* e */

    write_horz_char(40,-105,102,31);   /* f */
    write_horz_char(-135,-62,103,31);  /* g */
    write_horz_char(-45,-128,104,31);  /* h */
    write_horz_char(60,-100,105,31);   /* i */
    write_horz_char(160,-100,106,31);  /* j */
    write_horz_char(75,40,109,31);     /* m */
    write_horz_char(-55,-65,110,31);   /* n */
    write_horz_char(40,-35,112,31);    /* p */
    wait("");
}
/*****************************************************************************/

void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
    line = 29;
    else
    line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
```

```
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)         {
    case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

    case 'n':write_horz_str(-300,-210,"                          ",31);
            break;

    case 'N':write_horz_str(-300,-210,"                          ",31);
            break;

    default : break;
    }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
```

```
}
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_15.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescm,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void example_3_15  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);


        /*************************************************
                          MAIN PROGRAM
        *************************************************/


main()
{
    cls (1);
    adapt = getAdapter();
    if (adapt == 'V')
    {
       setMode(0x12);
       cls(1);
    }
    if (adapt == 'E')
    {
      setMode(16);
      cls(1);
    }
    if (adapt == 'C')
    {
      setMode(4);
      setCGAPalette(0);
      cls(1);
    }
    example_3_15();
    setMode(3);
}
```

```
/*************************************************************************/
static void example_3_15(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,160," For the graph in the Figure (a) the path"
                            " a, b, c, d is an Euler circuit",31);
    write_horz_str(-300,140," since all the edges are included and each"
                            " edge is included exactly once.",31);
    write_horz_str(-300,120," However, the graph in Figure (b) has"
                            " neither an Euler path nor",31);
    write_horz_str(-300,100," circuit because to include all the three"
                            " edges in a path we would",31);
    write_horz_str(-300,80," have to backtrack and use an edge twice."
                           " For the graph in Figure (c)",31);
    write_horz_str(-300,60," there is an Euler path a, b, c, d, e, f"
                           " but not an Euler circuit.",31);
    drawLine(-170,0,-130,0,11);          /* a */
    drawLine(-130,0,-170,-60,11);        /* b */
    drawLine(-170,-60,-130,-60,11);      /* c */
    drawLine(-130,-60,-170,0,11);        /* d */
    aspect = 1.0;
    fillOval(-170,0,3,16,aspect);
    fillOval(-130,0,3,16,aspect);
    fillOval(-170,-60,3,16,aspect);
    fillOval(-130,-60,3,16,aspect);
    fillOval(-170,0,3,16,aspect);

    drawLine(-30,0,-30,-60,11);
    drawLine(-60,0,-30,-60,11);
    drawLine(0,0,-30,-60,11);
    fillOval(-30,0,3,16,aspect);
    fillOval(-30,-60,3,16,aspect);
    fillOval(-60,0,3,16,aspect);
    fillOval(0,0,3,16,aspect);

    drawLine(70,-30,70,-60,11);   /* a */
    drawLine(70,-30,100,-30,11);  /* b */
    drawLine(100,-30,85,10,11);   /* c */
```

```
drawLine(85,10,140,-30,11);   /* d */
drawLine(140,-30,70,-60,11);  /* e */
drawLine(70,-60,100,-30,11);  /* f */
fillOval(70,-30,3,16,aspect);
fillOval(70,-60,3,16,aspect);
fillOval(100,-30,3,16,aspect);
fillOval(85,10,3,16,aspect);
fillOval(140,-30,3,16,aspect);
fillOval(140,-30,3,16,aspect);

write_horz_char(-150,20,97,31);     /* a */
write_horz_char(-170,-30,98,31);    /* b */
write_horz_char(-150,-65,99,31);    /* c */
write_horz_char(-135,-30,100,31);   /* d */

write_horz_char(60,-40,97,31);      /* a */
write_horz_char(75,-15,98,31);      /* b */
write_horz_char(80,-3,99,31);       /* c */
write_horz_char(113,5,100,31);      /* d */
write_horz_char(105,-50,101,31);    /* e */
write_horz_char(75,-35,102,31);     /* f */

write_horz_str(-155,-80,"(a)",31);   /* (a) */
write_horz_str(-30,-80,"(b)",31);    /* (b) */
write_horz_str(105,-80,"(c)",31);    /* (c) */

write_horz_str(-300,-100," As we proceed along an Euler path or"
                         " circuit, each time an intermediate",31);
write_horz_str(-300,-115," vertex is reached along some edge"
                         " there must be another edge for us",31);
write_horz_str(-300,-130," to exit that vertex. In fact, whenever"
                         " each vertex has even degree the",31);
write_horz_str(-300,-145," following algorithm produces an Euler circuit.",31);

wait("");

}
```

```
/*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
     line = 29;
     else
     line = 24;
     tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
      cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
```

```
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;

    case 'n':write_horz_str(-300,-210,"                    ",31);
              break;

    case 'N':write_horz_str(-300,-210,"                    ",31);
              break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow.ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : ex3_16.c
   AUTHOR       : Yavuz BAS
   DATE         : Feb. 4, 1990
   REVISED      : Mar. 5, 1990

   DESCRIPTION  : This program contains an example about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
                GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/

char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
/*************************************************
                FUNCTION DEFINITIONS
*************************************************/

void wait(char title[]);
void example_3_16  (void);
void ex_3_16_cont  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);


/*************************************************
                  MAIN PROGRAM
*************************************************/

main()
{
    cls (1);
    adapt = getAdapter();
    if (adapt == 'V')
    {
       setMode(0x12);
       cls(1);
    }
    if (adapt == 'E')
    {
      setMode(16);
      cls(1);
    }
    if (adapt == 'C')
    {
       setMode(4);
       setCGAPalette(0);
       cls(1);
    }
    example_3_16();
    setMode(3);
}
```

```
/*************************************************************************/
static void example_3_16(void)
{
/*************************************************************************/
  /* attach [Pageup] to the example_3_16 function */
  setonkey(0x4900,example_3_16,0);
/*************************************************************************/
  /* attach [Pagedown] to the ex_3_16_cont function */
  setonkey(0x5100,ex_3_16_cont,0);
/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,180," For the graph in the following figure,"
                            " it may be possible to look at the",31);
    write_horz_str(-300,160," graph and with minimal trial and error"
                            " construct an Euler circuit.",31);
    write_horz_str(-300,140," However for the purpose of illustration"
                            " let us use the Euler circuit",31);
    write_horz_str(-300,120," algorithm.",31);
    drawRect(-40,80,40,40,11);
    drawLine(40,40,30.0,11);  /* g */
    drawLine(40,40,120,0,11); /* e */
    drawLine(30.0,120.0,11);  /* f */

    aspect = 0.5;
    LINEWIDTH = 1;
    drawOval(-10,0,20,11,aspect);
    aspect = 1.0;
    fillOval(-40,80,3,16,aspect);
    fillOval(40,80,3,16,aspect);
    fillOval(40,40,3,16,aspect);
    fillOval(-40,40,3,16,aspect);
    fillOval(30,0,3,16,aspect);
    fillOval(120,0,3,16,aspect);
    fillOval(-50,0,3,16,aspect);
    write_horz_char(-50,45,83,31);  /* S */
    write_horz_char(30,-5,84,31);   /* T */
    write_horz_char(50,50,85,31);   /* U */
    write_horz_char(-50,90,86,31);  /* V */
```

```
    write_horz_char(50,90,87,31);      /* W */
    write_horz_char(130,0,88,31);      /* X */
    write_horz_char(-60,5,89,31);      /* Y */
    write_horz_char(0,95,97,31);       /* a */
    write_horz_char(45,70,98,31);      /* b */
    write_horz_char(0,55,99,31);       /* c */
    write_horz_char(-50,65,100,31);    /* d */
    write_horz_char(75,40,101,31);     /* e */
    write_horz_char(75,-5,102,31);     /* f */
    write_horz_char(40,25,103,31);     /* g */
    write_horz_char(-10,17,104,31);    /* h */
    write_horz_char(-10,-23,105,31);   /* i */
    write_horz_str(-300,-45," First choose some vertex at which to"
                            " begin, say V. Next choose an edge",31);
    write_horz_str(-300,-60," on V, say a. Then the other vertex on"
                            " the edge a is W and W is not",31);
    write_horz_str(-300,-75," equal to V. Since there is only one unused"
                            " edge on W, namely the edge b,",31);
    write_horz_str(-300,-90," we choose b.The other vertex on the"
                            " edge b is the vertex U and U is",31);
    write_horz_str(-300,-105," not equal to V. Since there are three"
                            " unused edges on U, arbitrarily",31);
    write_horz_str(-300,-120," choose an edge, say edge c. Then edge d"
                            " must be chosen next, returning",31);
    write_horz_str(-300,-135," our path to vertex V. These edges a,"
                            " b, c, d do not form an Euler circuit",31);
    write_horz_str(-300,-150," because some of the edges of the graph"
                            " are missing.",31);
    wait("");
    ex_3_16_cont();
}
/*************************************************************************/
static void ex_3_16_cont(void)
{
/*************************************************************************/
  /* attach [Pageup] to the example_3_16 function */
  setonkey(0x4900,example_3_16,0);
/*************************************************************************/
  /* attach [Pagedown] to the normal_exit function */
  setonkey(0x5100,normal_exit,0);
/*************************************************************************/
```

```
LINEWIDTH = 3;
cls(1);
drawRect(-309,210,310,-200,11);
drawRect(-40,160,40,120,11);
drawLine(40,120,30,80,11);          /* g */
drawLine(40,120,120,80,11);         /* e */
drawLine(30,80,120,80,11);          /* f */

aspect = 0.5;
LINEWIDTH = 1;
drawOval(-10,80,20,11,aspect);
aspect = 1.0;
fillOval(-40,160,3,16,aspect);
fillOval(40,160,3,16,aspect);
fillOval(40,120,3,16,aspect);
fillOval(-40,120,3,16,aspect);
fillOval(30,80,3,16,aspect);
fillOval(120,80,3,16,aspect);
fillOval(-50,80,3,16,aspect);

write_horz_char(-50,125,83,31);    /* S */
write_horz_char(30,75,84,31);      /* T */
write_horz_char(50,130,85,31);     /* U */
write_horz_char(-50,170,86,31);    /* V */
write_horz_char(50,170,87,31);     /* W */
write_horz_char(130,80,88,31);     /* X */
write_horz_char(-60,85,89,31);     /* Y */
write_horz_char(0,175,97,31);      /* a */
write_horz_char(45,150,98,31);     /* b */
write_horz_char(0,135,99,31);      /* c */
write_horz_char(-50,145,100,31);   /* d */
write_horz_char(75,120,101,31);    /* e */
write_horz_char(75,75,102,31);     /* f */
write_horz_char(40,105,103,31);    /* g */
write_horz_char(-10,97,104,31);    /* h */
write_horz_char(-10,57,105,31);    /* i */
write_horz_str(-300,37," Since edge e has not been used and is"
                        " on a vertex U which has been visited",31);
write_horz_str(-300,17," we start with U and the edge e. Thus,"
                        " we assign U the temporary name A.",31);
write_horz_str(-300,-3," Since the vertex X is on the edge e and X"
```

```
                                " is not equal to A, we choose",31);
    write_horz_str(-300,-23," the other edge f on X. Next we"
                                " arbitrarily choose the edge g, returning",31);
    write_horz_str(-300,-43," us to A. Now these two groups of edges"
                                " are put together by joining them",31);
    write_horz_str(-300,-63," at the vertex U to form a, b, e, f, g,"
                                " c, d. Note how the edges are",31);
    write_horz_str(-300,-83," put together so that a path with no"
                                " repetitions of edges is formed. Again",31);
    write_horz_str(-300,-103," this is not an Euler circuit. The process"
                                " is now started again with",31);
    write_horz_str(-300,-123," vertex T and edge h. Then the edges h"
                                " , i are obtained and now these are",31);
    write_horz_str(-300,-143," joined into the previously chosen"
                                " edges at vertex T to get a, b, e",31);
    write_horz_str(-300,-163," f, h, i, g, c, d which is an Euler circuit",31);

    wait("");
    normal_exit();
}

/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
        line = 29;
    else
        line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit example, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': normal_exit();
           break;
    case 'Y': normal_exit();
           break;

    case 'n':write_horz_str(-300,-210,"                          ",31);
           break;

    case 'N':write_horz_str(-300,-210,"                          ",31);
           break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   setMode(3);
   videoinit();
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_17.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
                GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
```

```
/*************************************************
                FUNCTION DEFINITIONS
 *************************************************/

void wait(char title[]);
void example_3_17(void);
void confirm_graph_exit(void);
void normal_exit(void);


/*************************************************
                MAIN PROGRAM
 *************************************************/


main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
     {
         setMode(0x12);
         cls(1);

     if (adapt == 'E')
     {
        setMode(16);
        cls(1);
     }
     if (adapt == 'C')
     {
        setMode(4);
        setCGAPalette(0);
        cls(1);
     }
     example_3_17();
     setMode(3);
}
```

```
/*************************************************************************/
static void example_3_17(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,160," For the multigraph in the"
                            " figure (a) an edge d is added between",31);
    write_horz_str(-300,140," the two vertices U and V of odd degree."
                            " This results in the",31);
    write_horz_str(-300,120," multigraph in Figure (b) for which an"
                            " Euler circuit, say e, a, d,",31);
    write_horz_str(-300,100," c, b can be found by using the Euler"
                            " circuit algorithm. Deleting the",31);
    write_horz_str(-300,80," edge e from this circuit gives the Euler"
                            " path a, d, c, b between",31);
    write_horz_str(-300,60," U and V for the multigraph in Figure (a).",31);

    drawLine(-110,20,-10,20,11);   /* a */
    drawLine(-10,20,-60,-70,11);   /* d */
    drawLine(-60,-70,-110,20,11);  /* c */

    drawLine(40,20,140,20,11);     /* a */
    drawLine(40,20,90,-70,11);     /* d */
    drawLine(90,-70,140,20,11);    /* c */

    aspect = .40;
    LINEWIDTH = 1;
    drawArc(-60,20,20,1800,0,11,aspect);
    drawArc(90,20,20,1800,0,11,aspect);
    drawArc(90,20,20,0,1800,7,aspect);
    aspect = 1.0;
    fillOval(-110,20,3,16,aspect);
    fillOval(-10,20,3,16,aspect);
    fillOval(-60,-70,3,16,aspect);
    fillOval(40,20,3,16,aspect);
    fillOval(140,20,3,16,aspect);
    fillOval(90,-70,3,16,aspect);


    write_horz_char(-60,35,97,31); /* a */
```

```
    write_horz_char(-60,-5,98,31);     /* b */
    write_horz_char(-95,-20,99,31);    /* c */
    write_horz_char(-30,-20,100,31);   /* d */

    write_horz_char(-120,20,85,31);    /* U */
    write_horz_char(-5,20,86,31);      /* V */
    write_horz_char(-60,-75,65,31);    /* A */

    write_horz_char(90,35,97,31);      /* a */
    write_horz_char(90,-5,98,31);      /* b */
    write_horz_char(55,-20,99,31);     /* c */
    write_horz_char(120,-20,100,31);   /* d */

    write_horz_char(30,20,85,31);    /* U */
    write_horz_char(145,20,86,31);   /* V */
    write_horz_char(90,-75,65,31);   /* A */
    write_horz_char(90,55,101,31);   /* e */
    write_horz_str(-60,-85,"(a)",11);
    write_horz_str(90,-85,"(b)",11);
    wait("Example 3.17");
}

/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
    line = 29;
    else
    line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    getch();
    cls(0);
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(-80,190," Please type y or n",31);
     ch = getch ();
   }
   switch (ch)        {
    case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

    case 'n':
            break;

    case 'N':
            break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.         */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_18.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Paths and
Circuits".

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*********************************************************
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *********************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
```

```
        /*************************************************
                    FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void example_3_18  (void);
void confirm_graph_exit(void);
void normal_exit(void);


        /*************************************************
                        MAIN PROGRAM
        *************************************************/

main()
{
    cls (1);
    adapt = getAdapter();
    if (adapt == 'V')
    {
        setMode(0x12);
        cls(1);
    }
    if (adapt == 'E')
    {
       setMode(16);
       cls(1);
    }
    if (adapt == 'C')
    {
       setMode(4);
       setCGAPalette(0);
       cls(1);
    }
    example_3_18();
    setMode(3);

}
```

```
/*****************************************************************************/
static void example_3_18 (void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    drawRect(-50,20,50,-80,31);
    drawLine(-50,20,0,-30,11);  /* c */
    drawLine(0,-30,50,-30,31); /* d */
    drawLine(50,-30,50,-80,11);  /* f */
    drawLine(0,-30,50,-80,31);  /* g */
    aspect = 1.0;
    fillOval(-50,20,3,16,aspect);
    fillOval(0,-30,3,16,aspect);
    fillOval(-50,-80,3,16,aspect);
    fillOval(50,-80,3,16,aspect);
    fillOval(50,20,3,16,aspect);
    fillOval(50,-30,3,16,aspect);

    write_horz_char(-60,20,85,31);     /* U */
    write_horz_char(55,20,86,31);      /* V */
    write_horz_char(-60,-85,87,31);    /* W */
    write_horz_char(55,-25,88,31);     /* X */
    write_horz_char(55,-85,89,31);     /* Y */
    write_horz_char(0,-15,90,31);      /* Z */
    write_horz_char(0,40,97,31);       /* a */
    write_horz_char(55,0,98,31);       /* b */
    write_horz_char(-20,5,99,31);      /* c */
    write_horz_char(25,-15,100,31);    /* d */
    write_horz_char(-60,-15,101,31);   /* e */
    write_horz_char(0,-85,102,31);     /* f */
    write_horz_char(5,-50,103,31);     /* g */
    write_horz_char(55,-50,104,31);    /* h */

    write_horz_str(-300,180," Suppose the graph in the following"
                            " Figure describes a system of airline",31);
    write_horz_str(-300,160," routes where the vertices are towns"
                            " and the edges represent airline",31);
    write_horz_str(-300,140," routes. The vertex U is the home base for"
                            " a salesperson who must",31);
    write_horz_str(-300,120," periodically visit all of the"
```

```
                                    " other cities. To be economical the salesperson",31);
    write_horz_str(-300,100," wants a path that starts at U, ends at"
                                    " U, and visits each of the other",31);
    write_horz_str(-300,80," vertices exactly once. A brief examination"
                                    " of the graph shows that the",31);
    write_horz_str(-300,60," edges a, b, d, g, f, e form a"
                                   " Hamiltonian cycle.",31);
    wait("Example 3.18");
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
        line = 29;
    else
        line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    getch();
    cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(-80,190," Please type y or n",31);
```

```
      ch = getch ();
    }
    switch (ch)         {
     case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
     case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

     case 'n':
            break;

     case 'N':
            break;

     default : break;
     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_19.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
```

```
/*************************************************
                FUNCTION DEFINITIONS
*************************************************/

void wait(char title[]);
void example_3_19  (void);
void confirm_graph_exit(void);
void normal_exit(void);



/*************************************************
                  MAIN PROGRAM
*************************************************/

main()
{
     cls (1);
     adapt = getAdapter();
      if (adapt == 'V')
     {
         setMode(0x12);
         cls(1);
     }
     if (adapt == 'E')
     {
         setMode(16);
         cls(1);
     }
     if (adapt == 'C')
     {
        setMode(4);
        setCGAPalette(0);
        cls(1);
     }
     example_3_19();
     setMode(3);
 }
```

```
/*************************************************************************/
static void example_3_19(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,190," Theorem 3.5 can be used to say that the"
                            " graph in Fig.(a) has a Hamiltonian",31);
    write_horz_str(-300,175," cycle because there are 6 vertices, each"
                            " with degree 3. However even",31);
    write_horz_str(-300,160," though the theorem says there is a"
                            " Hamiltonian cycle, it does not tell",31);
    write_horz_str(-300,145," us how to find one. Fortunately in this case"
                            " one can be found by a little",31);
    write_horz_str(-300,130," bit of trial and error.",31);
    write_horz_str(-300,115," On the other hand the graph in Figure (b)"
                            " does not have a Hamiltonian cycle",31);
    write_horz_str(-300,100," because no matter where we start we"
                            " end up on the left side needing to go",31);
    write_horz_str(-300,85," to a vertex on the left side, and there are"
                            " no edges connecting the vertices",31);
    write_horz_str(-300,70," on that side. Note that this graph does"
                            " not satisfy the conditions",31);
    write_horz_str(-300,55," of theorem 3.5 because it has vertices"
                            " of degree 2<(5/2). However the graph",31);
    write_horz_str(-300,40," in Fig.(c) also has 5 vertices, each with"
                           " degree 2; Yet it contains a",31);
    write_horz_str(-300,25," Hamiltonian cycle. Thus, when some vertices"
                            " have degree less than n/2,",31);
    write_horz_str(-300,10," it is not possible to conclude anything in"
                           " general about the existence or",31);
    write_horz_str(-300,-5," nonexistence of a Hamiltonian cycle.",31);


    drawLine(-200,-45,-100,-45,31);
    drawLine(-200,-45,-100,-95,11);
    drawLine(-200,-45,-100,-145,31);

    drawLine(-100,-45,-200,-95,31);
    drawLine(-100,-45,-200,-145,11);

    drawLine(-200,-95,-100,-95,31);
    drawLine(-200,-95,-100,-145,11);
```

```
drawLine(-100,-95,-200,-145,31);
drawLine(-200,-145,-100,-145,31);
write_horz_str(-150,-165,"(a)",31);
aspect = 1.0;
fillOval(-200,-45,3,16,aspect);
fillOval(-100,-45,3,16,aspect);
fillOval(-100,-95,3,16,aspect);
fillOval(-100,-145,3,16,aspect);
fillOval(-200,-145,3,16,aspect);
fillOval(-200,-95,3,16,aspect);

drawLine(-50,-45,50,-70,11);
drawLine(-50,-45,50,-120,11);

drawLine(-50,-95,50,-70,11);
drawLine(-50,-95,50,-120,11);

drawLine(-50,-145,50,-120,11);
drawLine(-50,-145,50,-70,11);
fillOval(-50,-45,3,16,aspect);
fillOval(50,-70,3,16,aspect);
fillOval(50,-120,3,16,aspect);
fillOval(-50,-95,3,16,aspect);
fillOval(-50,-145,3,16,aspect);
write_horz_str(0,-165,"(b)",31);

drawLine(100,-95,150,-45,11);
drawLine(150,-45,200,-95,11);
drawLine(200,-95,200,-145,11);
drawLine(200,-145,100,-145,11);
drawLine(100,-145,100,-95,11);
fillOval(100,-95,3,16,aspect);
fillOval(150,-45,3,16,aspect);
fillOval(200,-95,3,16,aspect);
fillOval(200,-145,3,16,aspect);
fillOval(100,-145,3,16,aspect);
write_horz_str(145,-165,"(c)",31);
wait("Example 3.19");
}
```

```
/*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
     line = 29;
     else
     line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     getch();
     cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(-80,190," Please type y or n",31);
      ch = getch ();
    }
    switch (ch)        {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'Y': setMode(3);
```

```
            videoinit();
            normal_exit();
            break;

    case 'n':
            break;

    case 'N':
            break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/

static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : ex3_39.c
  AUTHOR     : Yavuz BAS
  DATE       : Feb. 4, 1990
  REVISED    : Mar. 5, 1990

  DESCRIPTION : This program contains an example about the "Paths and
Circuits".

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*********************************************************
            GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *********************************************************/

char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
         *************************************************/
void wait(char title[]);
static void example_3_39 (void);
static void cont1 (void);
static void cont2 (void);
static void cont3 (void);
static void cont4 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
        /*************************************************
                        MAIN PROGRAM
         *************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
       {
          setMode(0x12);
          cls(1);
       }
      if (adapt == 'E')
       {
          setMode(16);
          cls(1);
       }
     if (adapt == 'C')
       {
          setMode(4);
          setCGAPalette(0);
          cls(1);}
   example_3_39();
   setMode(3);
}
```

```
/*************************************************************************/
static void example_3_39(void)
{

/*************************************************************************/
  /* attach [Pageup] to the example_3_39 function */
  setonkey(0x4900,example_3_39,0);

/*************************************************************************/
  /* attach [Pagedown] to the cont1 function */
  setonkey(0x5100,cont1,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine(-100,-80,-150,-110,11);     /* AS */
    drawLine(-150,-110,-100,-140,11);    /* SB */
    drawLine(-100,-140,-50,-110,11);     /* BC */
    drawLine(-50,-110,-100,-80,11);      /* CA */
    drawLine(-100,-80,0,-80,11);         /* AF */
    drawLine(-50,-110,-50,-80,11);       /* CE */
    drawLine(0,-80,0,-110,11);           /* FH */
    drawLine(-50,-110,50,-110,11);       /* CT */
    drawLine(50,-110,80,-90,11);         /* TI */
    drawLine(50,-110,50,-140,11);        /* TJ */
    drawLine(50,-140,0,-160,11);         /* JG */
    drawLine(0,-160,-50,-160,11);        /* GD */
    drawLine(-50,-160,-100,-140,11);     /* DB */

    aspect = 1.0;
    fillOval(-100,-80,3,16,aspect);
    fillOval(-150,-110,3,16,aspect);
    fillOval(-100,-140,3,16,aspect);
    fillOval(-50,-110,3,16,aspect);
```

```
fillOval(80,-90,3,16,aspect);
fillOval(0,-80,3,16,aspect);
fillOval(-50,-80,3,16,aspect);
fillOval(0,-110,3,16,aspect);
fillOval(50,-110,3,16,aspect);
fillOval(50,-140,3,16,aspect);
fillOval(0,-160,3,16,aspect);
fillOval(-50,-160,3,16,aspect);


write_horz_char(-100,-65,65,31);    /* A */
write_horz_char(-100,-145,66,31);   /* B */
write_horz_char(-45,-95,67,31);     /* C */
write_horz_char(-50,-165,68,31);    /* D */
write_horz_char(-50,-65,69,31);     /* E */
write_horz_char(0,-65,70,31);       /* F */
write_horz_char(0,-165,71,31);      /* G */
write_horz_char(5,-95,72,31);       /* H */
write_horz_char(80,-70,73,31);      /* I */
write_horz_char(55,-145,74,31);     /* J */
write_horz_char(-160,-110,83,31);   /* S */
write_horz_char(55,-110,84,31);     /* T */

write_horz_str(-300,180,"For the graph in the following Figure"
                        " there is more than one path from",31);
write_horz_str(-300,160,"S to T. One path is S, A, C, B, D, G, J,"
                        " T and another is S, A, E, F, H, T.",31);
write_horz_str(-300,140,"We will use the breadth-first search"
                        " algorithm to find the distance",31);
write_horz_str(-300,120,"and a path of minimal length between"
                        " S and T.",31);
wait("");
cont1();

}
```

```
/*************************************************************************/
static void cont1(void)
{

/*************************************************************************/
  /* attach [Pageup] to the example_3_39 function */
  setonkey(0x4900,example_3_39,0);

/*************************************************************************/
  /* attach [Pagedown] to the cont2 function */
  setonkey(0x5100,cont2,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine(-100,-80,-150,-110,11);    /* AS */
    drawLine(-150,-110,-100,-140,11);   /* SB */
    drawLine(-100,-140,-50,-110,11);    /* BC */
    drawLine(-50,-110,-100,-80,11);     /* CA */
    drawLine(-100,-80,-00,-80,11);      /* AF */
    drawLine(-50,-110,-50,-80,11);      /* CE */
    drawLine(0,-80,0,-110,11);          /* FH */
    drawLine(-50,-110,50,-110,11);      /* CT */
    drawLine(50,-110,80,-90,11);        /* TI */
    drawLine(50,-110,50,-140,11);       /* TJ */
    drawLine(50,-140,0,-160,11);        /* JG */
    drawLine(0,-160,-50,-160,11);       /* GD */
    drawLine(-50,-160,-100,-140,11);    /* DB */

    aspect = 1.0;
    fillOval(-100,-80,3,16,aspect);
    fillOval(-150,-110,3,16,aspect);
    fillOval(-100,-140,3,16,aspect);
    fillOval(-50,-110,3,16,aspect);
```

```
fillOval(80,-90,3,16,aspect);
fillOval(0,-80,3,16,aspect);
fillOval(-50,-80,3,16,aspect);
fillOval(0,-110,3,16,aspect);
fillOval(50,-110,3,16,aspect);
fillOval(50,-140,3,16,aspect);
fillOval(0,-160,3,16,aspect);
fillOval(-50,-160,3,16,aspect);

write_horz_char(-100,-65,65,31);    /* A */
write_horz_char(-100,-145,66,31);   /* B */
write_horz_char(-45,-95,67,31);     /* C */
write_horz_char(-50,-165,68,31);    /* D */
write_horz_char(-50,-65,69,31);     /* E */
write_horz_char(0,-65,70,31);       /* F */
write_horz_char(0,-165,71,31);      /* G */
write_horz_char(5,-95,72,31);       /* H */
write_horz_char(80,-70,73,31);      /* I */
write_horz_char(55,-145,74,31);     /* J */
write_horz_char(-160,-110,83,31);   /* S */
write_horz_char(55,-110,84,31);     /* T */

write_horz_str(-300,180,"First label S with 0, set L = {S}, and"
                        " let S have no predecessor. Since",31);
write_horz_str(-300,160,"T is not in L, we look at the vertices"
                        " adjacent to S, which are A and B.",31);
write_horz_str(-300,140,"To them the label 1 is assigned, and"
                        " L then becomes {S, A, B}. A and B",31);
write_horz_str(-300,120,"are given the predecessor S.",31);
write_horz_str(-105,-53,"1(S)",28);
write_horz_str(-105,-157,"1(S)",28);
write_horz_str(-180,-98,"0(-)",28);

wait("");
cont2();
}
```

```
/*************************************************************************/
static void cont2(void)
{

/*************************************************************************/
  /* attach [Pageup] to the cont1 function */
  setonkey(0x4900,cont1,0);

/*************************************************************************/
  /* attach [Pagedown] to the cont3 function */
  setonkey(0x5100,cont3,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine(-100,-80,-150,-110,11);    /* AS */
    drawLine(-150,-110,-100,-140,11);   /* SB */
    drawLine(-100,-140,-50,-110,11);    /* BC */
    drawLine(-50,-110,-100,-80,11);     /* CA */
    drawLine(-100,-80,-00,-80,11);      /* AF */
    drawLine(-50,-110,-50,-80,11);      /* CE */
    drawLine(0,-80,0,-110,11);          /* FH */
    drawLine(-50,-110,50,-110,11);      /* CT */
    drawLine(50,-110,80,-90,11);        /* TI */
    drawLine(50,-110,50,-140,11);       /* TJ */
    drawLine(50,-140,0,-160,11);        /* JG */
    drawLine(0,-160,-50,-160,11);       /* GD */
    drawLine(-50,-160,-100,-140,11);    /* DB */

    aspect = 1.0;
    fillOval(-100,-80,3,16,aspect);
    fillOval(-150,-110,3,16,aspect);
    fillOval(-100,-140,3,16,aspect);
    fillOval(-50,-110,3,16,aspect);
```

```
fillOval(80,-90,3,16,aspect);
fillOval(0,-80,3,16,aspect);
fillOval(-50,-80,3,16,aspect);
fillOval(0,-110,3,16,aspect);
fillOval(50,-110,3,16,aspect);
fillOval(50,-140,3,16.aspect);
fillOval(0,-160,3,16,aspect);
fillOval(-50,-160,3,16,aspect);

write_horz_char(-100,-65,65,31);    /* A */
write_horz_char(-100,-145,66,31);   /* B */
write_horz_char(-45,-95,67,31);     /* C */
write_horz_char(-50,-165,68,31);    /* D */
write_horz_char(-50,-65,69,31);     /* E */
write_horz_char(0,-65,70,31);       /* F */
write_horz_char(0,-165,71,31);      /* G */
write_horz_char(5,-95,72,31);       /* H */
write_horz_char(80,-70,73,31);      /* I */
write_horz_char(55,-145,74,31);     /* J */
write_horz_char(-160,-110,83,31);   /* S */
write_horz_char(55,-110,84,31);     /* T */

write_horz_str(-300,180,"Again since T is not in L, we find the"
                        " unlabeled vertices adjacent to",31);
write_horz_str(-300,160,"vertices with label 1. These are the"
                        " vertices E, C, and D, which are",31);
write_horz_str(-300,140,"then assigned the label 2. Now L = {"
                        "S, A, B, E, C, D} Since E and C are",31);
write_horz_str(-300,120,"adjacent to A which has label 1, E and"
                        " C each have predecessor A, and",31);
write_horz_str(-300,100,"similarly, D has predecessor B.",31);

write_horz_str(-45,-83,"2(A)",28);
write_horz_str(-55,-177,"2(B)",28);
write_horz_str(-55,-53,"2(A)",28);
```

```
    write_horz_str(-105,-53,"1(S)",28);
    write_horz_str(-105,-157,"1(S)",28);
    write_horz_str(-180,-98,"0(-)",28);

    wait("");
    cont3();

}
/*****************************************************************************/
static void cont3(void)
{
/*****************************************************************************/
  /* attach [Pageup] to the cont2 function */
  setonkey(0x4900,cont2,0);

/*****************************************************************************/
  /* attach [Pagedown] to the cont4 function */
  setonkey(0x5100,cont4,0);

/*****************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    drawLine(-100,-80,-150,-110,11);    /* AS */
    drawLine(-150,-110,-100,-140,11);   /* SB */
    drawLine(-100,-140,-50,-110,11);    /* BC */
    drawLine(-50,-110,-100,-80,11);     /* CA */
    drawLine(-100,-80,-00,-80,11);      /* AF */
    drawLine(-50,-110,-50,-80,11);      /* CE */
    drawLine(0,-80,0,-110,11);          /* FH */
    drawLine(-50,-110,50,-110,11);      /* CT */
    drawLine(50,-110,80,-90,11);        /* TI */
    drawLine(50,-110,50,-140,11);       /* TJ */
    drawLine(50,-140,0,-160,11);        /* JG */
    drawLine(0,-160,-50,-160,11);       /* GD */
    drawLine(-50,-160,-100,-140,11);    /* DB */
```

```
aspect = 1.0;
fillOval(-100,-80,3,16,aspect);
fillOval(-150,-110,3,16,aspect);
fillOval(-100,-140,3,16,aspect);
fillOval(-50,-110,3,16,aspect);
fillOval(80,-90,3,16,aspect);
fillOval(0,-80,3,16,aspect);
fillOval(-50,-80,3,16,aspect);
fillOval(0,-110,3,16,aspect);
fillOval(50,-110,3,16,aspect);
fillOval(50,-140,3,16,aspect);
fillOval(0,-160,3,16,aspect);
fillOval(-50,-160,3,16,aspect);

write_horz_char(-100,-65,65,31);    /* A */
write_horz_char(-100,-145,66,31);   /* B */
write_horz_char(-45,-95,67,31);     /* C */
write_horz_char(-50,-165,68,31);    /* D */
write_horz_char(-50,-65,69,31);     /* E */
write_horz_char(0,-65,70,31);       /* F */
write_horz_char(0,-165,71,31);      /* G */
write_horz_char(5,-95,72,31);       /* H */
write_horz_char(80,-70,73,31);      /* I */
write_horz_char(55,-145,74,31);     /* J */
write_horz_char(-160,-110,83,31);   /* S */
write_horz_char(55,-110,84,31);     /* T */

write_horz_str(-300,180,"Again because T is not in L, the"
                        " unlabeled vertices adjacent to those",31);
write_horz_str(-300,160,"with label 2 are determined. These"
                        " are F, H, and G, and they are",31);
write_horz_str(-300,140,"assigned the label 3 and included in L"
                        ". So L is now {S, A, B, E, C, D, F,",31);
write_horz_str(-300,120,"G, H}. Because F is adjacent to E with"
                        " label 2, F has predecessor E.",31);
write_horz_str(-300,100,"Similarly, H has predecessor C, and G"
```

```
        " has predecessor D.",31);
    write_horz_str(-45,-83,"2(A)",28);
    write_horz_str(-55,-177,"2(B)",28);
    write_horz_str(-55,-53,"2(A)",28);

    write_horz_str(-105,-53,"1(S)",28);
    write_horz_str(-105,-157,"1(S)",28);
    write_horz_str(-180,-98,"0(-)",28);

    write_horz_str(-5,-53,"3(E)",28);
    write_horz_str(-5,-177,"3(D)",28);
    write_horz_str(5,-83,"3(C)",28);

    wait("");
    cont4();
}
/*************************************************************************/
static void cont4(void)
{

/*************************************************************************/
  /* attach [Pageup] to the cont3 function */
  setonkey(0x4900,cont3,0);

/*************************************************************************/
  /* attach [Pagedown] to the normal_exit function */
  setonkey(0x5100,normal_exit,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    drawLine(-100,-80,-150,-110,11);    /* AS */
    drawLine(-150,-110,-100,-140,11);   /* SB */
    drawLine(-100,-140,-50,-110,11);    /* BC */
    drawLine(-50,-110,-100,-80,11);     /* CA */
```

```
drawLine(-100,-80,-00,-80,11);     /* AF */
drawLine(-50,-110,-50,-80,11);     /* CE */
drawLine(0,-80,0,-110,11);          /* FH */
drawLine(-50,-110,50,-110,11);     /* CT */
drawLine(50,-110,80,-90,11);        /* TI */
drawLine(50,-110,50,-140,11);       /* TJ */
drawLine(50,-140,0,-160,11);        /* JG */
drawLine(0,-160,-50,-160,11);       /* GD */
drawLine(-50,-160,-100,-140,11); /* DB */

aspect = 1.0;
fillOval(-100,-80,3,16,aspect);
fillOval(-150,-110,3,16,aspect);
fillOval(-100,-140,3,16,aspect);
fillOval(-50,-110,3,16,aspect);
fillOval(80,-90,3,16,aspect);
fillOval(0,-80,3,16,aspect);
fillOval(-50,-80,3,16,aspect);
fillOval(0,-110,3,16,aspect);
fillOval(50,-110,3,16,aspect);
fillOval(50,-140,3,16,aspect);
fillOval(0,-160,3,16,aspect);
fillOval(-50,-160,3,16,aspect);

write_horz_char(-100,-65,65,31);   /* A */
write_horz_char(-100,-145,66,31); /* B */
write_horz_char(-45,-95,67,31);     /* C */
write_horz_char(-50,-165,68,31);   /* D */
write_horz_char(-50,-65,69,31);     /* E */
write_horz_char(0,-65,70,31);        /* F */
write_horz_char(0,-165,71,31);      /* G */
write_horz_char(5,-95,72,31);        /* H */
write_horz_char(80,-70,73,31);      /* I */
write_horz_char(55,-145,74,31);     /* J */
write_horz_char(-160,-110,83,31); /* S */
write_horz_char(55,-110,84,31);     /* T */
```

```
    write_horz_str(-300,180,"Again T is not in L, and so the"
                            " unlabeled vertices adjacent to those with",31);
    write_horz_str(-300,160,"label 3 are located. These"
                            " are J and T, which are then assigned the",31);
    write_horz_str(-300,140,"label 4. Finally, the"
                            " predecessor of T is H and that of J is G.",31);
    write_horz_str(-300,120,"  Since the label 4 has been assigned"
                            " to T, the distance from S to T is 4.",31);
    write_horz_str(-300,100,"By looking at the predecessors, we find"
                            " that S, A, C, H, T is a shortest",31);
    write_horz_str(-300,80,"path from S to T.",31);

    write_horz_str(-45,-83,"2(A)",28);
    write_horz_str(-55,-177,"2(B)",28);
    write_horz_str(-55,-53,"2(A)",28);

    write_horz_str(-105,-53,"1(S)",28);
    write_horz_str(-105,-157,"1(S)",28);
    write_horz_str(-180,-98,"0(-)",28);

    write_horz_str(-5,-53,"3(E)",28);
    write_horz_str(-5,-177,"3(D)",28);
    write_horz_str(5,-83,"3(C)",28);

    write_horz_str(50,-157,"4(G)",28);
    write_horz_str(55,-122,"4(H)",28);

    drawLine(-100,-80,-150,-110,21);    /* AS */
    drawLine(-50,-110,-100,-80,21);     /* CA */
    drawLine(-50,-110,50,-110,21);      /* CT */

    wait("");
    normal_exit();

}
```

```
/*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
       if ((mode == 0x11) || (mode == 0x12))
          line = 29;
       else
          line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
      cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       write_horz_str(30,-210," Please type y or n",79);
       ch = getch ();
       write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': normal_exit();
             break;
     case 'Y': normal_exit();
```

```
            break;

     case 'n':write_horz_str(-300,-210,"                    ",31);
            break;

     case 'N':write_horz_str(-300,-210,"                    ",31);
            break;

     default : break;
     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.     */
/*************************************************************************/
static void normal_exit(void)
{
   setMode(3);
   videoinit();
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_40.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/

char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
```

```
        /*************************************************
                FUNCTION DEFINITIONS
         *************************************************/

void wait(char title[]);
static void example_3_40 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                MAIN PROGRAM
         *************************************************/

main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
        {
          setMode(0x12);
          cls(1);
        }
    if (adapt == 'E')
        {
          setMode(16);
          cls(1);
        }
   if (adapt == 'C')
        {
          setMode(4);
          setCGAPalette(0);
          cls(1);
        }
   example_3_40();
   setMode(3);
}
```

```
/*************************************************************************/
static void example_3_40(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    drawLine(-50,0,-100,-50,11);      /* BA */
    drawLine(-100,-50,-50,-100,11);  /* AC */
    drawLine(-50,-100,50,-100,11);   /* CF */
    drawLine(50,-100,100,-50,11);    /* FG */
    drawLine(100,-50,50,0,11);       /* GE */
    drawLine(50,0,-50,0,11);         /* EB */
    drawLine(-50,0,50,-100,11);      /* BF */
    drawLine(50,0,-50,-100,11);      /* EC */
    drawLine(50,0,50,-100,11);       /* EF */

    aspect = 1.0;
    fillOval(-50,0,3,16,aspect);
    fillOval(-100,-50,3,16,aspect);
    fillOval(-50,-100,3,16,aspect);
    fillOval(50,-100,3,16,aspect);
    fillOval(100,-50,3,16,aspect);
    fillOval(50,0,3,16,aspect);
    fillOval(0,-50,3,16,aspect);

    write_horz_char(-50,17,66,31);    /* B */
    write_horz_char(-110,-50,65,31); /* A */
    write_horz_char(-50,-112,67,31); /* C */
    write_horz_char(50,-112,70,31);  /* F */
    write_horz_char(105,-50,71,31);  /* G */
    write_horz_char(50,17,69,31);     /* E */
    write_horz_char(-2,-30,68,31);    /* D */
    write_horz_char(-90 -15,51,31);  /* 3 */
    write_horz_char(-90,-75,52,31);  /* 4 */
    write_horz_char(0,-83,54,31);     /* 6 */
    write_horz_char(80,-80,50,31);    /* 2 */
```

```
    write_horz_char(80,-10,52,31);   /* 4 */
    write_horz_char(0,17,50,31);      /* 2 */
    write_horz_char(-35,-25,49,31); /* 1 */
    write_horz_char(25,-30,49,31);   /* 1 */
    write_horz_char(-35,-60,50,31); /* 2 */
    write_horz_char(30,-60,53,31);   /* 5 */
    write_horz_char(55,-50,50,31);   /* 2 */
    write_horz_str(-300,180,"The graph in the following figure is"
                            " a weighted graph since each edge",31);
    write_horz_str(-300,160,"has a number assigned to it. For example"
                            " the weight of the edge on A and",31);
    write_horz_str(-300,140,"B is 3 and the weight of the edge on"
                            " D and F is 5. The weight of the",31);
    write_horz_str(-300,120,"path A, C, D, F is 4 + 2 + 5 = 11,"
                            " and the weight of the path F, D, B,",31);
    write_horz_str(-300,100,"E, D is 5 + 1 + 2 + 1 = 9.",31);
    wait(" Example 3.40 ");
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
      line = 29;
    else
      line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    getch();
    cls(0);
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(-80,190," Please type y or n",31);
     ch = getch ();
   }
   switch (ch)        {
    case 'y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'Y': setMode(3);
           videoinit();
           normal_exit();
           break;

    case 'n':
           break;

    case 'N':
           break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/* PROGRAM  : ex3_41.c
  AUTHOR     : Yavuz BAS
  DATE       : Feb. 4, 1990
  REVISED    : Mar. 5, 1990

  DESCRIPTION : This program contains an example about the "Paths and
Circuits".

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
         *************************************************/

void wait(char title[]);
static void example_3_41 (void);
static void confirm_graph_exit (void);
static void normal_exit(void);
        /*************************************************
                          MAIN PROGRAM
         *************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
       {
          setMode(0x12);
          cls(1);
       }
     if (adapt == 'E')
       {
          setMode(16);
          cls(1);
       }
     if (adapt == 'C')
       {
          setMode(4);
          setCGAPalette(0);
          cls(1);
       }
   example_3_41();
   setMode(3);

}
```

```
/*****************************************************************/
static void example_3_41(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine(-100,-50,0,-50,31);    /* AB */
    drawLine(0,-50,50,-20,11);      /* BC */
    drawLine(0,-50,50,-80,11);      /* BD */
    drawLine(50,-20,50,-80,11);     /* CD */
    drawLine(50,-80,100,-50,31);    /* DE */
    aspect = 1.0;
    fillOval(-100,-50,3,16,aspect);
    fillOval(0,-50,3,16,aspect);
    fillOval(50,-20,3,16,aspect);
    fillOval(50,-80,3,16,aspect);
    fillOval(100,-50,3,16,aspect);

    write_horz_char(-110,-50,65,31); /* A */
    write_horz_char(0,-58,66,31);    /* B */
    write_horz_char(50,-5,67,31);    /* C */
    write_horz_char(50,-85,68,31);   /* D */
    write_horz_char(105,-50,69,31);  /* E */

    write_horz_char(-50,-35,51,31); /* 3  */
    write_horz_str(15,-15,"-2",31);  /* -2 */
    write_horz_str(15,-70,"-3",31); /* -3 */
    write_horz_char(40,-45,49,31);   /* 1 */
    write_horz_char(65,-50,50,31);  /* 2 */

    write_horz_str(-300,180,"For the weighted graph in the following"
                            " figure the path A, B, D, E has",31);
    write_horz_str(-300,160,"weight 2, and the path A, B, D, C, B, D,"
                            " E has weight -2, which is a",31);
    write_horz_str(-300,140,"smaller weight than that for the first"
```

```
                                    " path. Note that as the cycle",31);
        write_horz_str(-300,120,"B, D, C is repeated, the weight of"
                                    " the path gets smaller and smaller.",31);
        write_horz_str(-300,100,"Thus, there is no path of smallest weight"
                                    " between A and E.",31);
        wait(" Example 3.41 ");
}
/*************************************************************************/
void wait(char title[])
{
        int tab,width,line,mode;
        mode = getMode(&width);
        if ((mode == 0x11) || (mode == 0x12))
           line = 29;
        else
           line = 24;
        tab = (width - strlen(title))/2;
        gotoxy(tab,0);
        writString(title,WHITE,0);
        tab = (width - 33)/2;
        gotoxy(tab,line);
        writString("Press any key to continue ...", WHITE,0);
        getch();
        cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
         write_horz_str(-80,190," Please type y or n",31);
```

```
        ch = getch ();
    }
    switch (ch)         {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'Y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'n':
               break;
     case 'N':
               break;
     default : break;
     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.        */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_42.c
  AUTHOR     : Yavuz BAS
  DATE       : Feb. 4, 1990
  REVISED    : Mar. 5, 1990

  DESCRIPTION : This program contains an example about the "Paths and
Circuits".

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /************************************************
                    FUNCTION DEFINITIONS
        ************************************************/
void wait(char title[]);
static void example_3_42 (void);
static void fig_3_47 (void);
static void fig_3_48 (void);
static void fig_3_49 (void);
static void fig_3_50 (void);
static void fig_3_51 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
        /************************************************
                        MAIN PROGRAM
        ************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
       {
          setMode(0x12);
          cls(1);
       }
     if (adapt == 'E')
       {
          setMode(16);
          cls(1);          }
    if (adapt == 'C')
       {
          setMode(4);
          setCGAPalette(0);
         cls(1);}
   example_3_42();
   setMode(3);
}
```

```
/*************************************************************************/
static void example_3_42(void)
{
/*************************************************************************/
  /* attach [Pageup] to the example_3_42 function */
  setonkey(0x4900,example_3_42,0);

/*************************************************** ***********************/
  /* attach [Pagedown] to the fig_3_47 function */
  setonkey(0x5100,fig_3_47,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine(-50,0,-100,-50,11);      /* AS */
    drawLine(-100,-50,-50,-100,11);   /* SB */
    drawLine(-50,-100,50,-100,11);    /* BD */
    drawLine(50,-100,100,-50,11);     /* DE */
    drawLine(100,-50,50,0,11);        /* EC */
    drawLine(50,0,-50,0,11);          /* CA */
    drawLine(-50,0,-50,-100,11);      /* AB */
    drawLine(50,0,-50,-100,11);       /* CB */
    drawLine(50,0,50,-100,11);        /* CD */

    aspect = 1.0;
    fillOval(-50.0,3,16,aspect);
    fillOval(-100,-50,3,16,aspect);
    fillOval(-50,-100,3,16,aspect);
    fillOval(50,-100,3,16,aspect);
    fillOval(100,-50,3,16,aspect);
    fillOval(50.0,3,16,aspect);

    write_horz_char(-50,17,65,31);    /* A */
    write_horz_char(-110,-50,83,31);  /* S */
```

```
    write_horz_char(-50,-112,66,31); /* B */
    write_horz_char(50,-112,68,31); /* D */
    write_horz_char(105,-50,69,31); /* E */
    write_horz_char(50,17,67,31);    /* C */
    write_horz_char(-90,-15,51,31); /* 3 */
    write_horz_char(-90,-75,49,31); /* 1 */
    write_horz_char(0,-83,53,31);    /* 5 */
    write_horz_char(80,-80,49,31);   /* 1 */
    write_horz_char(80,-10,51,31);   /* 3 */
    write_horz_char(0,17,50,31);     /* 2 */
    write_horz_char(-63,-50,49,31); /* 1 */
    write_horz_char(-5,-30,51,31);   /* 3 */
    write_horz_char(55,-50,49,31);   /* 1 */
    write_horz_str(-300,180,"for the weighted graph in the following"
                            " figure we want to find a",31);
    write_horz_str(-300,160,"shortest path and the distance from S to"
                            " every other vertex.",31);
    wait("");
    fig_3_47();
}
/************ ********************************************************************/
static void fig_3_47(void)
{

/*************************************************************************/
  /* attach [Pageup] to the example_3_42 function */
  setonkey(0x4900,example_3_42,0);

/*************************************************************************/
  /* attach [Pagedown] to the fig_3_48 function */
  setonkey(0x5100,fig_3_48,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
```

```
drawLine(-50,0,-100,-50,11);      /* A3 */
drawLine(-100,-50,-50,-100,11);  /* SB */
drawLine(-50,-100,50,-100,11);    /* BD */
drawLine(50,-100,100,-50,11);     /* DE */
drawLine(100,-50,50,0,11);        /* EC */
drawLine(50,0,-50,0,11);          /* CA */
drawLine(-50,0,-50,-100,11);      /* AB */
drawLine(50,0,-50,-100,11);       /* CB */
drawLine(50,0,50,-100,11);        /* CD */

aspect = 1.0;
fillOval(-50,0,3,16,aspect);
fillOval(-100,-50,3,16,aspect);
fillOval(-50,-100,3,16,aspect);
fillOval(50,-100,3,16,aspect);
fillOval(100,-50,3,16,aspect);
fillOval(50,0,3,16,aspect);


write_horz_char(-50,17,65,31); /* A */
write_horz_str(-125,-50,"S*",31); /* S */
write_horz_str(-135,-65,"0(-)",27);
write_horz_char(-50,-112,66,31); /* B */
write_horz_char(50,-112,68,31); /* D */
write_horz_char(105,-50,69,31); /* E */
write_horz_char(50,17,67,31); /* C */
write_horz_char(-90,-15,51,31); /* 3 */
write_horz_char(-90,-75,49,31); /* 1 */
write_horz_char(0,-83,53,31); /* 5 */
write_horz_char(80,-80,49,31); /* 1 */
write_horz_char(80,-10,51,31); /* 3 */
write_horz_char(0,17,50,31); /* 2 */
write_horz_char(-63,-50,49,31); /* 1 */
write_horz_char(-5,-30,51,31); /* 3 */
write_horz_char(55,-50,49,31); /* 1 */
```

```
    write_horz_str(-300,180," According to step 1, we start by putting"
                             " S in P and assigning",31);
    write_horz_str(-300,160," to S the label 0 and no predecessor. We"
                             " indicate this on the",31);
    write_horz_str(-300,140," graph by writing the label and"
                             " predecessor (in parenthesis) by S. We",31);
    write_horz_str(-300,120," use an asterisk to show that S is in P."
                             " The graph now looks ",31);
    write_horz_str(-300,100," like the following:",31);
    wait("");
    fig_3_48();
}
/*************************************************************************/
static void fig_3_48(void)
{

/*************************************************************************/
  /* attach [Pageup] to the fig_3_47 function */
  setonkey(0x4900,fig_3_47,0);

/*************************************************************************/
  /* attach [Pagedown] to the fig_3_49 function */
  setonkey(0x5100,fig_3_49,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine(-50,0,-100,-50,11);      /* AS */
    drawLine(-100,-50,-50,-100,11);   /* SB */
    drawLine(-50,-100,50,-100,11);    /* BD */
    drawLine(50,-100,100,-50,11);     /* DE */
    drawLine(100,-50,50,0,11);        /* EC */
    drawLine(50,0,-50,0,11);          /* CA */
    drawLine(-50,0,-50,-100,11);      /* AB */
```

```
drawLine(50,0,-50,-100,11);      /* CB */
drawLine(50,0,50,-100,11);       /* CD */

aspect = 1.0;
fillOval(-50,0,3,16,aspect);
fillOval(-100,-50,3,16,aspect);
fillOval(-50,-100,3,16,aspect);
fillOval(50,-100,3,16,aspect);
fillOval(100,-50,3,16,aspect);
fillOval(50,0,3,16,aspect);

write_horz_char(-50,17,65,31);          /* A */
write_horz_str(-65,30,"3(S)",27);
write_horz_str(-125,-50,"S*",31);       /* S */
write_horz_str(-135,-65,"0(-)",27);
write_horz_char(-50,-112,66,31);        /* B */
write_horz_str(-65,-124,"1(S)",27);
write_horz_char(50,-112,68,31);         /* D */
write_horz_str(35,-124,"inf(S)",27);
write_horz_char(105,-50,69,31);         /* E */
write_horz_str(100,-62,"inf(S)",27);
write_horz_char(50,17,67,31);           /* C */
write_horz_str(35,30,"inf(S)",27);
write_horz_char(-90,-15,51,31);         /* 3 */
write_horz_char(-90,-75,49,31);         /* 1 */
write_horz_char(0,-83,53,31);           /* 5 */
write_horz_char(80,-80,49,31);          /* 1 */
write_horz_char(80,-10,51,31);          /* 3 */
write_horz_char(0,17,50,31);            /* 2 */
write_horz_char(-63,-50,49,31);         /* 1 */
write_horz_char(-5,-30,51,31);          /* 3 */
write_horz_char(55,-50,49,31);          /* 1 */

write_horz_str(-300,180," Next, according to step 2, we assign the"
                        " label W(S,V) and the predecessor",31);
write_horz_str(-300,160," S to the other vertices."
```

```
                                " Recall that W(S,V) = inf when there is no edge",31);
    write_horz_str(-300,140," between S and V. The graph now looks"
                                " like the following:",31);
    wait("");
    fig_3_49();

}
/*************************************************************************/
static void fig_3_49(void)
{

/*************************************************************************/
  /* attach [Pageup] to the fig_3_48 function */
  setonkey(0x4900,fig_3_48,0);

/*************************************************************************/
  /* attach [Pagedown] to the fig_3_50 function */
  setonkey(0x5100,fig_3_50,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine(-50,-60,-100,-110,11);     /* AS */
    drawLine(-100,-110,-50,-160,11);    /* SB */
    drawLine(-50,-160,50,-160,11);      /* BD */
    drawLine(50,-160,100,-110,11);      /* DE */
    drawLine(100,-110,50,-60,11);       /* EC */
    drawLine(50,-60,-50,-60,11);        /* CA */
    drawLine(-50,-60,-50,-160,11);      /* AB */
    drawLine(50,-60,-50,-160,11);       /* CB */
    drawLine(50,-60,50,-160,11);        /* CD */

    aspect = 1.0;
    fillOval(-50,-60,3,16,aspect);
```

```
fillOval(-100,-110,3,16,aspect);
fillOval(-50,-160,3,16,aspect);
fillOval(50,-160,3,16,aspect);
fillOval(100,-110,3,16,aspect);
fillOval(50,-60,3,16,aspect);

write_horz_char(-50,-43,65,31);                 /* A */
write_horz_str(-65,-30,"2(B)",27);
write_horz_str(-125,-110,"S*",31);              /* S */
write_horz_str(-135,-125,"0(-)",27);
write_horz_str(-50,-172,"B*",31);               /* B */
write_horz_str(-65,-184,"1(S)",27);
write_horz_char(50,-172,68,31);                 /* D */
write_horz_str(35,-184,"6(B)",27);
write_horz_char(105,-110,69,31);                /* E */
write_horz_str(100,-122,"inf(S)",27);
write_horz_char(50,-43,67,31);                  /* C */
write_horz_str(35,-30,"4(B)",27);
write_horz_char(-90,-75,51,31);                 /* 3 */
write_horz_char(-90,-135,49,31);                /* 1 */
write_horz_char(0,-143,53,31);                  /* 5 */
write_horz_char(80,-140,49,31);                 /* 1 */
write_horz_char(80,-70,51,31);                  /* 3 */
write_horz_char(0,-43,50,31);                   /* 2 */
write_horz_char(-63,-110,49,31);                /* 1 */
write_horz_char(-5,-90,51,31);                  /* 3 */
write_horz_char(55,-110,49,31);                 /* 1 */

write_horz_str(-300,200,"Now we apply step 3. The vertex not in P"
                        " with the smallest label is B, and",31);
write_horz_str(-300,180,"so we add it to P. The vertices not in P"
                        " and adjacent to B are A, C, and D;",31);
write_horz_str(-300,160,"and we replace the label on each such"
                        " vertex X by the minimum of the old",31);
write_horz_str(-300,140,"label and (label on B) + W(B,X). These"
                        " numbers are as follows.",31);
```

```
    drawLine(-260,120,230,120,27);
    write_horz_str(-250,110,"Vertex X  Old label  (Label on B) +"
                            " W(B,X)  Minimum",31);
    drawLine(-260,95,230,95,27);
    write_horz_str(-250,85, "  A       3          1 + 1 = 2"
                            "          2 ",31);
    write_horz_str(-250,70, "  C       inf        1 + 3 = 4"
                            "          4 ",31);
    write_horz_str(-250,50, "  D       inf        1 + 5 = 6"
                            "          6 ",31);
    write_horz_str(-300,25, "Since each label is changed, we also"
                            " replace the predecessor of",31);
    write_horz_str(-300,5, "each of these vertices by B, producing"
                           " the following figure.",31);

    wait("");
    fig_3_50();

}

/*************************************************************************/
static void fig_3_50(void)
{

/*************************************************************************/
  /* attach [Pageup] to the fig_3_49 function */
  setonkey(0x4900,fig_3_49,0);

/*************************************************************************/
  /* attach [Pagedown] to the fig_3_51 function */
  setonkey(0x5100,fig_3_51,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
```

```
    drawLine(-300,120,300,120,27);
    write_horz_str(-300,110,"              Labels and predec"
                            "essors",31);
    write_horz_str(-300,95, " Vertex   S   A   B     C     "
                            " D      E   Vertex added to P",31);
    drawLine(-300,80,300,80,27);
    write_horz_str(-300,65, "       0(-) 3(S) 1(S)   inf(S)   inf(S)"
                            " inf(s)       S",31);
    write_horz_str(-300,50, "           2(B)       4(B)    6(B) "
                            "              B",31);
    write_horz_str(-300,35, "                                  "
                            "              A",31);
    write_horz_str(-300,20, "                                5(C) "
                            " 7(C)         C",31);
    write_horz_str(-300,5, "                                    "
                           " 6(D)         D",31);
    write_horz_str(-300,-10, "                                   "
                            "              E",31);
    write_horz_str(-300,180, "We continue in this way. The figure"
                            " below shows the labels, predecessors",31);
    write_horz_str(-300,160, "and vertices added to P at each stage"
                            ". No entry in a column indicates",31);
    write_horz_str(-300,140, "no change from the previous stage.",31);
    wait("");
    fig_3_51();
}
```

```
/*************************************************************************/
static void fig_3_51(void)
{

/*************************************************************************/
  /* attach [Pageup] to the fig_3_50 function */
  setonkey(0x4900,fig_3_50,0);

/*************************************************************************/
  /* attach [Pagedown] to the normal_exit function */
  setonkey(0x5100,normal_exit,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine(-50,0,-100,-50,11);      /* AS */
    drawLine(-100,-50,-50,-100,11);   /* SB */
    drawLine(-50,-100,50,-100,11);    /* BD */
    drawLine(50,-100,100,-50,11);     /* DE */
    drawLine(100,-50,50,0,11);        /* EC */
    drawLine(50,0,-50,0,11);          /* CA */
    drawLine(-50,0,-50,-100,11);      /* AB */
    drawLine(50,0,-50,-100,11);       /* CB */
    drawLine(50,0,50,-100,11);        /* CD */

    aspect = 1.0;
    fillOval(-50,0,3,16,aspect);
    fillOval(-100,-50,3,16,aspect);
    fillOval(-50,-100,3,16,aspect);
    fillOval(50,-100,3,16,aspect);
    fillOval(100,-50,3,16,aspect);
    fillOval(50,0,3,16,aspect);

    write_horz_str(-50,17,"A*",31); /* A */
```

```
write_horz_str(-65,30,"2(B)",27);
write_horz_str(-125,-50,"S*",31);                    /* S */
write_horz_str(-135,-65,"0(-)",27);
write_horz_str(-50,-112,"B*",31);                    /* B */
write_horz_str(-65,-124,"1(S)",27);
write_horz_str(50.-112,"D*",31);                     /* D */
write_horz_str(35,-124,"5(C)",27);
write_horz_str(105,-50,"E*",31);                     /* E */
write_horz_str(100,-62,"6(D)",27);
write_horz_str(50,17,"C*",31);                       /* C */
write_horz_str(35,30,"4(B)",27);
write_horz_char(-90,-15,51,31);                      /* 3 */
write_horz_char(-90,-75,49,31);                      /* 1 */
write_horz_char(0,-83,53,31);                        /* 5 */
write_horz_char(80,-80,49,31);                       /* 1 */
write_horz_char(80,-10,51,31);                       /* 3 */
write_horz_char(0,17,50,31);                         /* 2 */
write_horz_char(-63,-50,49,31);                      /* 1 */
write_horz_char(-5,-30,51,31);                       /* 3 */
write_horz_char(55,-50,49,31);                       /* 1 */

write_horz_str(-300,180," The final graph is shown in the following"
                        " figure. In this figure the",31);
write_horz_str(-300,160," label on each vertex gives the distance"
                        " between it and S, and a path",31);
write_horz_str(-300,140," of this length can be found by"
                        " backtracking through the predecessors",31);
write_horz_str(-300,120," of the vertices. For example the"
                        " distance from S to E is 6, and the",31);
write_horz_str(-300,100," path S, B, C, D, E has this length.",31);

wait("");
normal_exit();

}
```

```
/*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
      line = 29;
      else
      line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': normal_exit();
          break;
    case 'Y': normal_exit();
```

```
            break;

      case 'n':write_horz_str(-300,-210,"                    ",31);
            break;

      case 'N':write_horz_str(-300,-210,"                    ",31);
            break;

      default : break;
      }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   setMode(3);
   videoinit();
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_30.c
  AUTHOR     : Yavuz BAS
  DATE       : Feb. 4, 1990
  REVISED    : Mar. 5, 1990

  DESCRIPTION : This program contains an example about the "Paths and
Circuits".

  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
            GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/
char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
```

```
        /************************************************
                        FUNCTION DEFINITIONS
         ************************************************/
void wait(char title[]);
static void example_3_30 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
        /************************************************
                          MAIN PROGRAM
         ************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
      if (adapt == 'V')
        {
          setMode(0x12);
          cls(1);
        }
      if (adapt == 'E')
        {
          setMode(16);
          cls(1);
        }
      if (adapt == 'C')
        {
          setMode(4);
          setCGAPalette(0);
          cls(1);
        }
   example_3_30();
   setMode(3);

}
```

```
/*************************************************************************/
static void example_3_30(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawRect(50,0,150,-100,11);
    drawLine (150,0,50,-100,11);
    LINEWIDTH = 1;
    aspect = 1.0;
    drawArc(110,-40,73,3030,1470,11,aspect);
    LINEWIDTH = 3;
    drawRect(-50,0,-150,-100,11);
    drawLine (-150,0,-50,-100,11);
    drawLine (-50,0,-150,-100,11);
    fillOval(50,0,3,16,aspect);
    fillOval(50,-100,3,16,aspect);
    fillOval(150,-100,3,16,aspect);
    fillOval(150,0,3,16,aspect);
    fillOval(-50,0,3,16,aspect);
    fillOval(-50,-100,3,16,aspect);
    fillOval(-150,-100,3,16,aspect);
    fillOval(-150,0,3,16,aspect);
    write_horz_str(-300,180,"We have a great great deal of leeway"
                            " in drawing graphs. For instance,",31);
    write_horz_str(-300,160,"we readily accept the fact that"
                            " the two graphs in the following figure",31);
    write_horz_str(-300,140,"are the same. The only difference"
                            " between the graphs is the way the line",31);
    write_horz_str(-300,120,"representing one of the edges is"
                            " drawn. The structure of a graph is captured",31);
    write_horz_str(-300,100,"in the set of vertices and the set of"
                            " edges.",31);
    wait(" Isomorphisms ");
}
```

```
/*****************  ***********************************  **********************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
        line = 29;
     else
        line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ... ", WHITE,0);
     getch();
     cls(0);
}
/*************************************************************  **********/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
  ch = getch (),
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(-80,190," Please type y or n",31);
     ch = getch ();
   }
   switch (ch)        {
    case 'y': setMode(3);
          videoinit();
          normal_exit(),
          break;
```

```
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;

    case 'n':
              break;

    case 'N':
              break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}


/*****************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.        */
/*****************************************************************/
static void normal_exit(void)
{
  srestore(savescm);
  gotoxy_(crow,ccol);
  if(_mouse) msHidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : ex3_31.c
   AUTHOR       : Yavuz BAS
   DATE         : Feb. 4, 1990
   REVISED      : Mar. 5, 1990

   DESCRIPTION  : This program contains an example about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
                GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/

char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
         *************************************************/
void wait(char title[]);
static void example_3_31 (void);
static void ex_3_31cont1 (void);
static void ex_3_31cont2 (void);
static void ex_3_31cont3 (void);
static void ex_3_31cont4 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
        /*************************************************
                          MAIN PROGRAM
         *************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
       {
          setMode(0x12);
          cls(1);
       }
     if (adapt == 'E')
       {
          setMode(16);
          cls(1);
       }
    if (adapt == 'C')
      {
          setMode(4);
          setCGAPalette(0);
          cls(1);}
   example_3_31();
   setMode(3);
}
```

```
/*************************************************************************/
static void example_3_31(void)
{
/*************************************************************************/
  /* attach [Pageup] to the example_3_31 function */
  setonkey(0x4900,example_3_31,0);

/*************************************************************************/
  /* attach [Pagedown] to the ex_3_31cont1 function */
  setonkey(0x5100,ex_3_31cont1,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine (-100,-100,-120,-170,11);   /* AD */
    drawLine (-120,-170,-70,-130,11);    /* DC */
    drawLine (-70,-130,-130,-130,11);    /* CB */
    drawLine (-130,-130,-80,-170,11);    /* BE */
    drawLine (-80,-170,-100,-100,11);    /* EA */
    aspect = 1.0;
    fillOval(-100,-100,3,16,aspect);
    fillOval(-120,-170,3,16,aspect);
    fillOval(-70,-130,3,16,aspect);
    fillOval(-130,-130,3,16,aspect);
    fillOval(-80,-170,3,16,aspect);

    write_horz_char(-100,-80,65,31);   /* A */
    write_horz_char(-145,-130,66,31);  /* B */
    write_horz_char(-65,-130,67,31);   /* C */
    write_horz_char(-135,-170,68,31);  /* D */
    write_horz_char(-75,-170,69,31);   /* E */

    write_horz_char(100,-80,80,31);    /* P */
    write_horz_char(55,-130,81,31);    /* Q */
```

```
    write_horz_char(135,-130,82,31); /* R */
    write_horz_char(65,-170,83,31);  /* S */
    write_horz_char(125,-170,84,31); /* T */

    drawLine (100,-100,70,-130,11);  /* PQ */
    drawLine (70,-130,80,-170,11);   /* QS */
    drawLine (80,-170,120,-170,11);  /* ST */
    drawLine (120,-170,130,-130,11); /* TR */
    drawLine (130,-130,100,-100,11); /* RP */

    fillOval(100,-100,3,16,aspect);
    fillOval(80,-170,3,16,aspect);
    fillOval(130,-130,3,16,aspect);
    fillOval(70,-130,3,16,aspect);
    fillOval(120,-170,3,16,aspect);

    write_horz_str(-300,180,"Now we will try to show that the graphs"
                            " in the following figure are",31);
    write_horz_str(-300,160,"isomorphic. First, we have to"
                            " find a bijection f from the set {a, b, c, d, e}",31);
    write_horz_str(-300,140,"to the set {p, q, r, s, t}"
                            " that preserves edges. We will describe",31);
    write_horz_str(-300,120,"the function as a table of the form",31);
    write_horz_str(-100,85,"V    A  B  C  D  E",31);
    write_horz_str(-100,55,"f(V)",31);

    drawLine(-100,65,110,65,11);
    drawLine(-100,95,110,95,11);
    drawLine(-100,35,110,35,11);

    write_horz_str(-300,10,"where the entries in the second row"
                           " will give the image of each of the",31);
    write_horz_str(-300,-10,"elements in the first row.",31);
    wait("Isomorphic graphs");
    ex_3_31cont1();
}
```

```
/*****************************************************************************/
static void ex_3_31cont1(void)
{

/*****************************************************************************/
  /* attach [Pageup] to the example_3_31 function */
  setonkey(0x4900,example_3_31,0);

/*****************************************************************************/
  /* attach [Pagedown] to the ex_3_31cont2 function */
  setonkey(0x5100,ex_3_31cont2,0);

/*****************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine (-100,-100,-120,-170,11);   /* AD */
    drawLine (-120,-170,-70,-130,11);    /* DC */
    drawLine (-70,-130,-130,-130,11);    /* CB */
    drawLine (-130,-130,-80,-170,11);    /* BE */
    drawLine (-80,-170,-100,-100,11);    /* EA */

    aspect = 1.0;
    fillOval(-100,-100,3,16,aspect);
    fillOval(-120,-170,3,16,aspect);
    fillOval(-70,-130,3,16,aspect);
    fillOval(-130,-130,3,16,aspect);
    fillOval(-80,-170,3,16,aspect);

    write_horz_char(-100,-80,65,31);   /* A */
    write_horz_char(-145,-130,66,31);  /* B */
    write_horz_char(-65,-130,67,31);   /* C */
    write_horz_char(-135,-170,68,31);  /* D */
    write_horz_char(-75,-170,69,31);   /* E */
```

```
write_horz_char(100,-80,80,31);      /* P */
write_horz_char(55,-130,81,31);  /* Q */
write_horz_char(135,-130,82,31); /* R */
write_horz_char(65,-170,83,31);  /* S */
write_horz_char(125,-170,84,31); /* T */

drawLine (100,-100,70,-130,11);  /* PQ */
drawLine (70,-130,80,-170,11);   /* QS */
drawLine (80,-170,120,-170,11);  /* ST */
drawLine (120,-170,130,-130,11); /* TR */
drawLine (130,-130,100,-100,11); /* RP */

fillOval(100,-100,3,16,aspect);
fillOval(80,-170,3,16,aspect);
fillOval(130,-130,3,16,aspect);
fillOval(70,-130,3,16,aspect);
fillOval(120,-170,3,16,aspect);

write_horz_str(-300,180,"We start by arbitrarily assigning f(A)"
                        " = P. Now A is adjacent to D",31);
write_horz_str(-300,160,"and E, so P must be adjacent to"
                        " f(D) and f(E). This requires that",31);
write_horz_str(-300,140,"{f(D),f(E)}={Q,R} since Q and R are the"
                        " vertices adjacent to P. Arbitrarily,",31);
write_horz_str(-300,120,"we select f(D) = R and f(E) = Q. Then"
                        " our table, so far, looks like",31);
write_horz_str(-100,65,"V     A  B  C  D  E",31);
write_horz_str(-100,35,"f(V)  P        R  Q",31);
drawLine(-100,45,110,45,11);
drawLine(-100,75,110,75,11);
drawLine(-100,15,110,15,11);

wait("Isomorphic graphs ");
ex_3_31cont2();
}
```

```
/*************************************************************************/

static void ex_3_31cont2(void)
{

/*************************************************************************/
  /* attach [Pageup] to the ex_3_31cont1 function */
  setonkey(0x4900,ex_3_31cont1,0);

/*************************************************************************/
  /* attach [Pagedown] to the ex_3_31cont3 function */
  setonkey(0x5100,ex_3_31cont3,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine (-100,-100,-120,-170,11);   /* AD */
    drawLine (-120,-170,-70,-130,11);    /* DC */
    drawLine (-70,-130,-130,-130,11);    /* CB */
    drawLine (-130,-130,-80,-170,11);    /* BE */
    drawLine (-80,-170,-100,-100,11);    /* EA */

    aspect = 1.0;
    fillOval(-100,-100,3,16,aspect);
    fillOval(-120,-170,3,16,aspect);
    fillOval(-70,-130,3,16,aspect);
    fillOval(-130,-130,3,16,aspect);
    fillOval(-80,-170,3,16,aspect);

    write_horz_char(-100,-80,65,31);  /* A */
    write_horz_char(-145,-130,66,31);  /* B */
    write_horz_char(-65,-130,67,31);  /* C */
    write_horz_char(-135,-170,68,31); /* D */
    write_horz_char(-75,-170,69,31);  /* E */
```

```
    write_horz_char(100,-80,80,31);   /* P */
    write_horz_char(55,-130,81,31);   /* Q */
    write_horz_char(135,-130,82,31);  /* R */
    write_horz_char(65,-170,83,31);   /* S */
    write_horz_char(125,-170,84,31);  /* T */

    drawLine (100,-100,70,-130,11);  /* PQ */
    drawLine (70,-130,80,-170,11);   /* QS */
    drawLine (80,-170,120,-170,11);  /* ST */
    drawLine (120,-170,130,-130,11); /* TR */
    drawLine (130,-130,100,-100,11); /* RP */
    fillOval(100,-100,3,16,aspect);
    fillOval(80,-170,3,16,aspect);
    fillOval(130,-130,3,16,aspect);
    fillOval(70,-130,3,16,aspect);
    fillOval(120,-170,3,16,aspect);

    write_horz_str(-300,180,"Our remaining task is to map the set"
                            " {B,C} onto the set {S,T}.",31);
    write_horz_str(-300,160,"At this stage, our choices are limited."
                            " Observe that B is adjacent to C",31);
    write_horz_str(-300,140,"and E. Since f(E) = Q, we must have f(B)"
                            " adjacent to Q. Vertices P and S",31);
    write_horz_str(-300,120,"are adjacent to Q, but we already have f(A)"
                            " = P, so we are forced to take",31);
    write_horz_str(-300,100,"f(B) = S to be sure that f is a bijection"
                            ". Then we have to let f(C) equal T.",31);
    write_horz_str(-300,80,"The complete table is",31);
    write_horz_str(-100,45,"V    A  B  C  D  E",31);
    write_horz_str(-100,15,"f(V)  P  S  T  R  Q",31);
    drawLine(-100,25,110,25,11);
    drawLine(-100,55,110,55,11);
    drawLine(-100,-5,110,-5,11);
    wait(" Isomorphic graphs ");
    ex_3_31cont3();
}
```

```
/*************************************************************************/
static void ex_3_31cont3(void)
{

/*************************************************************************/
  /* attach [Pageup] to the ex_3_31cont2 function */
  setonkey(0x4900,ex_3_31cont2,0);

/*************************************************************************/
  /* attach [Pagedown] to the normal_exit function */
  setonkey(0x5100,normal_exit,0);

/*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine (-100,-100,-120,-170,11);   /* AD */
    drawLine (-120,-170,-70,-130,11);    /* DC */
    drawLine (-70,-130,-130,-130,11);    /* CB */
    drawLine (-130,-130,-80,-170,11);    /* BE */
    drawLine (-80,-170,-100,-100,11);    /* EA */

    aspect = 1.0;
    fillOval(-100,-100,3,16,aspect);
    fillOval(-120,-170,3,16,aspect);
    fillOval(-70,-130,3,16,aspect);
    fillOval(-130,-130,3,16,aspect);
    fillOval(-80,-170,3,16,aspect);

    write_horz_char(-100,-80,65,31);   /* A */
    write_horz_char(-145,-130,66,31);  /* B */
    write_horz_char(-65,-130,67,31);   /* C */
    write_horz_char(-135,-170,68,31);  /* D */
    write_horz_char(-75,-170,69,31);   /* E */
```

```
write_horz_char(100,-80,80,31);   /* P */
write_horz_char(55,-130,81,31);   /* Q */
write_horz_char(135,-130,82,31);  /* R */
write_horz_char(65,-170,83,31);   /* S */
write_horz_char(125,-170,84,31);  /* T */

drawLine (100,-100,70,-130,11);   /* PQ */
drawLine (70,-130,80,-170,11);    /* QS */
drawLine (80,-170,120,-170,11);   /* ST */
drawLine (120,-170,130,-130,11);  /* TR */
drawLine (130,-130,100,-100,11);  /* RP */

fillOval(100,-100,3,16,aspect);
fillOval(80,-170,3,16,aspect);
fillOval(130,-130,3,16,aspect);
fillOval(70,-130,3,16,aspect);
fillOval(120,-170,3,16,aspect);

write_horz_str(-300,180,"While we used some of the adjacency"
                        " information to construct",31);
write_horz_str(-300,160,"f, we have not checked all the conditions"
                        " necessary for isomorphism.",31);
write_horz_str(-300,140,"First, f is certainly a bijection. Next"
                        " we need to check that the mapping",31);
write_horz_str(-300,120,"{U,V} -> {f(U),f(V)} is a bijection between"
                        " the sets of edges. We can",31);
write_horz_str(-300,100,"check this by inspection in this case,"
                        " since there are only five edges",31);
write_horz_str(-300,80,"in each graph. The following table shows"
                       " the correspondence between edges.",31);
write_horz_str(-100,45,"{U,V}      {f(U),f(V)}",31);
write_horz_str(-100,15,"{A,D}       {P,R}",31);
write_horz_str(-100,0,"{A,E}       {P,Q}",31);
write_horz_str(-100,-15,"{B,C}       {S,T}",31);
write_horz_str(-100,-30,"{B,E}       {S,Q}",31);
write_horz_str(-100,-45,"{C,D}       {T,R}",31);
```

```
    drawLine(-110,25,110,25,11);
    drawLine(-110,55,110,55,11);
    drawLine(-110,-65,110,-65,11);
    drawLine(-110,55,-110,-65,11);
    drawLine(110,55,110,-65,11);
    wait(" Isomorphic graphs ");
    normal_exit();
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
       line = 29;
    else
       line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
```

```
      write_horz_str(30,-210," Please type y or n",79);
       ch = getch ();
       write_horz_str(30,-210,"                ",31);
     }
     switch (ch)        {
     case 'y': normal_exit();
            break;
     case 'Y': normal_exit();
            break;

     case 'n':write_horz_str(-300,-210,"                        ",31);
            break;

     case 'N':write_horz_str(-300,-210,"                        ",31);
            break;

     default : break;
     }
     if(_mouse&MS_CURS) msshowcur();
     chgonkey(kblist);     /* restore any hidden hot keys */

}
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
   setMode(3);
   videoinit();
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : ex3_32.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an example about the "Paths and
Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/

char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
```

```
/*************************************************
            FUNCTION DEFINITIONS
 *************************************************/

void wait(char title[]);
static void example_3_32 (void);
st itic void confirm_graph_exit(void);
static void normal_exit(void);

/*************************************************
                MAIN PROGRAM
 *************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
      if (adapt == 'V')
        {
          setMode(0x12);
          cls(1);
        }
      if (adapt == 'E')
        {
          setMode(16);
          cls(1);
        }
      if (adapt == 'C')
        {
          setMode(4);
          setCGAPalette(0);
          cls(1);
        }
    example_3_32();
    setMode(3);

}
```

```
/*************************************************************************/
static void example_3_32(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    drawRect(-150,100,-50,0,11);
    drawLine (-150,100,-100,150,11); /* BC */
    drawLine (-100,150,-50,100,11);  /* CD */

    aspect = 1.0;
    fillOval(-150,100,3,16,aspect);
    fillOval(-150,0,3,16,aspect);
    fillOval(-50,0,3,16,aspect);
    fillOval(-50,100,3,16,aspect);
    fillOval(-100,150,3,16,aspect);

    drawLine (100,150,50,100,11);   /* QP */
    drawLine (50,100,110,0,11);     /* PT */
    drawLine (110,0,150,100,11);    /* TS */
    drawLine (150,100,100,150,11);  /* SQ */
    drawLine (100,150,200,140,11);  /* QR */
    drawLine (200,140,110,0,11);    /* RT */

    fillOval(100,150,3,16,aspect);
    fillOval(50,100,3,16,aspect);
    fillOval(110,0,3,16,aspect);
    fillOval(150,100,3,16,aspect);
    fillOval(200,140,3,16,aspect);

    write_horz_char (-165,100,66,31); /* B */
    write_horz_char (-165,0,65,31);  /* A */
    write_horz_char (-45,0,69,31);   /* E */
    write_horz_char (-45,100,68,31); /* D */
    write_horz_char (-100,165,67,31); /* C */
```

```
write_horz_char (100,165,81,31);  /* Q */
write_horz_char (35,100,80,31);   /* P */
write_horz_char (115,-5,84,31);   /* T */
write_horz_char (155,100,83,31);  /* S */
write_horz_char (200,155,82,31);  /* R */

write_horz_char (-100,-20,71,31);  /* G */
write_horz_char (-93,-25,49,31);   /* 1 */
write_horz_char (110,-20,71,31);   /* G */
write_horz_char (117,-25,50,31);   /* 2 */

write_horz_str(-300,190,"Let's show that the graphs G1 and G2"
                        " are not isomorphic.",31);
write_horz_str(-300,-45,"We cannot easily show this by inspecting"
                        " the graphs because each of the",31);
write_horz_str(-300,-65,"graphs is connected and has five vertices"
                        " and six edges. However, before we",31);
write_horz_str(-300,-85,"start checking the 120 bijections of {A,B,"
                        "C,D,E} onto {P,Q,R,S,T}, let's",31);
write_horz_str(-300,-105,"look at the way the graphs are put"
                         " together. G1 has a cycle B, C, D, B of",31);
write_horz_str(-300,-125,"length 3. An isomorphism from G1 to G2"
                         " must map this cycle, vertex by vertex,",31);
write_horz_str(-300,-145,"onto a cycle in G2 of the same length. But"
                         " G2 has no cycle of length 3, so",31);
write_horz_str(-300,-165,"there can be no isomorphism of G1 onto"
                         " G2.",31);
wait(" Not isomorphic graphs ");

}
```

```
/*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
         line = 29;
      else
         line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      getch();
      cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(-80,190," Please type y or n",31);
      ch = getch ();
    }
    switch (ch)        {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
```

```
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;

    case 'n':
              break;

    case 'N':
              break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.     */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : pq1.c
   AUTHOR       : Yavuz BAS
   DATE         : Feb. 4, 1990
   REVISED      : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo      C
compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************/
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/



char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescm,crow,ccol;
```

```
        /************************************************
                    FUNCTION DEFINITIONS
        ************************************************/

void wait(char title[]);
void question_1  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /************************************************
                    MAIN PROGRAM
        ************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
       if (adapt == 'V')
         {
           setMode(0x12);
           cls(1);
         }
       if (adapt == 'E')
         {
           setMode(16);
           cls(1);
         }
      if (adapt == 'C')
         {
           setMode(4);
           setCGAPalette(0);
           cls(1);
         }
   question_1();
   setMode(3);
}
```

```
/*****************************************************************************/
static void question_1(void)
{
   LINEWIDTH = 3;
   cls(1);
   drawRect(-309,210,310,-200,11);
   write_horz_str(-300,160," Determine if the graph in the following"
                           " Figure is a graph.",31);
  aspect = 1.0;
  fillOval(-50,100,3,16,aspect);
  fillOval(50,100,3,16,aspect);
  fillOval(0,60,3,16,aspect);
  write_horz_str(-300,30," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,30," Please type y or n",31);
     Ch = getch ();
     if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
     write_horz_str(0,30,"                    ",31);
  }
 switch (Ch)        {
  case 'y' :write_horz_str(-30,30," y ",31);
        write_horz_str(-30,10," Yes, it is a graph that consists",31);
        write_horz_str(-30,-10," of three distinct vertices but",31);
        write_horz_str(-30,-30," no edges.",31);
        break;
  case 'Y' :write_horz_str(-30,30," Y ",31);
        write_horz_str(-30,10," Yes, it is a graph that consists",31);
        write_horz_str(-30,-10," of three distinct vertices but",31);
        write_horz_str(-30,-30," no edges.",31);
        break;
  case 'n' :write_horz_str(-30,30," n ",31);
        write_horz_str(-30,10," Yes, it is a graph that consists",31);
        write_horz_str(-30,-10," of three distinct vertices but",31);
        write_horz_str(-30,-30," no edges.",31);
```

```
        break;
  case 'N' :write_horz_str(-30,30," N ",31);
        write_horz_str(-30,10," Yes, it is a graph that consists",31);
        write_horz_str(-30,-10," of three distinct vertices but",31);
        write_horz_str(-30,-30," no edges.",31);
        break;
  default : break;   }
  wait("Exercise 1");
}
/*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
      line = 29;
    else
      line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
```

```
    while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
    write_horz_str(30,-210," Please type y or n",79);
    ch = getch ();
    write_horz_str(30,-210,"                ",31);
  }
  switch (ch)       {
   case 'y': setMode(3);
         videoinit();
         normal_exit();
         break;
   case 'Y': setMode(3);
         videoinit();
         normal_exit();
         break;
   case 'n':write_horz_str(-300,-210,"                        ",31);
         break;
   case 'N':write_horz_str(-300,-210,"                        ",31);
         break;
   default : break;
   }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescm);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : pq2.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*********************************************************
                GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *********************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescm,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void question_2  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                            MAIN PROGRAM
        *************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
         {
            setMode(0x12);
            cls(1);
          }
       if (adapt == 'E')
         {
            setMode(16);
            cls(1);
          }
      if (adapt == 'C')
         {
            setMode(4);
            setCGAPalette(0);
            cls(1);
          }
   question_2();
   setMode(3);
}
```

```
/*************************************************************************/
static void question_2(void)
{
   LINEWIDTH = 3;
   cls(1);
   drawRect(-309,210,310,-200,11);
   write_horz_str(-300,160," Determine if the multigraph in the following"
                           " Figure is a graph.",31);
  LINEWIDTH = 1;
  aspect = 0.5;
  drawOval(0,120,20,11,aspect);
  drawOval(0,0,20,11,aspect);
  LINEWIDTH = 3;
  drawLine(40,120,40,0,11);

  aspect = 1.0;
  fillOval(-40,120,3,16,aspect);
  fillOval(-40,0,3,16,aspect);
  fillOval(40,120,3,16,aspect);
  fillOval(40,0,3,16,aspect);

  write_horz_str(-300,-30," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,-30," Please type y or n",31);
     Ch = getch ();
     if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
     write_horz_str(0,-30,"                    ",31);
  }

 switch (Ch)        {
  case 'y' :write_horz_str(-30,-30," y ",31);
        write_horz_str(-30,-50," No, it is not a graph because",31);
        write_horz_str(-30,-70," there are parallel edges.",31);
        break;
```

```
  case 'Y' :write_horz_str(-30,-30," Y ",31);
        write_horz_str(-30,-50," No, it is not a graph because",31);
        write_horz_str(-30,-70," there are parallel edges.",31);
        break;
  case 'n' :write_horz_str(-30,-30," n ",31);
        write_horz_str(-30,-50," No, it is not a graph because",31);
        write_horz_str(-30,-70," there are parallel edges.",31);
        break;
  case 'N' :write_hor   str(-30,-30," N ",31);
        write_horz_str(-30,-50," No, it is not a graph because",31);
        write_horz_str(-30,-70," there are parallel edges.",31);
        break;
  default : break;   }
  wait("Exercise 2");
}



/*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
       line = 29;
     else
       line = 24;
      tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
    write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;
    case 'n':write_horz_str(-300,-210,"                          ",31);
            break;
    case 'N':write_horz_str(-300,-210,"                          ",31);
            break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
```

```
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : pq3.c
   AUTHOR       : Yavuz BAS
   DATE         : Feb. 4, 1990
   REVISED      : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/



char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/
void wait(char title[]);
void question_3  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                          MAIN PROGRAM
        *************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
           setMode(0x12);
          cls(1);
        }
     if (adapt == 'E')
       {
           setMode(16);
           cls(1);
       }
     if (adapt == 'C')
      {
           setMode(4);
           setCGAPalette(0);
           cls(1);
       }
   question_3();
   setMode(3);
}
```

```
/*****************************************************************************/
static void question_3(void)
{
   LINEWIDTH = 3;
   cls(1);
   drawRect(-309,210,310,-200,11);
   write_horz_str(-300,160," Determine if the multigraph in the following"
                           " Figure is connected.",31);
  drawLine(20,120,20,20,11);
  drawLine(-20,120,-20,20,11);
  aspect = 1.0;
  fillOval(20,120,3,16,aspect);
  fillOval(-20,120,3,16,aspect);
  fillOval(20,20,3,16,aspect);
  fillOval(-20,20,3,16,aspect);

  write_horz_str(-300,-30," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,-30," Please type y or n",31);
     Ch = getch ();
     if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
     write_horz_str(0,-30,"                    ",31);
  }
 switch (Ch)        {
  case 'y':write_horz_str(-30,-30," y ",31);
        write_horz_str(-30,-50," No, it is not connected, because",31);
        write_horz_str(-30,-70," there are two different components",31);
        write_horz_str(-30,-90," that are not connected to eachother",31);
        break;
  case 'Y' :write_horz_str(-30,-30," Y ",31);
        write_horz_str(-30,-50," No, it is not connected, because",31);
        write_horz_str(-30,-70," there are two different components",31);
        write_horz_str(-30,-90," that are not connected to eachother",31);
        break;
```

```
   case 'n' :write_horz_str(-30,-30," n ",31);
         write_horz_str(-30,-50," No, it is not connected, because",31);
         write_horz_str(-30,-70," there are two different components",31);
         write_horz_str(-30,-90," that are not connected to eachother",31);
         break;
   case 'N' :write_horz_str(-30,-30," N ",31);
         write_horz_str(-30,-50," No, it is not connected, because",31);
         write_horz_str(-30,-70," there are two different components",31);
         write_horz_str(-30,-90," that are not connected to eachother",31);
          break;
   default : break;   }
   wait("Exercise 3");
}

/*************************************************************************/


void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
       line = 29;
     else
       line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab, 0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
     cls(0);
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
   case 'y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'Y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'n':write_horz_str(-300,-210,"                          ",31);
           break;
    case 'N':write_horz_str(-300,-210,"                          ",31);
           break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : pq4.c
   AUTHOR     : Yavuz BAS
   DATE       : Feb. 4, 1990
   REVISED    : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/



char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*****************************************************
                        FUNCTION DEFINITIONS
        *****************************************************/

void wait(char title[]);
void question_4  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*****************************************************
                        MAIN PROGRAM
        *****************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
           setMode(0x12);
           cls(1);
        }
      if (adapt == 'E')
        {
           setMode(16);
           cls(1);
         }
      if (adapt == 'C')
         {
           setMode(4);
           setCGAPalette(0);
           cls(1);
         }
   question_4();
   setMode(3);
}
```

```
/
*************************************************************************/
static void question_4(void)
{
   LINEWIDTH = 3;
   cls(1);
   drawRect(-309,210,310,-200,11);
  write_horz_str(-300,160," Determine if the multigraph in the following"
                          " Figure is connected.",31);
  drawLine(0,120,-40,70,11);
  drawLine(-40,70,0,20,11);

  aspect = 1.0;
  drawOval(20,120,20,11,aspect);
  drawOval(20,20,20,11,aspect);
  fillOval(0,120,3,16,aspect);
  fillOval(-40,70,3,16,aspect);
  fillOval(0,20,3,16,aspect);
  fillOval(40,20,3,16,aspect);
  fillOval(40,120,3,16,aspect);

  write_horz_str(-300,-30," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,-30," Please type y or n",31);
     Ch = getch ();
     if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
     write_horz_str(0,-30,"                    ",31);
  }

 switch (Ch)        {
 case 'y':write_horz_str(-30,-30," y ",31);
        write_horz_str(-30,-50," Yes, it is connected because",31);
        write_horz_str(-30,-70," every two vertices of the graph",31);
        write_horz_str(-30,-90," are connected to eachother",31);
```

```
        break;
   case 'Y' :write_horz_str(-30,-30," Y ",31);
        write_horz_str(-30,-50," Yes, it is connected because",31);
        write_horz_str(-30,-70," every two vertices of the graph",31);
        write_horz_str(-30,-90," are connected to eachother",31);
        break;
   case 'n' :write_horz_str(-30,-30," n ",31);
        write_horz_str(-30,-50," Yes, it is connected because",31);
        write_horz_str(-30,-70," every two vertices of the graph",31);
        write_horz_str(-30,-90," are connected to eachother",31);
        break;
   case 'N' :write_horz_str(-30,-30," N ",31);
        write_horz_str(-30,-50," Yes, it is connected because",31);
        write_horz_str(-30,-70," every two vertices of the graph",31);
        write_horz_str(-30,-90," are connected to eachother",31);
         break;
   default : break;   }
   wait("Exercise 4");
}
/
*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
       line = 29;
     else
       line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
```

```
    cls(0);
}
/
*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
     write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
              videoinit();
              normal_exit();
              break;
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;
    case 'n':write_horz_str(-300,-210,"                          ",31);
              break;
    case 'N':write_horz_str(-300,-210,"                          ",31);
              break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/
*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : pq5.c
   AUTHOR     : Yavuz BAS
   DATE       : Feb. 4, 1990
   REVISED    : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*****************************************  **********
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        ************************************************************/



char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*****************************************************
                        FUNCTION DEFINITIONS
        *****************************************************/

void wait(char title[]);
void question_5  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*****************************************************
                        MAIN PROGRAM
        *****************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
           setMode(0x12);
           cls(1);
         }
     if (adapt == 'E')
        {
           setMode(16);
           cls(1);
         }
     if (adapt == 'C')
       {
           setMode(4);
           setCGAPalette(0);
           cls(1);
        }
   question_5();
   setMode(3);
}
```

```
/
*****************************************************************************/
static void question_5(void)
{
   LINEWIDTH = 3;
   cls(1);
   drawRect(-309,210,310,-200,11);
   write_horz_str(-300,160," Determine if the multigraph in the following"
                           " Figure has an Euler path.",31);
  drawLine(0,100,100,100,11);
  drawLine(0,100,100,50,11);
  drawLine(0,100,100,0,11);

  drawLine(100,100,0,50,11);
  drawLine(100,100,0,0,11);

  drawLine(0,50,100,50,11);
  drawLine(0,50,100,0,11);

  drawLine(100,50,0,0,11);

  drawLine(0,0,100,0,11);
  aspect = 1.0;
  fillOval(0,100,3,16,aspect);
  fillOval(100,100,3,16,aspect);
  fillOval(100,50,3,16,aspect);
  fillOval(0,50,3,16,aspect);
  fillOval(0,0,3,16,aspect);
  fillOval(100,0,3,16,aspect);

  write_horz_str(-300,-30," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,-30," Please type y or n",31);
     Ch = getch ();
```

```
    if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
    write_horz_str(0,-30,"                    ",31); }
switch (Ch)        {
 case 'y':write_horz_str(-30,-30," y ",31);
        write_horz_str(-30,-50," No, it does not have an Euler",31);
        write_horz_str(-30,-70," path, because you cannot find a",31);
        write_horz_str(-30,-90," path that includes exactly once",31);
        write_horz_str(-30,-110," all the edges of the graph.",31);
        break;
  case 'Y' :write_horz_str(-30,-30," Y ",31);
        write_horz_str(-30,-50," No, it does not have an Euler",31);
        write_horz_str(-30,-70," path, because you cannot find a",31);
        write_horz_str(-30,-90," path that includes exactly once",31);
        write_horz_str(-30,-110," all the edges of the graph.",31);
        break;

  case 'n' :write_horz_str(-30,-30," n ",31);
        write_horz_str(-30,-50," No, it does not have an Euler",31);
        write_horz_str(-30,-70," path, because you cannot find a",31);
        write_horz_str(-30,-90," path that includes exactly once",31);
        write_horz_str(-30,-110," all the edges of the graph.",31);
        break;
  case 'N' :write_horz_str(-30,-30," N ",31);
        write_horz_str(-30,-50," No, it does not have an Euler",31);
        write_horz_str(-30,-70," path, because you cannot find a",31);
        write_horz_str(-30,-90," path that includes exactly once",31);
        write_horz_str(-30,-110," all the edges of the graph.",31);
         break;
  default : break;   }

  wait("Exercise 5");
}
```

```
/
*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
        line = 29;
      else
        line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
       if(waitkey()==ESC) confirm_graph_exit();
       cls(0);
}
/
*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)         {
```

```
    case 'y': setMode(3);
              videoinit();
              normal_exit();
              break;
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;
    case 'n':write_horz_str(-300,-210,"                    ",31);
              break;
    case 'N':write_horz_str(-300,-210,"                    ",31);
              break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/
*************************************************** ********************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/
*****************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : pq6.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
/*************************************************
                FUNCTION DEFINITIONS
*************************************************/

void wait(char title[]);
void question_6  (void);
void delay_ (unsigned duration);
static void confirm_graph_exit(void);
static void normal_exit(void);

/*************************************************
                MAIN PROGRAM
*************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
       {
          setMode(0x12);
          cls(1);
       }
     if (adapt == 'E')
       {
          setMode(16);
          cls(1);
       }
     if (adapt == 'C')
       {
          setMode(4);
          setCGAPalette(0);
          cls(1);
       }
  question_6();
  setMode(3);
}
```

```
/
*************************************************************************/
static void question_6(void)
{
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);
     write_horz_str(-300,160," Determine if the multigraph in the following"
                              " Figure has an Euler path.",31);
     drawLine(-110,100,40,100,11);   /* c */
     drawLine(-110,100,-140,60,11); /* b */
     drawLine(-140,60,30,50,11);      /* n */
     drawLine(30,50,40,100,11);       /* p */
     drawLine(30,50,30,-40,11);       /* f */
     drawLine(30,-40,100,30,11);      /* i */
     drawLine(30,50,100,30,11);       /* e */
     drawLine(40,100,100,30,11);      /* d */
     drawLine(30,-40,-140,25,11);     /* h */
     drawLine(-140,25,-140,60,11);   /* g */

     aspect = .30;
     LINEWIDTH = 1;
     drawArc(65,-5,37,2500,730,11,aspect);
     aspect = 2.0;
     drawArc(-125,76,38,400,2000,11,aspect);
     aspect = 1.0;
     drawOval(58,118,25,11,aspect);
     fillOval(-110,100,3,16,aspect);
     fillOval(40,100,3,16,aspect);
     fillOval(-140,60,3,16,aspect);
     fillOval(30,50,3,16,aspect);
     fillOval(30,-40,3,16,aspect);
     fillOval(100,30,3,16,aspect);
     fillOval(-140,25,3,16,aspect);
     write_horz_char(30,-50,90,31);       /* Z */
     write_horz_char(105,25,89,31);       /* Y */
     write_horz_char(10,70,88,31);        /* X */
     write_horz_char(50,110,87,31);       /* W */
     write_horz_char(-100,115,86,31);    /* V */
```

```
    write_horz_char(-160,60,85,31);     /* U */
    write_horz_char(-150,25,82,31);     /* R */
    write_horz_char(-160,90,97,31);     /* a */
    write_horz_char(-120,80,98,31);     /* b */
    write_horz_char(-60,115,99,31);     /* c */
    write_horz_char(80,75,100,31);      /* d */
    write_horz_char(60,35,101,31);      /* e */

    write_horz_char(40,5,102,31);       /* f */
    write_horz_char(-135,48,103,31);    /* g */
    write_horz_char(-45,-18,104,31);    /* h */
    write_horz_char(60,10,105,31);      /* i */
    write_horz_char(160,10,106,31);     /* j */
    write_horz_char(75,150,109,31);     /* m */
    write_horz_char(-55,45,110,31);     /* n */
    write_horz_char(40,75,112,31);      /* p */

    write_horz_str(-300,-70," Enter your choice as y or n --->",31);
    Ch = waitkey();
    if(Ch==ESC) confirm_graph_exit();
    while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
        write_horz_str(0,-70," Please type y or n",31);
        Ch = getch ();
        if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
        write_horz_str(0,-70,"                    ",31);
    }
switch (Ch)         {
case 'y':write_horz_str(-30,-70," y ",31);
        write_horz_str(-30,-90," Yes, it has an Eulerian path,",31);
        write_horz_str(-30,-110," now press a key to see how.",31);
        break;
 case 'Y' :write_horz_str(-30,-70," Y ",31);
        write_horz_str(-30,-90," Yes, it has an Eulerian path,",31);
        write_horz_str(-30,-110," now press a key to see how.",31);
        break;

 case 'n' :write_horz_str(-30,-70," n ",31);
        write_horz_str(-30,-90," Yes, it has an Eulerian path,",31);
        write_horz_str(-30,-110," now press a key to see how.",31);
```

```
      break;
case 'N' :write_horz_str(-30,-70," N ",31);
      write_horz_str(-30,-90," Yes, it has an Eulerian path,",31);
      write_horz_str(-30,-110," now press a key to see how.",31);
       break;
default : break;   }
if(waitkey()==ESC) confirm_graph_exit();
aspect = 2.0;
drawArc(-125,76,38,400,2000,28,aspect);
delay_(36);

LINEWIDTH = 3;
drawLine(-140,60,-110,100,28); /* b */
delay_(36);
drawLine(-110,100,40,100,28);  /* c */
delay_(36);

LINEWIDTH = 1;
aspect = 1.0;
drawOval(58,118,25,28,aspect);
delay_(36);

LINEWIDTH = 3;
drawLine(40,100,100,30,28);   /* d */
delay_(36);
drawLine(100,30,30,-40,28);  /* i */
delay_(36);

 aspect = .30;
 LINEWIDTH = 1;
drawArc(65,-5,37,2500,730,28,aspect);
delay_(36);

LINEWIDTH = 3;
drawLine(100,30,30,50,28);   /* e */
delay_(36);
```

```
    drawLine(30,50,30,-40,28);   /* f */
    delay_(36);
    drawLine(30,-40,-140,25,28);  /* h */
    delay_(36);
    drawLine(-140,25,-140,60,28); /* g */
    delay_(36);
    drawLine(-140,60,30,50,28);  /* n */
    delay_(36);
    drawLine(30,50,40,100,28);    /* p */
    delay_(36);
    write_horz_str(-30,-130," Namely, starting from vertex V,",31);
    write_horz_str(-30,-150," a, b, c, m, d, j, i, e, f,",31);
    write_horz_str(-30,-170," h, g, n, p is an Euler path.",31);
    wait("Exercise 6");
    cls(0);
}
/
*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
        line = 29;
      else
         line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
       if(waitkey()==ESC) confirm_graph_exit();
}
```

```
/
*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
      write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'Y': setMode(3);
           videoinit();
           normal_exit();
           break;
    case 'n':write_horz_str(-300,-210,"                          ",31);
           break;
    case 'N':write_horz_str(-300,-210,"                          ",31);
           break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/
*****************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/
*****************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : pq7.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*****************************************************
                        FUNCTION DEFINITIONS
         *****************************************************/

void wait(char title[]);
void question_7  (void);
void delay_ (unsigned duration);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*****************************************************
                          MAIN PROGRAM
         *****************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
        {
          setMode(0x12);
          cls(1);
         }
     if (adapt == 'E')
        {
          setMode(16);
          cls(1);
         }
    if (adapt == 'C')
       {
          setMode(4);
          setCGAPalette(0);
          cls(1);
       }
  question_7();
  setMode(3);
}
```

```
/
*************************************************************************/
static void question_7(void)
{
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);
     write_horz_str(-300,160," Determine if the graph in the following"
                             " Figure has an Euler circuit.",31);
     drawLine (-100,100,-120,30,11);   /* AD */
     drawLine (-120,30,-70,70,11);     /* DC */
     drawLine (-70,70,-130,70,11);     /* CB */
     drawLine (-130,70,-80,30,11);     /* BE */
     drawLine (-80,30,-100,100,11);    /* EA */

     aspect = 1.0;
     fillOval(-100,100,3,16,aspect);
     fillOval(-120,30,3,16,aspect);
     fillOval(-70,70,3,16,aspect);
     fillOval(-130,70,3,16,aspect);
     fillOval(-80,30,3,16,aspect);

     write_horz_char(-100,120,65,31); /* A */
     write_horz_char(-145,70,66,31);  /* B */
     write_horz_char(-65,70,67,31);   /* C */
     write_horz_char(-135,30,68,31);  /* D */
     write_horz_char(-75,30,69,31);   /* E */

   write_horz_str(-300,-70," Enter your choice as y or n --->",31);
   Ch = waitkey();
   if(Ch==ESC) confirm_graph_exit();
   while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
      write_horz_str(0,-70," Please type y or n",31);
      Ch = getch ();
      if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
      write_horz_str(0,-70,"                    ",31);
```

```
  }
switch (Ch)        {
 case 'y':write_horz_str(-30,-70," y ",31);
       write_horz_str(-30,-90," Yes, it has an Euler circuit,",31);
       write_horz_str(-30,-110," now press a key to see how.",31);
       break;
  case 'Y' :write_horz_str(-30,-70," Y ",31);
       write_horz_str(-30,-90," Yes, it has an Euler circuit,",31);
       write_horz_str(-30,-110," now press a key to see how.",31);
       break;
  case 'n' :write_horz_str(-30,-70," n ",31);
       write_horz_str(-30,-90," Yes, it has an Euler circuit,",31);
       write_horz_str(-30,-110," now press a key to see how.",31);
       break;
  case 'N' :write_horz_str(-30,-70," N ",31);
       write_horz_str(-30,-90," Yes, it has an Euler circuit,",31);
       write_horz_str(-30,-110," now press a key to see how.",31);
        break;
  default : break;   }
  if(waitkey()==ESC) confirm_graph_exit();
  drawLine (-100,100,-120,30,28);   /* AD */
  delay_(36);
  drawLine (-120,30,-70,70,28);      /* DC */
  delay_(36);
  drawLine (-70,70,-130,70,28);      /* CB */
  delay_(36);
  drawLine (-130,70,-80,30,28);      /* BE */
  delay_(36);
  drawLine (-80,30,-100,100,28);     /* EA */
  write_horz_str(-30,-130," Namely, starting from vertex A,",31);
  write_horz_str(-30,-150," and following the vertices D, C",31);
  write_horz_str(-30,-170," B, E, A gives us an Euler circuit.",31);
  wait("Exercise 7");
  cls(0);
}
```

```
/
*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
        line = 29;
      else
        line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
}
/
*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       write_horz_str(30,-210," Please type y or n",79);
       ch = getch ();
       write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': setMode(3);
```

```
            videoinit();
            normal_exit();
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;
    case 'n':write_horz_str(-300,-210,"                    ",31);
            break;
    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor    */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.            */
/
*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : pq8.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
         *************************************************/

void wait(char title[]);
void question_8  (void);
void delay_ (unsigned duration);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                        MAIN PROGRAM
         *************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
      if (adapt == 'V')
        {
          setMode(0x12);
          cls(1);
         }
     if (adapt == 'E')
        {
          setMode(16);
          cls(1);
         }
      if (adapt == 'C')
        {
          setMode(4);
           setCGAPalette(0);
           cls(1);
         }
  question_8();
  setMode(3);
}
```

```
/
*************************************************************************/
static void question_8(void)
{
    LINEWIDTH = 3;
    cls(1);
   drawRect(-309,210,310,-200,11);
  write_horz_str(-300,160," Determine if the graph in the following"
                          " Figure has an Euler circuit.",31);
  drawLine(-50,105,50,80,11);
  drawLine(-50,105,50,30,11);

  drawLine(-50,55,50,80,11);
  drawLine(-50,55,50,30,11);

  drawLine(-50,5,50,30,11);
  drawLine(-50,5,50,80,11);
  aspect = 1.0;
  fillOval(-50,105,3,16,aspect);
  fillOval(50,80,3,16,aspect);
  fillOval(50,30,3,16,aspect);
  fillOval(-50,55,3,16,aspect);
  fillOval(-50,5,3,16,aspect);
  write_horz_str(-300,-70," Enter your choice as y or n --->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     write_horz_str(0,-70," Please type y or n",31);
     Ch = getch ();
     if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
     write_horz_str(0,-70,"                    ",31); }
 switch (Ch)        {
  case 'y':write_horz_str(-30,-70,"y ",31);
        write_horz_str(-30,-90,"No, it doesn't have an  Euler",31);
        write_horz_str(-30,-110,"circuit, because there is no path",31);
        write_horz_str(-30,-130,"that includes exactly once all the",31);
```

```
        write_horz_str(-30,-150,"edges of the graph and has the same",31);
        write_horz_str(-30,-170,"initial and terminal vertices.",31);
        break;
   case 'Y' :write_horz_str(-30,-70," Y ",31);
        write_horz_str(-30,-90,"No, it doesn't have an  Euler",31);
        write_horz_str(-30,-110,"circuit, because there is no path",31);
        write_horz_str(-30,-130,"that includes exactly once all the",31);
        write_horz_str(-30,-150,"edges of the graph and has the same",31);
        write_horz_str(-30,-170,"initial and terminal vertices.",31);
        break;
   case 'n' :write_horz_str(-30,-70," n ",31);
        write_horz_str(-30,-90,"No, it doesn't have an  Euler",31);
        write_horz_str(-30,-110,"circuit, because there is no path",31);
        write_horz_str(-30,-130,"that includes exactly once all the",31);
        write_horz_str(-30,-150,"edges of the graph and has the same",31);
        write_horz_str(-30,-170,"initial and terminal vertices.",31);
        break;
   case 'N' :write_horz_str(-30,-70," N ",31);
        write_horz_str(-30,-90,"No, it doesn't have an  Euler",31);
        write_horz_str(-30,-110,"circuit, because there is no path",31);
        write_horz_str(-30,-130,"that includes exactly once all the",31);
        write_horz_str(-30,-150,"edges of the graph and has the same",31);
        write_horz_str(-30,-170,"initial and terminal vertices.",31);
         break;
   default : break;   }
   wait("Exercise 8");
   cls(0);
}
```

```
/
*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
        line = 29;
      else
        line = 24;
       tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
}
/
*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
   write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
    }
    switch (ch)         {
     case 'y': setMode(3);
```

```
            videoinit();
            normal_exit();
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;
    case 'n':write_horz_str(-300,-210,"                    ",31);
            break;
    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor    */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.            */
/
*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : pq9.c
   AUTHOR       : Yavuz BAS
   DATE         : Feb. 4, 1990
   REVISED      : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
/*************************************************
                FUNCTION DEFINITIONS
 *************************************************/

void wait(char title[]);
void question_9  (void);
void delay_ (unsigned duration);
static void confirm_graph_exit(void);
static void normal_exit(void);

/*************************************************
                MAIN PROGRAM
 *************************************************/

main()
{
     adapt = getAdapter();
     if (adapt == 'V')
       {
          setMode(0x12);
          cls(1);
       }
      if (adapt == 'E')
       {
          setMode(16);
          cls(1);
       }
      if (adapt == 'C')
       {
          setMode(4);
          setCGAPalette(0);
          cls(1);
       }
  question_9();
  setMode(3);
}
```

```
/
*************************************************************************/
static void question_9(void)
{
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);
     write_horz_str(-300,160," Determine if the graph in the following"
                             " Figure has an Hamiltonian cycle.",31);
     drawLine (-100,100,-120,30,11);   /* AD */
     drawLine (-120,30,-70,70,11);     /* DC */
     drawLine (-70,70,-130,70,11);     /* CB */
     drawLine (-130,70,-80,30,11);     /* BE */
     drawLine (-80,30,-100,100,11);    /* EA */
     aspect = 1.0;
     fillOval(-100,100,3,16,aspect);
     fillOval(-120,30,3,16,aspect);
     fillOval(-70,70,3,16,aspect);
     fillOval(-130,70,3,16,aspect);
     fillOval(-80,30,3,16,aspect);

     write_horz_char(-100,120,65,31); /* A */
     write_horz_char(-145,70,66,31);  /* B */
     write_horz_char(-65,70,67,31);   /* C */
     write_horz_char(-135,30,68,31);  /* D */
     write_horz_char(-75.30,69,31);   /* E */

     write_horz_str(-300,-70," Enter your choice as y or n --->",31);
     Ch = waitkey();
     if(Ch==ESC) confirm_graph_exit();
     while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
       write_horz_str(0,-70," Please type y or n",31);
        Ch = getch ();
        if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
        write_horz_str(0,-70,"                    ",31); }
 switch (Ch)        {
```

```
case 'y':write_horz_str(-30,-70," y ",31);
      write_horz_str(-30,-90," Yes, it has an Hamiltonian cycle,",31);
      write_horz_str(-30,-110," as well as an Euler circuit.",31);
      write_horz_str(-30,-130," Now press a key to see how.",31);
      break;
 case 'Y' :write_horz_str(-30,-70," Y ",31);
      write_horz_str(-30,-90," Yes, it has an Hamiltonian cycle,",31);
      write_horz_str(-30,-110," as well as an Euler circuit.",31);
      write_horz_str(-30,-130," Now press a key to see how.",31);
      break;
 case 'n' :write_horz_str(-30,-70," n ",31);
      write_horz_str(-30,-90," Yes, it has an Hamiltonian cycle,",31);
      write_horz_str(-30,-110," as well as an Euler circuit.",31);
      write_horz_str(-30,-130," Now press a key to see how.",31);
      break;
 case 'N' :write_horz_str(-30,-70," N ",31);
      write_horz_str(-30,-90," Yes, it has an Hamiltonian cycle,",31);
      write_horz_str(-30,-110," as well as an Euler circuit.",31);
      write_horz_str(-30,-130," Now press a key to see how.",31);
       break;
 default : break;   }
 if(waitkey()==ESC) confirm_graph_exit();
 drawLine (-100,100,-120,30,28);   /* AD */
 delay_(36);
 drawLine (-120,30,-70,70,28);     /* DC */
 delay_(36);
 drawLine (-70,70,-130,70,28);     /* CB */
 delay_(36);
 drawLine (-130,70,-80,30,28);     /* BE */
 delay_(36);
 drawLine (-80,30,-100,100,28);    /* EA */
 write_horz_str(-30,-130," Namely, starting from vertex A,",31);
 write_horz_str(-30,-150," and following the vertices D, C",31);
 write_horz_str(-30,-170," B, E, A gives us an Hamiltonian cycle.",31);
 wait("Exercise 9");
 cls(0);
```

```
}
/
*****************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
        line = 29;
      else
        line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
       if(waitkey()==ESC) confirm_graph_exit();
}
/
*****************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       write_horz_str(30,-210," Please type y or n",79);
       ch = getch ();
       write_horz_str(30,-210,"                    ",31);
    }
    switch (ch)         {
     case 'y': setMode(3);
```

```
            videoinit();
            normal_exit();
            break;
     case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;
     case 'n':write_horz_str(-300,-210,"                    ",31);
            break;
     case 'N':write_horz_str(-300,-210,"                    ",31);
            break;
     default : break;
     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor    */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.            */
/
*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : pq10.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "Paths and Circuits".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void question_10  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                        MAIN PROGRAM
        *************************************************/
main()
{
      adapt = getAdapter();
      if (adapt == 'V')
         {
           setMode(0x12);
           cls(1);
         }
      if (adapt == 'E')
        {
           setMode(16);
           cls(1);
         }
      if (adapt == 'C')
        {
           setMode(4);
           setCGAPalette(0);
           cls(1);
         }
   question_10();
   setMode(3);
}
```

```
/
*****************************************************************************/
static void question_10(void)
{
   LINEWIDTH = 3;
   cls(1);
   drawRect(-309,210,310,-200,11);
   write_horz_str(-300,160," Determine if the graph in the following"
                           " Figure has an Hamiltonian cycle.",31);

   drawRect(-25,75,25,25,11);
   drawLine(-25,75,25,25,11);
   drawLine(25,75,-25,25,11);
   drawLine(0,100,-50,50,11);
   drawLine(-50,50,0,0,11);
   drawLine(50,50,0,0,11);
   drawLine(50,50,0,100,11);

   aspect = 1.0;
   fillOval(0,100,3,16,aspect);
   fillOval(0,50,3,16,aspect);
   fillOval(-25,75,3,16,aspect);
   fillOval(-50,50,3,16,aspect);
   fillOval(-25,25,3,16,aspect);
   fillOval(0,0,3,16,aspect);
   fillOval(25,25,3,16,aspect);
   fillOval(50,50,3,16,aspect);
   fillOval(25,75,3,16,aspect);

   write_horz_str(-300,-70," Enter your choice as y or n --->",31);
   Ch = waitkey();
   if(Ch==ESC) confirm_graph_exit();
   while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
      write_horz_str(0,-70," Please type y or n",31);
      Ch = getch ();
      if ((Ch == 'y') || (Ch == 'Y') || (Ch == 'n') || (Ch == 'N'))
```

```
      write_horz_str(0,-70,"                    ",31); }
  switch (Ch)       {
   case 'y':write_horz_str(-30,-70," y ",31);
         write_horz_str(-30,-90," No, it does not have an Hamiltonian",31);
         write_horz_str(-30,-110," cycle, because there is no cycle",31);
         write_horz_str(-30,-130," that includes each vertex.",31);
         break;
    case 'Y' :write_horz_str(-30,-70," Y ",31);
         write_horz_str(-30,-90," No, it does not have an Hamiltonian",31);
         write_horz_str(-30,-110," cycle, because there is no cycle",31);
         write_horz_str(-30,-130," that includes each vertex.",31);
         break;
    case 'n' :write_horz_str(-30,-70," n ",31);
         write_horz_str(-30,-90," No, it does not have an Hamiltonian",31);
         write_horz_str(-30,-110," cycle, because there is no cycle",31);
         write_horz_str(-30,-130," that includes each vertex.",31);
         break;
    case 'N' :write_horz_str(-30,-70," N ",31);
         write_horz_str(-30,-90," No, it does not have an Hamiltonian",31);
         write_horz_str(-30,-110," cycle, because there is no cycle",31);
         write_horz_str(-30,-130," that includes each vertex.",31);
          break;
    default  : break;   }
    wait("Exercise 10");
    cls(0);
}
/
*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
       line = 29;
     else
       line = 24;
```

```
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
}
/
*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
    write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
      write_horz_str(30,-210,"                ",31);    }
   switch (ch)        {
   case 'y': setMode(3);
          videoinit();
          normal_exit();
          break;
   case 'Y': setMode(3);
          videoinit();
          normal_exit();
          break;
   case 'n':write_horz_str(-300,-210,"                        ",31);
          break;
   case 'N':write_horz_str(-300,-210,"                        ",31);
          break;
   default : break;     }
```

```
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor    */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.            */
/
*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM   : color.c
   AUTHOR    : Yavuz BAS
   DATE      : Feb. 4, 1990
   REVISED   : Mar. 5, 1990

   DESCRIPTION : This program contains the tutorial for the "coloring a  graph."
   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C compiler Version 2.0.
*/
/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)    /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
#define _GRAPH_T_DEFINED
```

```
/* function prototypes */
static void add_shadow     (void);
static void confirm_quit   (void);
static void disp_sure_msg  (void);
static void do_nothing     (void);
static void error_exit     (int errnum);
static void move_window    (int nsrow, int scol);
static void normal_exit    (void);
static void press_a_key    (int wrow);
static void quit_window    (void);
static void restore_cursor (void);
static void short_delay    (void);
static void size_window    (int nerow,int necol);
static void example_3_22   (void);
static void example_3_23   (void);
static void example_3_24   (void);
static void example_3_25   (void);
static void example_3_26   (void);
static void example_3_27   (void);
static void example_3_28   (void);
static void example_3_29   (void);
static void thm_3_6        (void);
static void thm_3_7        (void);
static void Welsh_Powell   (void);
static void definition_3_15(void);
static void exercises      (void);
static void coloring       (void);
static void Pexercise1(void);
static void Pexercise2(void);
static void Pexercise3(void);
static void Pexercise4(void);
static void exercise1(void);
static void exercise2(void);
static void exercise3(void);
static void exercise4(void);
static void Pex_3_22   (void);
```

```
static void Pex_3_23   (void);
static void Pex_3_24   (void);
static void Pex_3_25   (void);
static void Pex_3_26   (void);
static void Pex_3_27   (void);
static void Pex_3_28   (void);
static void Pex_3_29   (void);
static void Pdef_3_15(void);
static void Pthm_3_6(void);
static void Pthm_3_7(void);
static void PWelsh_Powell(void);
static void following22(void);
static void following23(void);
static void following24(void);
static void following25(void);
static void following26(void);
static void following27(void);
static void following28(void);
static void following29(void);
static void followingthm_3_6(void);
/
*****************************************************************************/
/* miscellaneous global variables */
static int *savescm,crow,ccol;
static WINDOW w[10];
static char ssan[10];
/
*****************************************************************************/
/* error message table */
static char *error_text[]= {
   NULL,  /* ermum =  0, no error   */
   NULL,  /* ermum == 1, windowing error */
   "Syntax:  CXLDEMO [-switches]\n\n"
      "\t -c = CGA snow elimination\n"
      "\t -b = BIOS screen writing\n"
      "\t -m = force monochrome text attributes",
```

```
    "Memory allocation error"
};
/* miscellaneous defines */
#define SHORT_DELAY 18
#define H_WINTITLE  33
/
*************************************************************************/
/* this function will add a shadow to the active window                  */
/
*************************************************************************/
static void add_shadow(void)
{
   wshadow(LGREY|_BLACK);
}
/
*************************************************************************/
/* this function pops open a window and confirms that the user really    */
/* wants to quit the demo.  If so, it terminates the demo program.       */
/
*************************************************************************/
static void confirm_quit(void)
{
   struct _onkey_t *kblist;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
   add_shadow();
   wputs("\n Quit demo, are you sure? \033A\156Y\b");
   clearkeys();
   showcur();
   if(wgetchf("YN",'Y')=='Y') normal_exit();
   wclose();
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);      /* restore any hidden hot keys */
}
```

```
/
*************************************************************************/
/* this function is called by the pull-down demo for a prompt */
static void disp_sure_msg(void)
{
  wprints(0,2,WHITE|_BLUE,"Are you sure?");
}
/
*************************************************************************/
/*                    this function is used as a dummy function for                    */
/*                    several menu items in the pull-down demo                         */
/
*************************************************************************/
static void do_nothing(void)
{
}
/
*************************************************************************/
/* this function handles abnormal termination.  If it is passed an error code of 1, then */
/* it is a windowing system error. Otherwise the error message is looked up in the   */
/*error message table.                                                                 */
/
*************************************************************************/
static void error_exit(int errnum)
{
  if(errnum) {
    printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
          exit(errnum);
  }
}
/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor          */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.                  */
/
*************************************************************************/
```

```
static void normal_exit(void)
{
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}




/
*****************************************************************************/
/* this function displays a pause message then pauses for a keypress        */
/
*****************************************************************************/
static void press_a_key(int wrow)
{
   register int attr1;
   register int attr2;
   attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
   attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
   wcenters(wrow,attr1,"Press a key");
   wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
   hidecur();
   if(waitkey()==ESC) confirm_quit();
   wcenters(wrow,attr1,"          ");
   wprints(wrow,0,attr2,"         ");
}
/
*****************************************************************************/
static void short_delay(void)
{
   delay_(SHORT_DELAY);
```

```
}
/
*************************************************************************/
/* this function is called by the pull-down menu demo anytime            */
/* the  selection bar moves on or off the [Q]uit menu items.              */
/
*************************************************************************/
static void quit_window(void)
{
   static WINDOW handle=0;

   if(handle) {
      wactiv(handle);
      wclose();
      handle=0;
   }
   else {
      handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
      wputs(" Quit takes you back to the\n demo program's main menu.");
   }
}
/
*************************************************************************/
static void restore_cursor(void)
{
   wtextattr(WHITE|_MAGENTA);
   showcur();
}
/
*************************************************************************/
static void size_window(int nerow,int necol)
{
   wsize(nerow,necol);
   short_delay();
}
```

```
/
*************************************************************************/
static void move_window(int nsrow,int nscol)
{
   if(wmove(nsrow,nscol)) error_exit(1);
   short_delay();
}
/
*************************************************************************/
void main()
{
  coloring();
}
/
*************************************************************************/
static void coloring (void)

{
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   cclrscrn(LGREY|_GREEN);
   definition_3_15();
   srestore(scrn);
}
/******* ************************************************************************/
 void definition_3_15(void)
 {
   /*****************************************************************/
   /* attach [Pageup] to the definition_3_15 function */
   setonkey(0x4900,Pdef_3_15,0);
   /*****************************************************************/
   /* attach [Pagedown] to the example_3_22 function */
   setonkey(0x5100,Pex_3_22,0);
   /*****************************************************************/
   if((w[1]=wopen(6,10,10,70,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
      error_exit(1);
```

```
  wtitle("[To color a graph]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" To color a graph means to assign a color to each vertex such"
         " that adjacent vertices have different colors.");
  press_a_key(2);
  if((w[2]=wopen(12,10,17,70,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
    error_exit(1);
  wtitle("[Chromatic number]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" When a graph G can be colored with n colors but not with a"
         " smaller number of colors, G is said to have chromatic number"
         " n.");
  press_a_key(2);
  wclose();
  wclose();
  following22();
 }
/
*************************************************************************/
void following22(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the definition_3_15 function */
  setonkey(0x4900,Pdef_3_15,0);
  /*********************************************************************/
  /* attach [Pagedown] to the example_3_23 function */
  setonkey(0x5100,Pex_3_23,0);
  /*********************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_CYAN,WHITE|_CYAN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see an example....");
   press_a_key(2);
   wcloseall();
   example_3_22();
}
```

```
/
*****************************************************************************/
 void example_3_22(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_22.exe",NULL);
  srestore(scrn);
  following23();
 }
/
*****************************************************************************/
void following23(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the example_3_22 function */
  setonkey(0x4900,Pex_3_22,0);
  /*************************************************************************/
  /* attach [Pagedown] to the example_3_24 function */
  setonkey(0x5100,Pex_3_24,0);
  /*************************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_RED,WHITE|_RED))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see an example....");
   press_a_key(2);
   wcloseall();
   example_3_23();
}
/
*****************************************************************************/
 void example_3_23(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_23.exe",NULL);
```

```
  srestore(scrn);
  example_3_24();
 }
/
*************************************************************************/
 void example_3_24(void)
 {
  /*************************************************************/
  /* attach [Pageup] to the example_3_23 function */
  setonkey(0x4900,Pex_3_23,0);
  /*************************************************************/
  /* attach [Pagedown] to the example_3_25 function */
  setonkey(0x5100,Pex_3_25,0);
  /*************************************************************/
  if((w[1]=wopen(9,10,15,70,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
    error_exit(1);
  wtitle("[Example 3.24]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" The complete graph Kn with n vertices can be colored using"
         " n colors. But since every vertex is adjacent to every other"
         " vertex, a smaller number of colors will not work. Thus, Kn"
         " has chromatic number n.");
  press_a_key(4);
  wclose();
  following25();

 }


/
*************************************************************************/
void following25(void)
{
  /*************************************************************/
  /* attach [Pageup] to the example_3_24 function */
  setonkey(0x4900,Pex_3_24,0);
```

```
  /*****************************************************************/
  /* attach [Pagedown] to the example_3_26 function */
  setonkey(0x5100,Pex_3_26,0);
  /*****************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see an example....");
   press_a_key(2);
   wcloseall();
   example_3_25();
}
/
*******************************************************************************/
 void example_3_25(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_25.exe",NULL);
  srestore(scrn);
  following26();
 }
/
*******************************************************************************/
void following26(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the example_3_25 function */
  setonkey(0x4900,Pex_3_25,0);
  /*****************************************************************/
  /* attach [Pagedown] to the thm_3_6 function */
  setonkey(0x5100,Pthm_3_6,0);
  /*****************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_BROWN,YELLOW|_BROWN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
```

```
    wputsw("Press a key to see an example....");
    press_a_key(2);
    wcloseall();
    example_3_26();
}
/
*************************************************************************/
 void example_3_26(void)
 {
   register int *scrn;
   if((scrn=ssave())==NULL) error_exit(3);
   spawnl(P_WAIT,"ex3_26.exe",NULL);
   srestore(scrn);
   followingthm_3_6();
 }
/
*************************************************************************/
void followingthm_3_6(void)
{
   /*********************************************************************/
   /* attach [Pageup] to the example_3_26 function */
   setonkey(0x4900,Pex_3_26,0);
   /*********************************************************************/
   /* attach [Pagedown] to the example_3_27 function */
   setonkey(0x5100,Pex_3_27,0);
   /*********************************************************************/
    if((w[1]=wopen(20,42,24,77,3,LCYAN|_LGREY,WHITE|_LGREY))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Press a key to see a theorem....");
    press_a_key(2);
    wcloseall();
    thm_3_6();
}
/
*************************************************************************/
```

```
 void thm_3_6(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"thm3_6.exe",NULL);
  srestore(scrn);
  following27();
 }
/
*************************************************************************/
void following27(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the theorem_3_6 function */
  setonkey(0x4900,Pthm_3_6,0);
  /*********************************************************************/
  /* attach [Pagedown] to the theorem_3_7 function */
  setonkey(0x5100,Pthm_3_7,0);
  /*********************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_BLUE,WHITE|_BLUE))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see an example....");
   press_a_key(2);
   wcloseall();
   example_3_27();
}
/
*************************************************************************/
 void example_3_27(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_27.exe",NULL);
  srestore(scrn);
  thm_3_7();
```

```
 }
/
*****************************************************************/
 void thm_3_7(void)
 {
  /*****************************************************************/
  /* attach [Pageup] to the example_3_27 function */
  setonkey(0x4900,Pex_3_27,0);
  /*****************************************************************/
  /* attach [Pagedown] to the example_3_28 function */
  setonkey(0x5100,Pex_3_28,0);
  /*****************************************************************/
  if((w[1]=wopen(5,10,10,70,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
    error_exit(1);
  wtitle("[Theorem 3.7]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" For a graph G, the chromatic number of G cannot exceed one"
           " more than the maximum of the degrees of the vertices of G.");
  press_a_key(3);
  if((w[2]=wopen(11,10,23,70,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
    error_exit(1);
  wtitle("[Proof of Theorem 3.7]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" Let K be maximum of the degrees of the vertices of G. We"
       " will show that G can be colored using k+1 colors C0,...,Ck."
       " First a vertex V is selected and the color C0 is assigned to"
       " it. Next some other vertex W is picked. Since there are at"
       " most k vertices adjacent to W and there are at least k+1"
       " colors available to choose from, there is at least one color"
       " (possibly many) that has not been used on a vertex adjacent"
       " to W. Choose such a color. This process can be continued"
       " until all the vertices of G are colored.");
  press_a_key(10);
  wclose();
  wclose();
  following28();
```

```
  }
/
*************************************************************************/
void following28(void)
{
  /*********************************************************************/
  /* attach [Pageup] to the theorem_3_7 function */
  setonkey(0x4900,Pthm_3_7,0);
  /*********************************************************************/
  /* attach [Pagedown] to the Welsh_Powell function */
  setonkey(0x5100,PWelsh_Powell,0);
  /*********************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_GREEN,WHITE|_GREEN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see an example....");
   press_a_key(2);
   wcloseall();
   example_3_28();
}
/
*************************************************************************/
 void example_3_28(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_28.exe",NULL);
  srestore(scrn);
  Welsh_Powell();
 }
/
*************************************************************************/
 void Welsh_Powell(void)
 {
  /*********************************************************************/
```

```
/* attach [Pageup] to the example_3_28 function */
setonkey(0x4900,Pex_3_28,0);
/*****************************************************************/
/* attach [Pagedown] to the example_3_29 function */
setonkey(0x5100,Pex_3_29,0);
/*****************************************************************/
if((w[1]=wopen(1,2,5,74,3,LCYAN|_MAGENTA,WHITE|_MAGENTA))==0)
 error_exit(1);
wtitle("[Welsh and Powell Algorithm]",TCENTER,_LGREY|BROWN);
whelpcat(H_WINTITLE);
wputsw(" This algorithm gives a coloring of a graph by labeling the"
       " vertices according to their degrees.");
press_a_key(2);
if((w[2]=wopen(6,2,11,74,3,LCYAN|_CYAN,WHITE|_CYAN))==0)
  error_exit(1);
wtitle("[Step 1]",TCENTER,_LGREY|BROWN);
whelpcat(H_WINTITLE);
wputsw(" (label the vertices by degree). Label the vertices V1, V2,"
       "...,Vn such that deg(V1) >= deg(V2) >= ... >= deg(Vn).(Ties"
       " can be broken arbitrarily.)");
press_a_key(3);
if((w[3]=wopen(11,2,17,74,3,LCYAN|_BLUE,WHITE|_BLUE))==0) error_exit(1);
wtitle("[Step 2]",TCENTER,_LGREY|BROWN);
whelpcat(H_WINTITLE);
wputsw(" (color first uncolored vertex and uncolored vertices adjacent"
    " to it). Assign an unused color to the first uncolored vertex"
    " in the list of vertices. Go through the list of vertices in"
    " order, assigning this new color to any vertex not adjacent to"
    " any other vertex with this color.");
press_a_key(4);
if((w[4]=wopen(17,2,21,74,3,LCYAN|_LGREY,WHITE|_LGREY))==0)
  error_exit(1);
wtitle("[Step 3]",TCENTER,_LGREY|BROWN);
whelpcat(H_WINTITLE);
wputsw(" (graph colored?). If some of the vertices are uncolored, then"
       " return to step 2.");
```

```
  press_a_key(2);
  if((w[5]=wopen(21,2,24,74,3,LCYAN|_BROWN,WHITE|_BROWN))==0)
    error_exit(1);
  wtitle("[Step 4]",TCENTER,_LGREY!BROWN);
  whelpcat(H_WINTITLE);
  wputsw(" (done). A coloring of the graph has been constructed.");
  press_a_key(1);
  wcloseall();
  following29();
 }
/
*****************************************************************************/
void following29(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the Welsh_Powell function */
  setonkey(0x4900,PWelsh_Powell,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exercise1 function */
  setonkey(0x5100,Pexercise1,0);
  /*************************************************************************/
   if((w[1]=wopen(20,42,24,77,3,LCYAN|_CYAN,WHITE|_CYAN))==0)
     error_exit(1);
   whelpcat(H_WINTITLE);
   wputsw("Press a key to see an example....");
   press_a_key(2);
   wcloseall();
   example_3_29();
}
/
*****************************************************************************/
 void example_3_29(void)
 {
  register int *scrn;
  if((scrn=ssave())==NULL) error_exit(3);
  spawnl(P_WAIT,"ex3_29 exe",NULL);
```

```
   srestore(scrn);
   exercise1();
  }



/
*************************************************************************/
void exercise1(void)
{
   register int *scrn;
   /*********************************************************************/
   /* attach [Pageup] to the example_3_29 function */
   setonkey(0x4900,Pex_3_29,0);
   /*********************************************************************/
   /* attach [Pagedown] to the exercise2 function */
   setonkey(0x5100,Pexercise2,0);
   /*********************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see some exercises...");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"cq1.exe",NULL);
    srestore(scrn);
    exercise2();
}
/
*************************************************************************/
void exercise2(void)
{
   register int *scrn;
   /*********************************************************************/
```

```
   /* attach [Pageup] to the exercise1 function */
   setonkey(0x4900,Pexercise1,0);
   /*********************************************************************/
   /* attach [Pagedown] to the exercise3 function */
   setonkey(0x5100,Pexercise3,0);
   /*********************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 2..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"cq2.exe",NULL);
    srestore(scrn);
    exercise3();
}
/
**********************************************************************/
void exercise3(void)
{
   register int *scrn;
   /*********************************************************************/
   /* attach [Pageup] to the exercise2 function */
   setonkey(0x4900,Pexercise2,0);
   /*********************************************************************/
   /* attach [Pagedown] to the exercise4 function */
   setonkey(0x5100,Pexercise4,0);
   /*********************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 3..");
    press_a_key(2);
    wcloseall();
```

```
    spawnl(P_WAIT,"cq3.exe",NULL);
    srestore(scrn);
    exercise4();
}
/
*************************************************************************/
void exercise4(void)
{
   register int *scrn;
   /*********************************************************************/
   /* attach [Pageup] to the exercise3 function */
   setonkey(0x4900,Pexercise3,0);
   /*********************************************************************/
   /* attach [Pagedown] to the normal_exit */
   setonkey(0x5100,normal_exit,0);
   /*********************************************************************/
   if((scrn=ssave())==NULL) error_exit(3);
    if((w[1]=wopen(20,42,24,77,3,MAGENTA|_BLUE,WHITE|_BLUE))==0)
      error_exit(1);
    whelpcat(H_WINTITLE);
    wputsw("Now let us see exercise 4..");
    press_a_key(2);
    wcloseall();
    spawnl(P_WAIT,"cq4.exe",NULL);
    srestore(scrn);
    normal_exit();
}
/
*************************************************************************/
/* this routine calls example_3_22 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/
*************************************************************************/
void Pex_3_22()
{
   wcloseall();
```

```
    following22();
}
/
*************************************************************************/
/* this routine calls example_3_23 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/
*************************************************************************/
void Pex_3_23()
{
    wcloseall();
    following23();
}


/
*************************************************************************/
/* this routine calls example_3_24 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/
*************************************************************************/
void Pex_3_24()
{
    wcloseall();
    example_3_24();
}
/
*************************************************************************/
/* this routine calls example_3_25 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/
*************************************************************************/
void Pex_3_25()
{
    wcloseall();
    following25();
```

```
}
/
*************************************************************************/
/* this routine calls example_3_26 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/
*************************************************************************/
void Pex_3_26()
{
   wcloseall();
   following26();
}
/
*************************************************************************/
/* this routine calls example_3_27 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/
*************************************************************************/
void Pex_3_27()
{
   wcloseall();
   following27();
}
/
*************************************************************************/
/* this routine calls example_3_28 routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/
*************************************************************************/
void Pex_3_28()
{
   wcloseall();
   following28();
}
/
*************************************************************************/
```

```
/* this routine calls example_3_29 routine whenever Pageup or Pagedown          */
/* keys are pressed.                                                            */
/
*******************************************************************************/
void Pex_3_29()
{
   wcloseall();
   following29();
}
/
*******************************************************************************/
/* this routine calls theorem_3_6 routine whenever Pageup or Pagedown           */
/* keys are pressed.                                                            */
/
*******************************************************************************/
void Pthm_3_6()
{
   wcloseall();
   followingthm_3_6();
}
/
*******************************************************************************/
/* this routine calls theorem_3_7 routine whenever Pageup or Pagedown           */
/* keys are pressed.                                                            */
/
*******************************************************************************/
void Ptlm_3_7()
{
   wcloseall();
   thm_3_7();
}
/
*******************************************************************************/
/* this routine calls definition_3_15 routine whenever Pageup or Pagedown       */
/* keys are pressed.                                                            */
```

```
/
*************************************************************************/
void Pdef_3_15()
{
  wcloseall();
  definition_3_15();
}
/
*************************************************************************/
/* this routine calls Welsh_Powell routine whenever Pageup or Pagedown   */
/* keys are pressed.                                                     */
/
*************************************************************************/
void PWelsh_Powell()
{
  wcloseall();
  Welsh_Powell();
}
/
*************************************************************************/
/* this routine calls exercise1 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/
*************************************************************************/
void Pexercise1()
{
  wcloseall();
  exercise1();
}
/
*************************************************************************/
/* this routine calls exercise2 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/
*************************************************************************/
void Pexercise2()
```

```
{
   wcloseall();
   exercise2();
}
/
*************************************************************************/
/* this routine calls exercise3 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/
*************************************************************************/
void Pexercise3()
{
   wcloseall();
   exercise3();
}
/
*************************************************************************/
/* this routine calls exercise4 routine whenever Pageup or               */
/* Page down keys are pressed.                                           */
/
*************************************************************************/
void Pexercise4()
{
   wcloseall();
   exercise4();
}
```

```
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
            GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/



char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;

        /*************************************************************
                    FUNCTION DEFINITIONS
        *************************************************************/

void wait(char title[]);
static void example_3_22 (void);
void confirm_graph_exit(void);
void normal_exit(void);
```

```
/*************************************************
                    MAIN PROGRAM
*************************************************/

main()
{
     cls (1);
     adapt = getAdapter();
    if (adapt == 'V')
    {
       setMode(0x12);
       cls(1);
    }
    if (adapt == 'E')
    {
       setMode(16);
       cls(1);
    }
    if (adapt == 'C')
    {
       setMode(4);
    setCGAPalette(0);
       cls(1);
    }
     }
   example_3_22();
   setMode(3);

}
```

```
/
*****************************************************************************/
static void example_3_22(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine (-100,70,-150,30,11);
    drawLine (-150,30,-150,-30,11);
    drawLine (-150,-30,-100,-70,11);
    drawLine (-100,-70,-50,-30,11);
    drawLine (-50,-30,-50,30,11);
    drawLine (-50,30,-100,70,11);
    drawLine (100,70,50,30,11);
    drawLine (50,30,50,-30,11);
    drawLine (50,-30,100,-70,11);
    drawLine (100,-70,150,-30,11);
    drawLine (150,-30,150,30,11);
    drawLine (150,30,100,70,11);

    aspect = 1.0;

    fillOval(-100,70,3,16,aspect);
    fillOval(-150,30,3,16,aspect);
    fillOval(-150,-30,3,16,aspect);
    fillOval(-100,-70,3,16,aspect);
    fillOval(-50,-30,3,16,aspect);
    fillOval(-50,30,3,16,aspect);
    fillOval(100,70,3,20,aspect);
    fillOval(50,30,3,18,aspect);
    fillOval(50,-30,3,20,aspect);
    fillOval(100,-70,3,18,aspect);
    fillOval(150,-30,3,20,aspect);
    fillOval(150,30,3,18,aspect);
```

```
write_horz_char (-110,85,65,31);  /* A */
write_horz_char (-160,30,66,31);  /* B */
write_horz_char (-160,-30,67,31); /* C */
write_horz_char (-110,-70,68,31); /* D */
write_horz_char (-45,-30,69,31);  /* E */
write_horz_char (-45,30,70,31);  /* F */

write_horz_char (90,85,65,20);  /* A */
write_horz_char (40,30,66,18);  /* B */
write_horz_char (40,-30,67,20); /* C */
write_horz_char (90,-70,68,18); /* D */
write_horz_char (155,-30,69,20);  /* E */
write_horz_char (155,30,70,18);  /* F */

write_horz_str(-120,-90,"Figure (a)",31);
write_horz_str(80,-90,"Figure (b)",31);

write_horz_str(-300,180,"The graph in Figure (a) has chromatic"
                        " number 2 since the vertices A, C",31);
write_horz_str(-300,160,"and E can be colored with one"
                        " color (red) and the other three vertices",31);
write_horz_str(-300,140,"with a second color (green)"
                        " as shown in Figure (b). In general, if",31);
write_horz_str(-300,120,"a cycle has even number of vertices"
                        ", then it can be colored using 2 colors.",31);

wait(" Example 3.22 ");

}
```

```
/
*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
     line = 29;
     else
     line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     getch();
     cls(0);
}
/
*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(-80,190," Please type y or n",31);
      ch = getch ();
    }
    switch (ch)        {
    case 'y': setMode(3);
          videoinit();
```

```
            normal_exit();
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

    case 'n':
            break;

    case 'N':
            break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}


/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor   */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.           */
/
*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*********************************************************
            GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *********************************************************/



char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;

        /***********************************************
         *                                             *
         *          FUNCTION DEFINITIONS               *
         *                                             *
         ***********************************************/

void wait(char title[]);
static void example_3_23 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
```

```
/*************************************************
                    MAIN PROGRAM
*************************************************/

main()
{
    cls (1);
    adapt = getAdapter();
    if (adapt == 'V')
    {
       setMode(0x12);
       cls(1);
    }
    if (adapt == 'E')
    {
       setMode(16);
       cls(1);
    }
    if (adapt == 'C')
    {
      setMode(4);
      setCGAPalette(0);
      cls(1);
    }
    }
   example_3_23();
   setMode(3);

}
/
*************************************************************************/
static void example_3_23(void)
{
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);
     drawLine (-100,70,-150,30,11);
     drawLine (-150,30,-150,-30,11);
     drawLine (-150,-30,-50,-30,11);
```

```
drawLine (-50,-30,-50,30,11);
drawLine (-50,30,-100,70,11);
aspect = 1.0;
fillOval(-100,70,3,16,aspect);
fillOval(-150,30,3,16,aspect);
fillOval(-150,-30,3,16,aspect);
fillOval(-50,-30,3,16,aspect);
fillOval(-50,30,3,16,aspect);

drawLine (100,70,50,30,11);
drawLine (50,30,50,-30,11);
drawLine (50,-30,150,-30,11);
drawLine (150,-30,150,30,11);
drawLine (150,30,100,70,11);
fillOval(100,70,3,20,aspect);
fillOval(50,30,3,18,aspect);
fillOval(50,-30,3,20,aspect);
fillOval(150,-30,3,18,aspect);
fillOval(150,30,3,30,aspect);

write_horz_char (-110,85,65,31);   /* A */
write_horz_char (-160,30,66,31);   /* B */
write_horz_char (-160,-30,67,31);  /* C */
write_horz_char (-45,-30,68,31);   /* D */
write_horz_char (-45,30,69,31);    /* E */

write_horz_char (90,85,65,20);     /* A */
write_horz_char (40,30,66,18);     /* B */
write_horz_char (40,-30,67,20);    /* C */
write_horz_char (155,-30,68,18);   /* D */
write_horz_char (155,30,69,30);    /* E */

write_horz_str(-120,-50,"Figure (a)",31);
write_horz_str(80,-50,"Figure (b)",31);

write_horz_str(-300,-70,"When a cycle has odd number of"
```

```
                                    " vertices, such as that in Figure (a)",31);
        write_horz_str(-300,-90,"then 3 colors must be used. If"
                                    " we try to alternate colors with the color",31);
        write_horz_str(-300,-110,"red assigned to vertices A"
                                    " and C, then it is not possible to use",31);
        write_horz_str(-300,-130,"either red or green for E. Using"
                                    " three colors to color a cycle with an odd",31);
        write_horz_str(-300,-150,"number of vertices is illustrated"
                                    " in the Figure above.",31);
        wait(" Example 3.23 ");
}

/
*************************************************************************/
void open_matrix_paren(int x1,int y1,int x2,int y2,int color)
{

            extern unsigned long int PATTERN;
            drawLine (x1 ,y1,x1+5,y1,color);
            drawLine (x1 ,y1,x1,y2,color);
            drawLine (x1 ,y2,x1+5,y2,color);
            drawLine (x2 ,y1,x2-5,y1,color);
            drawLine (x2 ,y1,x2,y2,color);
            drawLine (x2,y2,x2-5,y2,color);
}
/
*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
    line = 29;
    else
    line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
```

```
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
     getch();
     cls(0);
}
/
*****************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(-80,190," Please type y or n",31);
      ch = getch ();
    }
    switch (ch)         {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'Y': setMode(3);
               videoinit();
               normal_exit();
               break;

     case 'n':
               break;

     case 'N':
               break;
```

```
    default : break;    }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor    */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.            */
/
*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
            GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/

char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;


        /************************************************
                    FUNCTION DEFINITIONS
         ************************************************/

void wait(char title[]);
static void example_3_25 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
```

```
        /*************************************************
                        MAIN PROGRAM
        *************************************************/
main()
{
      cls (1);
      adapt = getAdapter();
       if (adapt == 'V')
          {
           setMode(0x12);
           cls(1);
          }
        if (adapt == 'E')
          {
           setMode(16);
           cls(1);
          }
        if (adapt == 'C')
          {
            setMode(4);
            setCGAPalette(0);
            cls(1);
          }
    example_3_25();
    setMode(3);
}
/
*************************************************************************/
static void example_3_25(void)
{
      LINEWIDTH = 3;
      cls(1);
      drawRect(-309,210,310,-200,11);

      drawLine (-100,70,-110,40,11);
      drawLine (-110,40,-150,20,11);
```

```
drawLine (-110,40,-110,0,11);
drawLine (-110,40,-70,20,11);
drawLine (-70,20,-70,-10,11);
drawLine (-110,0,-150,-40,11);
aspect = 1.0;
fillOval(-100,70,3,16,aspect);
fillOval(-110,40,3,16,aspect);
fillOval(-150,20,3,16,aspect);
fillOval(-110,0,3,16,aspect);
fillOval(-70,20,3,16,aspect);
fillOval(-70,-10,3,16,aspect);
fillOval(-150,-40,3,16,aspect);

drawLine (100,70,90,40,11);
drawLine (90,40,50,20,11);
drawLine (90,40,90,0,11);
drawLine (90,40,130,20,11);
drawLine (130,20,130,-10,11);
drawLine (90,0,50,-40,11);
fillOval(100,70,3,20,aspect);
fillOval(90,40,3,18,aspect);
fillOval(50,20,3,20,aspect);
fillOval(90,0,3,20,aspect);
fillOval(130,20,3,20,aspect);
fillOval(130,-10,3,18,aspect);
fillOval(50,-40,3,18,aspect);
write_horz_char (-110,85,65,31);   /* A */
write_horz_char (-120,55,66,31);   /* B */
write_horz_char (-160,20,67,31);   /* C */
write_horz_char (-65,20,68,31);    /* D */
write_horz_char (-65,-10,69,31);   /* E */
write_horz_char (-105,0,70,31);    /* F */
write_horz_char (-145,-40,71,31);  /* G */

write_horz_char (90,85,65,20);   /* A */
write_horz_char (80,55,66,18);   /* B */
```

```
    write_horz_char (40,20,67,20);    /* C */
    write_horz_char (135,20,68,20);   /* D */
    write_horz_char (135,-10,69,18);  /* E */
    write_horz_char (95,0,70,20);     /* F */
    write_horz_char (55,-40,71,18);   /* G */

    write_horz_str(-130,-70,"Figure (a)",31);
    write_horz_str(80,-70,"Figure (b)",31);

    write_horz_str(-300,160,"The graph in the following figure"
                            " can be colored with two colors as",31);
    write_horz_str(-300,140,"indicated in Figure (b).",31);
    wait(" Example 3.25 ");
}

/
*************************************************************************/
void open_matrix_paren(int x1,int y1,int x2,int y2,int color)
{

    extern unsigned long int PATTERN;
    drawLine (x1 ,y1,x1+5,y1,color);
    drawLine (x1 ,y1,x1,y2,color);
    drawLine (x1 ,y2,x1+5,y2,color);
    drawLine (x2 ,y1,x2-5,y1,color);
    drawLine (x2 ,y1,x2,y2,color);
    drawLine (x2,y2,x2-5,y2,color);
}
/
*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
       line = 29;
    else
      line = 24;
```

```
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    getch();
    cls(0);
}
/
*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(-80,190," Please type y or n",31);
     ch = getch ();
   }
   switch (ch)        {
   case 'y': setMode(3);
          videoinit();
          normal_exit();
          break;
   case 'Y': setMode(3);
          videoinit();
          normal_exit();
          break;
   case 'n':
          break;
   case 'N':
          break;
```

```
    default : break;     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor    */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.            */
/
*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*********************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *********************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;

        /************************************************
                FUNCTION DEFINITIONS
        ************************************************/

void wait(char title[]);
static void example_3_26 (void);
static void corfirm_graph_exit(void);
static void normal_exit(void);
```

```
        /************************************************
                        MAIN PROGRAM
         ************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
       if (adapt == 'V')
         {
            setMode(0x12);
            cls(1);
          }
       if (adapt == 'E')
           {
            setMode(16);
            cls(1);
           }
      if (adapt == 'C')
           {
            setMode(4);
            setCGAPalette(0);
            cls(1);
           }
     example_3_26();
     setMode(3);

}
/
*************************************************************************/
static void example_3_26(void)
{
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);
     drawLine(-150,100,150,100,11);
```

```
    drawLine(150,100,-150,0,11);
    drawLine(-150,0,150,0,11);
    drawLine(150,0,-150,-100,11);
    drawLine(-150,-100,150,-100,11);
    aspect = 1.0;
    fillOval(-150,100,3,16,aspect);
    fillOval(150,100,3,16,aspect);
    fillOval(-150,0,3,16,aspect);
    fillOval(150,0,3,16,aspect);
    fillOval(-150,-100,3,16,aspect);
    fillOval(150,-100,3,16,aspect);

    write_horz_char(-165,100,65,31);        /* A */
    write_horz_char(155,100,68,31);         /* D */
    write_horz_char(-165,0,66,31);          /* B */
    write_horz_char(155,0,69,31);           /* E */
    write_horz_char(-165,-100,67,31);       /* C */
    write_horz_char(155,-100,70,31);        /* F */

    write_horz_str(-300,180,"The graph in the following figure"
                            " has chromatic number 2 since the",31);
    write_horz_str(-300,160,"vertices on the left can be colored"
                            " with one color and the vertices",31);
    write_horz_str(-300,140,"on the right can be colored with a"
                            " second.",31);
    wait(" Example 3.26 ");
}


/
*************************************************************************/
void open_matrix_paren(int x1,int y1,int x2,int y2,int color)
 {
    extern unsigned long int PATTERN;
    drawLine (x1 ,y1,x1+5,y1,color);
    drawLine (x1 ,y1,x1,y2,color);
    drawLine (x1 ,y2,x1+5,y2,color);
    drawLine (x2 ,y1,x2-5,y1,color);
```

```
    drawLine (x2 ,y1,x2,y2,color);
    drawLine (x2,y2,x2-5,y2,color);
  }

/
*************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
       line = 29;
    else
       line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    getch();
    cls(0);
}
/
*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(-80,190," Please type y or n",31);
      ch = getch ();
    }
```

```
    switch (ch)         {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
     case 'Y': setMode(3);
               videoinit();
               normal_exit();
               break;

     case 'n':
               break;

     case 'N':
               break;

     default : break;
     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}


/
*************************************************************************/
/* this function handles normal termination.  The original screen and cursor    */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.            */
/
*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
```

```
    exit(0);
}
```

```
/* PROGRAM  : thm3_6.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains a theorem about the "Coloring a
Graph".

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*********************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *********************************************************/


char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*****************************************************
                        FUNCTION DEFINITIONS
        *****************************************************/
void wait(char title[]);
static void thm_3_6 (void);
static void thm_3_6cont (void);
static void thm_3_6cont2 (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*****************************************************
                        MAIN PROGRAM
        *****************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
       {
          setMode(0x12);
          cls(1);
       }
    if (adapt == 'E')
      {
          setMode(16);
          cls(1);
      }
   if (adapt == 'C')
     {
          setMode(4);
          setCGAPalette(0);
          cls(1);
     }
   thm_3_6();
   setMode(3);
}
```

```
/*************************************************************************/
static void thm_3_6(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the thm_3_6 function */
  setonkey(0x4900,thm_3_6,0);
  /*************************************************************************/
  /* attach [Pagedown] to the thm_3_6cont function */
  setonkey(0x5100,thm_3_6cont,0);
  /*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    drawLine(0,0,-20,-50,11);
    drawLine(-20,-50,-50,-100,11);
    drawLine(-20,-50,0,-100,11);
    drawLine(0,0,30,-50,11);
    drawLine(30,-50,50,-100,11);

    aspect = 1.9;
    LINEWIDTH = 1;
    drawArc(0,-100,100,900,1100,11,aspect);
    drawArc(0,-100,100,1200,1350,11,aspect);
    drawArc(0,-100,100,1420,1520,11,aspect);
    drawArc(0,-100,100,1580,1620,11,aspect);
    drawArc(0,-100,100,1715,1720,11,aspect);

    aspect = 1.0;
    fillOval(0,0,3,20,aspect);
    fillOval(-20,-50,3,18,aspect);
    fillOval(-50,-100,3,20,aspect);
    fillOval(0,-100,3,20,aspect);
    fillOval(30,-50,3,18,aspect);
    fillOval(50,-100,3,20,aspect);
    write_horz_char(0,20,86,31);          /* V */
    write_horz_char(-55,-105,87,31);   /* W */
```

```
    write_horz_str(10,0,"red",20);
    write_horz_str(-15,-50,"green",18);
    write_horz_str(-45,-100,"red",20);
    write_horz_str(5,-100,"red",20);
    write_horz_str(35,-50,"green",18);
    write_horz_str(55,-100,"red",20);
    write_horz_char(-60,-50,63,31);      /* ? */
    write_horz_str(-300,190,"A graph G has no cycle of odd length"
                            " if and only if G can be colored",31);
    write_horz_str(-300,170,"with two colors.",31);
    write_horz_str(-300,150,"Proof.",21);
    write_horz_str(-250,150,"Let's assume that G has no cycles of"
                            " odd length. A vertex V is",31);
    write_horz_str(-300,130,"selected and the color red is assigned to it."
                            " Then to each vertex adjacent",31);
    write_horz_str(-300,110,"to V the color green is assigned. Now the"
                            " vertices adjacent to the recently",31);
    write_horz_str(-300,90,"colored green vertices are assigned the"
                           " color red. Can one of these red",31);
    write_horz_str(-300,70,"vertices, say W, be adjacent to the vertex"
                           " V, which also has color red?",31);
    write_horz_str(-300,50,"The diagram in the following Figure"
                           " illustrates this situation.",31);
    wait("");
    thm_3_6cont();
}
```

```
/*************************************************************************/
static void thm_3_6cont(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the thm_3_6 function */
  setonkey(0x4900,thm_3_6,0);
  /*************************************************************************/
  /* attach [Pagedown] to the thm_3_6cont2 function */
  setonkey(0x5100,thm_3_6cont2,0);
  /*************************************************************************/

    LINEWIDTH = 3;
    cls(1);

    drawRect(-309,210,310,-200,11);
    drawLine(0,0,-20,-50,11);
    drawLine(-20,-50,-50,-100,11);
    drawLine(-20,-50,0,-100,11);
    drawLine(0,0,30,-50,11);
    drawLine(30,-50,50,-100,11);
    drawLine(-50,-100,-100,-150,11);
    drawLine(-50,-100,0,-150,11);
    drawLine(0,-100,50,-150,11);
    drawLine(50,-100,100,-150,11);

    aspect = 1.0;
    fillOval(0,0,3,20,aspect);
    fillOval(-20,-50,3,18,aspect);
    fillOval(-50,-100,3,20,aspect);
    fillOval(0,-100,3,20,aspect);
    fillOval(30,-50,3,18,aspect);
    fillOval(50,-100,3,20,aspect);
    fillOval(-100,-150,3,18,aspect);
    fillOval(0,-150,3,18,aspect);
    fillOval(50,-150,3,18,aspect);
    fillOval(100,-150,3,18,aspect);
```

```
write_horz_char(10,15,86,31);      /* V */
write_horz_str(10,0,"red",20);
write_horz_str(-15,-50,"green",18);
write_horz_str(-90,-100,"red",20);
write_horz_str(-30,-100,"red",20);
write_horz_str(35,-50,"green",18);
write_horz_str(65,-95,"red",20);
write_horz_str(-110,-160,"green",18);
write_horz_str(-10,-160,"green",18);
write_horz_str(40,-160,"green",18);
write_horz_str(90,-160,"green",18);

write_horz_str(-300,190,"We see that if V and W were adjacent"
                        ", then there would be a cycle of length",31);
write_horz_str(-300,170,"3. In a similar fashion, any"
                        " other vertex just colored red is not adjacent",31);
write_horz_str(-300,150,"to any other with color"
                        " red, for otherwise there would be a cycle of odd",31);
write_horz_str(-300,130,"length. Next the vertices which are"
                        " adjacent to those just colored red are",31);
write_horz_str(-300,110,"assigned the color green. Again"
                        " if any two vertices colored green were",31);
write_horz_str(-300,90,"adjacent, then there would be a"
                        " cycle of odd length. Continue by coloring",31);
write_horz_str(-300,70,"red the vertices adjacent those"
                        " colored green vertices. As before none of",31);
write_horz_str(-300,50,"these newly colored vertices can be"
                        " adjacent to a previously colored red",31);
write_horz_str(-300,30,"vertex. This process is repeated"
                        " until there are no uncolored vertices",31);
write_horz_str(-300,10,"adjacent to color vertices.",31);
wait("");
thm_3_6cont2();


}
```

```
/*************************************************************************/
static void thm_3_6cont2(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the thm_3_6cont function */
  setonkey(0x4900,thm_3_6cont,0);
  /*************************************************************************/
  /* attach [Pagedown] to the normal_exit function */
  setonkey(0x5100,normal_exit,0);
  /*************************************************************************/

    LINEWIDTH = 3;
    cls(1);

    drawRect(-309,210,310,-200,11);
    drawLine(0,0,-20,-50,11);
    drawLine(-20,-50,-50,-100,11);
    drawLine(-20,-50,0,-100,11);
    drawLine(0,0,30,-50,11);
    drawLine(30,-50,50,-100,11);
    drawLine(-50,-100,-100,-150,11);
    drawLine(-50,-100,0,-150,11);
    drawLine(0,-100,50,-150,11);
    drawLine(50,-100,100,-150,11);

    aspect = 1.0;
    fillOval(0,0,3,20,aspect);
    fillOval(-20,-50,3,18,aspect);
    fillOval(-50,-100,3,20,aspect);
    fillOval(0,-100,3,20,aspect);
    fillOval(30,-50,3,18,aspect);
    fillOval(50,-100,3,20,aspect);
    fillOval(-100,-150,3,18,aspect);
    fillOval(0,-150,3,18,aspect);
    fillOval(50,-150,3,18,aspect);
    fillOval(100,-150,3,18,aspect);
```

```
    write_horz_char(10,15,86,31);      /* V */
    write_horz_str(10,0,"red",20);
    write_horz_str(-15,-50,"green",18);
    write_horz_str(-90,-100,"red",20);
    write_horz_str(-30,-100,"red",20);
    write_horz_str(35,-50,"green",18);
    write_horz_str(65,-95,"red",20);
    write_horz_str(-110,-160,"green",18);
    write_horz_str(-10,-160,"green",18);
    write_horz_str(40,-160,"green",18);
    write_horz_str(90,-160,"green",18);

    write_horz_str(-300,170,"If the graph is not connected"
                            ", then there will be vertices",31);
    write_horz_str(-300,150,"that are not adjacent to any of the"
                            " colored vertices and hence are",31);
    write_horz_str(-300,130,"uncolored. For these vertices the"
                            " above process is repeated again",31);
    write_horz_str(-300,110,"using the colors red and blue."
                            " Eventually all the vertices can be",31);
    write_horz_str(-300,90,"colored with two colors.",31);
    wait("");
    normal_exit();
}
/*****************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
      line = 29;
    else
      line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
```

```
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
    cls(0);
}
/*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
      write_horz_str(30,-210,"                ",31);
  }
   switch (ch)        {
   case 'y': normal_exit();
          break;
   case 'Y': normal_exit();
          break;
   case 'n':write_horz_str(-300,-210,"                          ",31);
          break;
   case 'N':write_horz_str(-300,-210,"                          ",31);
          break;
   default : break;
   }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
   setMode(3);
   videoinit();
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*********************************************************
            GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *********************************************************/



char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;


        /************************************************
                    FUNCTION DEFINITIONS
         ************************************************/

void wait(char title[]);
static void example_3_27 (void);
static void confirm_graph_exit(void);
static void normal_exit (void);
```

```
        /************************************************
                        MAIN PROGRAM
         ************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
       {
         setMode(0x12);
         cls(1);
       }
     if (adapt == 'E')
        {
          setMode(16);
          cls(1);
        }
     if (adapt == 'C')
       {
         setMode(4);
         setCGAPalette(0);
         cls(1);
       }
    example_3_27();
    setMode(3);
}
/*****************************************************************************/

static void example_3_27(void)
{
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);

     drawLine(-150,0,-50,0,11);
     drawLine(-150,0,-150,-80,11);
     drawLine(-100,0,-100,-80,11);
     drawLine(-100,-80,-50,-80,11);
     drawLine(-100,-40,-50,-40,11);
```

```
drawLine(-50,0,-50,-80,11);
aspect = 1.0;

fillOval(-150,0,3,16,aspect);
fillOval(-50,0,3,16,aspect);
fillOval(-150,-80,3,16,aspect);
fillOval(-100,0,3,16,aspect);
fillOval(-100,-80,3,16,aspect);
fillOval(-50,-80,3,16,aspect);
fillOval(-100,-40,3,16,aspect);
fillOval(-50,-40,3,16,aspect);

drawLine(50,0,150,0,11);
drawLine(50,0,50,-80,11);
drawLine(100,0,100,-80,11);
drawLine(100,-80,150,-80,11);
drawLine(100,-40,150,-40,11);
drawLine(150,0,150,-80,11);

fillOval(50,0,3,18,aspect);
fillOval(150,0,3,18,aspect);
fillOval(50,-80,3,20,aspect);
fillOval(100,0,3,20,aspect);
fillOval(100,-80,3,20,aspect);
fillOval(150,-80,3,18,aspect);
fillOval(100,-40,3,18,aspect);
fillOval(150,-40,3,20,aspect);

write_horz_char(-45,0,65,31);        /* A */
write_horz_char(-110,-40,66,31);     /* D */
write_horz_char(-110,-80,68,31);     /* B */
write_horz_char(-160,-80,69,31);     /* E */
write_horz_char(-45,-40,67,31);      /* C */
write_horz_char(-160,0,70,31);       /* F */
write_horz_char(-45,-80,71,31);      /* G */
write_horz_char(-100,15,86,31);      /* V */
```

```
    write_horz_char(155,0,65,18);       /* A */
    write_horz_char(90,-40,66,18);      /* D */
    write_horz_char(90,-80,68,20);      /* B */
    write_horz_char(40,-80,69,20);      /* E */
    write_horz_char(155,-40,67,20);     /* C */
    write_horz_char(40,0,70,18);        /* F */
    write_horz_char(155,-80,71,18);     /* G */
    write_horz_char(100,15,86,20);      /* V */

    write_horz_str(-300,180,"The coloring process is started by"
                            " selecting a vertex V and coloring",31);
    write_horz_str(-300,160,"it red. Then since, F, B, and A are"
                            " the vertices adjacent to V, the",31);
    write_horz_str(-300,140,"color green is assigned to them. The"
                            " uncolored vertices adjacent to the",31);
    write_horz_str(-300,120,"green colored vertices are C, D, and E"
                            ", and so the color red is assigned",31);
    write_horz_str(-300,100,"to them. Finally the vertex G is an"
                            " uncolored vertex adjacent to red",31);
    write_horz_str(-300,80,"vertices and so is assigned the color"
                           " green.",31);
    wait(" Example 3.27 ");
}

/*************************************************************************/

void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
       line = 29;
    else
       line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
```

```
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    getch();
    cls(0);
}
/*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     write_horz_str(-80,190," Please type y or n",31);
     ch = getch ();
   }
   switch (ch)        {
    case 'y': setMode(3);
              videoinit();
              normal_exit();
              break;
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;
    case 'n':
              break;
    case 'N':
              break;
    default : break;    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
```

```
}
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.         */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*********************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *********************************************************/



char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;


        /************************************************
                    FUNCTION DEFINITIONS
         ************************************************/

void wait(char title[]);
static void example_3_28 (void);
static void confirm_graph_exit(void);
static void normal_exit (void);
```

```
        /*************************************************
                        MAIN PROGRAM
         *************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
       if (adapt == 'V')
          {
            setMode(0x12);
             cls(1);
          }
       if (adapt == 'E')
           {
             setMode(16);
             cls(1);
           }
       if (adapt == 'C')
          {
           setMode(4);
           setCGAPalette(0);
           cls(1);
          }
    example_3_28();
    setMode(3);
}
/*************************************************************************/
static void example_3_28(void)
{
      LINEWIDTH = 3;
      cls(1);
      drawRect(-309,210,310,-200,11);
      drawLine(0,0,-150,-100,11);
      drawLine(0,0,-50,-100,11);
      drawLine(0,0,50,-100,11);
```

```
    drawLine(0.0,150,-100,11);
    aspect = 1.0;
    fillOval(0,0,3,16,aspect);
    fillOval(-150,-100,3,16,aspect);
    fillOval(-50,-100,3,16,aspect);
    fillOval(50,-100,3,16,aspect);
    fillOval(150,-100,3,16,aspect);

    write_horz_str(-300,180,"The procedure described in theorem"
                            " 3.7 may use more colors than are",31);
    write_horz_str(-300,160,"really necessary. The graph in the"
                            " following Figure has a vertex of",31);
    write_horz_str(-300,140,"degree 4, which is the maximum degree"
                            ", and so by theorem 3.7 can be",31);
    write_horz_str(-300,120,"colored using 1 + 4 = 5 colors."
                            " However, we have already seen that",31);
    write_horz_str(-300,100,"it can be colored using two colors.",31);
    wait(" Example 3.28 ");
}


/*************************************************************************/

void open_matrix_paren(int x1,int y1,int x2,int y2,int color)
{

    extern unsigned long int PATTERN;
    drawLine (x1 ,y1,x1+5,y1,color);
    drawLine (x1 ,y1,x1,y2,color);
    drawLine (x1 ,y2,x1+5,y2,color);
    drawLine (x2 ,y1,x2-5,y1,color);
    drawLine (x2 ,y1,x2,y2,color);
    drawLine (x2,y2,x2-5,y2,color);
}
```

```
/*************************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
      if ((mode == 0x11) || (mode == 0x12))
          line = 29;
      else
           line = 24;
       tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      getch();
      cls(0);
}
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-80,210,"Quit process, are you sure (y/n)?",31);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       write_horz_str(-80,190," Please type y or n",31);
       ch = getch ();
    }
    switch (ch)         {
     case 'y': setMode(3);
               videoinit();
               normal_exit();
               break;
```

```
    case 'Y': setMode(3);
              videoinit();
              normal_exit();
              break;

    case 'n':
              break;

    case 'N':
              break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"


        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *************************************************************/
char adapt;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;


        /*************************************************************
                    FUNCTION DEFINITIONS
         *************************************************************/
void wait(char title[]);
static void example_3_29 (void);
static void ex3_29cont  (void);
static void ex_3_29cont2  (void);
static void ex_3_29cont3  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);
```

```
        /************************************************
                        MAIN PROGRAM
         ************************************************/
main()
{
      cls (1);
      adapt = getAdapter();
       if (adapt == 'V')
          {
            setMode(0x12);
            cls(1);
           }
       if (adapt == 'E')
           {
            setMode(16);
            cls(1);
           }
      if (adapt == 'C')
          {
            setMode(4);
             setCGAPalette(0);
             cls(1);
          }
    example_3_29();
    setMode(3);
}
/*************************************************************************/
static void example_3_29(void)
{
/*************************************************************************/
   /* attach [Pageup] to the example_3_29 function */
   setonkey(0x4900,example_3_29,0);
/*************************************************************************/
   /* attach [Pagedown] to the ex3_29cont function */
   setonkey(0x5100,ex3_29cont,0);
/*************************************************************************/
```

```
LINEWIDTH = 3;
cls(1);
drawRect(-309,210,310,-200,11);

drawLine(-100,-50,-150,-100,11);     /* AE */
drawLine(-150,-100,-150,-150,11);  /* ED */
drawLine(-150,-150,-50,-150,11);  /* DC */
drawLine(-50,-150,-40,-100,11);    /* CB */
drawLine(-40,-100,-100,-50,11);      /* BA */
drawLine(-100,-50,-100,-125,11);     /* AG */
drawLine(-100,-100,-150,-100,11);   /* FE */
drawLine(-100,-100,-150,-150,11);  /* FD */

aspect = 1.0;
fillOval(-100,-50,3,16,aspect);
fillOval(-150,-100,3,16,aspect);
fillOval(-150,-150,3,16,aspect);
fillOval(-50,-150,3,16,aspect);
fillOval(-40,-100,3,16,aspect);
fillOval(-100,-125,3,16,aspect);
fillOval(-100,-100,3,16,aspect);

write_horz_str(-130,-165,"Figure (a)",31);

write_horz_char(-100,-30,65,31);   /* A */
write_horz_char(-35,-100,66,31);   /* B */
write_horz_char(-45,-150,67,31);   /* C */
write_horz_char(-165,-150,68,31);  /* D */
write_horz_char(-165,-100,69,31);  /* E */
write_horz_char(-95,-100,70,31);   /* F */
write_horz_char(-95,-125,71,31);   /* G */

drawLine(100,-50,50,-100,11);        /* AE */
drawLine(50,-100,50,-150,11);        /* ED */
drawLine(50,-150,150,-150,11);       /* DC */
drawLine(150,-150,160,-100,11);     /* CB */
```

```
drawLine(160,-100,100,-50,11);     /* BA */
drawLine(100,-50,100,-125,11);     /* AG */
drawLine(100,-100,50,-100,11);     /* FE */
drawLine(100,-100,50,-150,11);     /* FD */

fillOval(100,-50,3,16,aspect);
fillOval(50,-100,3,16,aspect);
fillOval(50,-150,3,16,aspect);
fillOval(150,-150,3,16,aspect);
fillOval(160,-100,3,16,aspect);
fillOval(100,-125,3,16,aspect);
fillOval(100,-100,3,16,aspect);

write_horz_char(100,-30,86,31);    /* V */
write_horz_char(107,-35,50,31);    /* 2 */
write_horz_char(165,-100,86,31);   /* V */
write_horz_char(172,-105,53,31);   /* 5 */
write_horz_char(155,-150,86,31);   /* V */
write_horz_char(162,-155,54,31);   /* 6 */
write_horz_char(35,-150,86,31);    /* V */
write_horz_char(42,-155,52,31);    /* 4 */
write_horz_char(32,-100,86,31);    /* V */
write_horz_char(39,-105,51,31);    /* 3 */
write_horz_char(105,-100,86,31);   /* V */
write_horz_char(112,-105,49,31);   /* 1 */
write_horz_char(105,-125,86,31);   /* V */
write_horz_char(112,-130,55,31);   /* 7 */
write_horz_str(70,-165,"Figure (b)",31);

write_horz_str(-300,200,"For the graph in Figure (a) vertex"
                        " F has the largest degree, namely, 4,",31);
write_horz_str(-300,180,"and so F is given the label V1. Vertices"
                        " A, D, and E have degree 3 and so",31);
write_horz_str(-300,160,"are assigned labels V2, V3, and V4 in"
                        " some random manner. Likewise,",31);
write_horz_str(-300,140,"vertices B and C with degree 2 are"
```

```
                                " assigned labels V5 and V6. Vertex G",31);
    write_horz_str(-300,120,"is the only remaining vertex and is"
                                " assigned V7. This is shown in Figure (b).",31);
    write_horz_str(-300,100,"   The adjacency list form of"
                                " representation is very convenient to use with",31);
    write_horz_str(-300,80,"the Welsh and Powell algorithm."
                                " For the graph in Figure (b), the adjacency",31);
    write_horz_str(-300,60,"list representation is given next.",31);
    wait("");
    ex3_29cont();
}
/*************************************************************************/
static void ex3_29cont(void)

{

  /*************************************************************************/
  /* attach [Pageup] to the example_3_29 function */
  setonkey(0x4900,example_3_29,0);
  /*************************************************************************/
  /* attach [Pagedown] to the ex3_29cont function */
  setonkey(0x5100,ex_3_29cont2,0);
  /*************************************************************************/
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);

    drawLine(-100,-50,-150,-100,11);    /* AE */
    drawLine(-150,-100,-150,-150,11);   /* ED */
    drawLine(-150,-150,-50,-150,11);    /* DC */
    drawLine(-50,-150,-40,-100,11);     /* CB */
    drawLine(-40,-100,-100,-50,11);     /* BA */
    drawLine(-100,-50,-100,-125,11);    /* AG */
    drawLine(-100,-100,-150,-100,11);   /* FE */
    drawLine(-100,-100,-150,-150,11);   /* FD */
```

```
aspect = 1.0;
fillOval(-100,-50,3,16,aspect);
fillOval(-150,-100,3,16,aspect);
fillOval(-150,-150,3,16,aspect);
fillOval(-50,-150,3,16,aspect);
fillOval(-40,-100,3,16,aspect);
fillOval(-100,-125,3,16,aspect);
fillOval(-100,-100,3,16,aspect);

write_horz_str(-130,-165,"Figure (a)",31);

write_horz_char(-100,-30,65,31);   /* A */
write_horz_char(-35,-100,66,31);   /* B */
write_horz_char(-45,-150,67,31);   /* C */
write_horz_char(-165,-150,68,31);  /* D */
write_horz_char(-165,-100,69,31);  /* E */
write_horz_char(-95,-100,70,31);   /* F */
write_horz_char(-95,-125,71,31),   /* G */

drawLine(100,-50,50,-100,11);      /* AE */
drawLine(50,-100,50,-150,11);      /* ED */
drawLine(50,-150,150,-150,11);     /* DC */
drawLine(150,-150,160,-100,11);    /* CB */
drawLine(160,-100,100,-50,11);     /* BA */
drawLine(100,-50,100,-125,11);     /* AG */
drawLine(100,-100,50,-100,11);     /* FE */
drawLine(100,-100,50,-150,11);     /* FD */

fillOval(100,-50,3,16,aspect);
fillOval(50,-100,3,16,aspect);
fillOval(50,-150,3,16,aspect);
fillOval(150,-150,3,16,aspect);
fillOval(160,-100,3,16,aspect);
fillOval(100,-125,3,16,aspect);
fillOval(100,-100,3,16,a
```

```
write_horz_char(100,-30,86,31);    /* V */
write_horz_char(107,-35,50,31);    /* 2 */
write_horz_char(165,-100,86,31);   /* V */
write_horz_char(172,-105,53,31);   /* 5 */
write_horz_char(155,-150,86,31);   /* V */
write_horz_char(162,-155,54,31);   /* 6 */
write_horz_char(35,-150,86,31);    /* V */
write_horz_char(42,-155,52,31);    /* 4 */
write_horz_char(32,-100,86,31);    /* V */
write_horz_char(39,-105,51,31);    /* 3 */
write_horz_char(105,-100,86,31);   /* V */
write_horz_char(112,-105,49,31);   /* 1 */
write_horz_char(105,-125,86,31);   /* V */
write_horz_char(112,-130,55,31);   /* 7 */

write_horz_str(70,-165,"Figure (b)",31);

write_horz_str(-50,150,"V :V ,V ,V ,V",11); /**********************/
write_horz_char(-43,145,49,11);             /*  V1:V2,V3,V4,V7      */
write_horz_char(-18,145,50,11);             /*                      */
write_horz_char(7,145,51,11);               /**********************/
write_horz_char(32,145,52,11);
write_horz_char(57,145,55,11);

write_horz_str(-50,130,"V :V ,V ,V",11); /*******************/
write_horz_char(-43,125,50,11);          /*  V2:V1,V3,V5    */
write_horz_char(-18,125,49,11);          /*                 */
write_horz_char(7,125,51,11);            /*******************/
write_horz_char(32,125,53,11);

write_horz_str(-50,110,"V :V ,V ,V",11); /****************/
write_horz_char(-43,105,51,11);          /*  V3:V1,V2,V4  */
write_horz_char(-18,105,49,11);          /*               */
write_horz_char(7,105,50,11);            /*               */
write_horz_char(32,105,52,11);           /****************/
```

```
    write_horz_str(-50,90,"V :V ,V ,V",11); /****************/
    write_horz_char(-43,85,52,11);          /* V4:V1,V3,V6  */
    write_horz_char(-18,85,49,11);          /*              */
    write_horz_char(7,85,51,11);            /****************/
    write_horz_char(32,85,54,11);

    write_horz_str(-50,70,"V :V ,V",11);   /****************/
    write_horz_char(-43,65,53,11);          /* V5:V2,V6     */
    write_horz_char(-18,65,50,11);          /*              */
    write_horz_char(7,65,54,11);            /****************/

    write_horz_str(-50,50,"V :V ,V",11);    /********************/
    write_horz_char(-43,45,54,11);          /* V6:V4,V5         */
    write_horz_char(-18,45,52,11);          /********************/
    write_horz_char(7,45,53,11);

    write_horz_str(-50,30,"V :V",11);      /********************/
    write_horz_char(-43,25,55,11);          /* V7:V1            */
    write_horz_char(-18,25,49,11);         /********************/
    wait("");
    ex_3_29cont2();
}
/*****************************************************************************/

static void ex_3_29cont2(void)
{
  /*****************************************************************************/
   /* attach [Pageup] to the ex3_29cont function */
   setonkey(0x4900,ex3_29cont,0);
  /*****************************************************************************/
   /* attach [Pagedown] to the ex_3_29cont3 function */
   setonkey(0x5100,ex_3_29cont3,0);
  /*****************************************************************************/
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);
}
```

```
drawLine(-100,-50,-150,-100,11);    /* AE */
drawLine(-150,-100,-150,-150,11);   /* ED */
drawLine(-150,-150,-50,-150,11);    /* DC */
drawLine(-50,-150,-40,-100,11);     /* CB */
drawLine(-40,-100,-100,-50,11);     /* BA */
drawLine(-100,-50,-100,-125,11);    /* AG */
drawLine(-100,-100,-150,-100,11);   /* FE */
drawLine(-100,-100,-150,-150,11);   /* FD */

aspect = 1.0;
fillOval(-100,-50,3,16,aspect);
fillOval(-150,-100,3,16,aspect);
fillOval(-150,-150,3,16,aspect);
fillOval(-50,-150,3,16,aspect);
fillOval(-40,-100,3,16,aspect);
fillOval(-100,-125,3,16,aspect);
fillOval(-100,-100,3,16,aspect);
write_horz_str(-130,-165,"Figure (a)",31);

write_horz_char(-100,-30,65,31);    /* A */
write_horz_char(-35,-100,66,31);    /* B */
write_horz_char(-45,-150,67,31);    /* C */
write_horz_char(-165,-150,68,31);   /* D */
write_horz_char(-165,-100,69,31);   /* E */
write_horz_char(-95,-100,70,31);    /* F */
write_horz_char(-95,-125,71,31);    /* G */

drawLine(100,-50,50,-100,11);       /* AE */
drawLine(50,-100,50,-150,11);       /* ED */
drawLine(50,-150,150,-150,11);      /* DC */
drawLine(150,-150,160,-100,11);     /* CB */
drawLine(160,-100,100,-50,11);      /* BA */
drawLine(100,-50,100,-125,11);      /* AG */
drawLine(100,-100,50,-100,11);      /* FE */
drawLine(100,-100,50,-150,11);      /* FD */
```

```
fillOval(100,-50,3,16,aspect);
fillOval(50,-100,3,16,aspect);
fillOval(50,-150,3,16,aspect);
fillOval(150,-150,3,16,aspect);
fillOval(160,-100,3,16,aspect);
fillOval(100,-125,3,16,aspect);
fillOval(100,-100,3,16,aspect);

write_horz_char(100,-30,86,31);   /* V */
write_horz_char(107,-35,50,31);   /* 2 */
write_horz_char(165,-100,86,31);  /* V */
write_horz_char(172,-105,53,31);  /* 5 */
write_horz_char(155,-150,86,31);  /* V */
write_horz_char(162,-155,54,31);  /* 6 */
write_horz_char(35,-150,86,31);   /* V */
write_horz_char(42,-155,52,31);   /* 4 */
write_horz_char(32,-100,86,31);   /* V */
write_horz_char(39,-105,51,31);   /* 3 */
write_horz_char(105,-100,86,31);  /* V */
write_horz_char(112,-105,49,31);  /* 1 */
write_horz_char(105,-125,86,31);  /* V */
write_horz_char(112,-130,55,31);  /* 7 */
write_horz_str(70,-165,"Figure (b)",31);

write_horz_str(-300,200,"In the Welsh and Powell algorithm"
                        " the first uncolored vertex in the list",31);
write_horz_str(-300,180,"V1, is given the color red. Then we go"
                        " down the list looking for the",31);
write_horz_str(-300,160,"next vertex which is not adjacent to V1"
                        ", that is, for the lowest",31);
write_horz_str(-300,140,"numbered vertex that is not in the list"
                        " folowing V1. This is the vertex V5",31);
write_horz_str(-300,120,"which is assigned the color red. We"
                        " continue down the list looking for the",31);
write_horz_str(-300,100,"next vertex which is adjacent to neither"
                        " V1 nor V5. Since there are none, we",31);
```

```
    write_horz_str(-300,80,"go back to the top of the list and find"
                           " the first uncolored vertex, which is",31);
    write_horz_str(-300,60,"V2, and assign it the color green. Then"
                           " the next uncolored vertex is found",31);
    write_horz_str(-300,40,"which is not adjacent to V2. This"
                           " is V4 and it is given the color",31);
    write_horz_str(-300,20,"green. Continuing in this same manner"
                           " we find that V7 is the next",31);
    write_horz_str(-300,0,"uncolored vertex adjacent to neither"
                          " V2 nor V4, and so V7 is assigned",31);
    wait("");
    ex_3_29cont3();
}

/*************************************************************************/
static void ex_3_29cont3(void)
{
  /*************************************************************************/
   /* attach [Pageup] to the ex_3_29cont2 function */
   setonkey(0x4900,ex_3_29cont2,0);
  /*************************************************************************/
   /* attach [Pagedown] to the normal_exit function */
   setonkey(0x5100,normal_exit,0);
  /*************************************************************************/
     LINEWIDTH = 3;
     cls(1);
     drawRect(-309,210,310,-200,11);

     drawLine(-100,-50,-150,-100,11);    /* AE */
     drawLine(-150,-100,-150,-150,11);   /* ED */
     drawLine(-150,-150,-50,-150,11);    /* DC */
     drawLine(-50,-150,-40,-100,11);     /* CB */
     drawLine(-40,-100,-100,-50,11);     /* BA */
     drawLine(-100,-50,-100,-125,11);    /* AG */
     drawLine(-100,-100,-150,-100,11);   /* FE */
     drawLine(-100,-100,-150,-150,11);   /* FD */
```

```
aspect = 1.0;
fillOval(-100,-50,3,16,aspect);
fillOval(-150,-100,3,16,aspect);
fillOval(-150,-150,3,16,aspect);
fillOval(-50,-150,3,16,aspect);
fillOval(-40,-100,3,16,aspect);
fillOval(-100,-125,3,16,aspect);
fillOval(-100,-100,3,16,aspect);

write_horz_str(-130,-165,"Figure (a)",31);

write_horz_char(-100,-30,65,31);   /* A */
write_horz_char(-35,-100,66,31);   /* B */
write_horz_char(-45,-150,67,31);   /* C */
write_horz_char(-165,-150,68,31);  /* D */
write_horz_char(-165,-100,69,31);  /* E */
write_horz_char(-95,-100,70,31);   /* F */
write_horz_char(-95,-125,71,31);   /* G */

drawLine(100,-50,50,-100,11);      /* AE */
drawLine(50,-100,50,-150,11);      /* ED */
drawLine(50,-150,150,-150,11);     /* DC */
drawLine(150,-150,160,-100,11);    /* CB */
drawLine(160,-100,100,-50,11);     /* BA */
drawLine(100,-50,100,-125,11);     /* AG */
drawLine(100,-100,50,-100,11);     /* FE */
drawLine(100,-100,50,-150,11);     /* FD */

fillOval(100,-50,3,16,aspect);
fillOval(50,-100,3,16,aspect);
fillOval(50,-150,3,16,aspect);
fillOval(150,-150,3,16,aspect);
fillOval(160,-100,3,16,aspect);
fillOval(100,-125,3,16,aspect);
fillOval(100,-100,3,16,aspect);
```

```
    write_horz_char(100,-30,86,31);   /* V */
    write_horz_char(107,-35,50,31);   /* 2 */
    write_horz_char(165,-100,86,31);  /* V */
    write_horz_char(172,-105,53,31);  /* 5 */
    write_horz_char(155,-150,86,31);  /* V */
    write_horz_char(162,-155,54,31);  /* 6 */
    write_horz_char(35,-150,86,31);   /* V */
    write_horz_char(42,-155,52,31);   /* 4 */
    write_horz_char(30,-100,86,31);   /* V */
    write_horz_char(39,-105,51,31);   /* 3 */
    write_horz_char(105,-100,86,31);  /* V */
    write_horz_char(112,-105,49,31);  /* 1 */
    write_horz_char(105,-125,86,31);  /* V */
    write_horz_char(112,-130,55,31);  /* 7 */
    write_horz_str(70,-165,"Figure (b)",31);

    write_horz_str(-300,180,"the color green. Since all the uncolored"
                            " vertices are adjacent to green",31);
    write_horz_str(-300,160,"vertices, we go back to the top to"
                            " find the next uncolored vertex, which",31);
    write_horz_str(-300,140,"is V3, and assign the color yellow."
                            " Since V6 is uncolored and not",31);
    write_horz_str(-300,120,"adjacent to V3, it is assigned the"
                            " color yellow. So the",31);
    write_horz_str(-300,100,"graph can be colored with 3 colors.",31);
    normal_exit();

}
```

```
/*****************************************************************/
void wait(char title[])
{
      int tab,width,line,mode;
      mode = getMode(&width);
       if ((mode == 0x11) || (mode == 0x12))
            line = 29;
       else
            line = 24;
      tab = (width - strlen(title))/2;
      gotoxy(tab,0);
      writString(title,WHITE,0);
      tab = (width - 33)/2;
      gotoxy(tab,line);
      writString("Press any key to continue ...", WHITE,0);
      if(waitkey()==ESC) confirm_graph_exit();
      cls(0);
}
/*****************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=ch_onkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit example, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
        write_horz_str(30,-210," Please type y or n",79);
       ch = getch ();
       write_horz_str(30,-210,"                ",31);
    }
    switch (ch)         {
     case 'y': normal_exit();
            break;
     case 'Y': normal_exit();
```

```
/* PROGRAM  : cq1.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "coloring a
graph."

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        *************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
static int *savescm,crow,ccol;
double aspect;
```

```
            break;

    case 'n':write_horz_str(-300,-210,"                    ",31);
            break;

    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;

    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}


/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
  setMode(3);
  videoinit();
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
        /*************************************************
                    FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void question_1  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                        MAIN PROGRAM
        *************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
           setMode(0x12);
           cls(1);
         }
     if (adapt == 'E')
        {
           setMode(16);
           cls(1);
         }
     if (adapt == 'C')
       {
           setMode(4);
           setCGAPalette(0);
           cls(1);
       }
   question_1();
   setMode(3);
}
```

```
/*************************************************************************/
static void question_1(void)
{
  LINEWIDTH = 3;
  cls(1);
  drawRect(-309,210,310,-200,11);
  write_horz_str(-300,160," Find the chromatic number for the following"
                          " graph.",31);
  drawRect(-25.75,25,25,11);
  drawLine(-25,75,25,25,11);
  drawLine(25,75,-25,25,11);
  drawLine(0,100,-50,50,11);
  drawLine(-50,50,0,0,11);
  drawLine(50,50,0,0,11);
  drawLine(50,50,0,100,11);

  aspect = 1.0;
  fillOval(0,100,3,16,aspect);
  fillOval(0,50,3,16,aspect);
  fillOval(-25,75,3,16,aspect);
  fillOval(-50,50,3,16,aspect);
  fillOval(-25,25,3,16,aspect);
  fillOval(0,0,3,16,aspect);
  fillOval(25,25,3,16,aspect);
  fillOval(50,50,3,16,aspect);
  fillOval(25,75,3,16,aspect);

  write_horz_str(-300,50," Choose one of the followings:",31);
  write_horz_str(-300,30," a) 3",31);
  write_horz_str(-300,10," b) 4",31);
  write_horz_str(-300,-10," c) 5",31);
  write_horz_str(-300,-30," d) 9",31);
  write_horz_str(-300,-60," Type the letter of which you believe"
                          " is true--->",31);
 Ch = waitkey();
 if(Ch==ESC) confirm_graph_exit();
```

```
 while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd') )) {
    write_horz_str(0,-80," Please type one of a, b, c, or d",31);
    Ch = getch ();
    if ((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
    write_horz_str(0,-80,"                                    ",31);
 }

switch (Ch)        {
 case 'a' :write_horz_str(90,-60," a ",31);
        write_horz_str(-300,-120," Yes, because the graph can be"
                " colored with 3 colors but not with a",31);
        write_horz_str(-300,-140," smaller number of colors. Press"
                " a key to see how.",31);
        break;
 case 'b' :write_horz_str(90,-60," b ",31);
        write_horz_str(-300,-120," No, the answer must be 3 because"
                " the graph can be colored with 3 colors",31);
        write_horz_str(-300,-140," but not with a smaller number of"
                " colors. Press a key to see how.",31);
        break;
 case 'c' :write_horz_str(90,-60," c ",31);
        write_horz_str(-300,-120," No, the answer must be 3 because"
                " the graph can be colored with 3 colors",31);
        write_horz_str(-300,-140," but not with a smaller number of"
                " colors. Press a key to see how.",31);
        break;

 case 'd' :write_horz_str(90,-60," d ",31);
        write_horz_str(-300,-120," No, the answer must be 3 because"
                " the graph can be colored with 3 colors",31);
        write_horz_str(-300,-140," but not with a smaller number of"
                " colors. Press a key to see how.",31);
        break;

 default : break;   }
```

```
  getch();
  fillOval(0,100,3,20,aspect);
  fillOval(0,50,3,20,aspect);
  fillOval(-25,75,3,18,aspect);
  fillOval(-50,50,3,20,aspect);
  fillOval(-25,25,3,30,aspect);
  fillOval(0,0,3,20,aspect);
  fillOval(25,25,3,18,aspect);
  fillOval(50,50,3,20,aspect);
  fillOval(25,75,3,30,aspect);
  wait("");
  cls(0);
}


/*****************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
        line = 29;
     else
        line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
       write_horz_str(30,-210," Please type y or n",79);
       ch = getch ();
       write_horz_str(30,-210,"                ",31);
    }
    switch (ch)        {
     case 'y': setMode(3);
            videoinit();
            normal_exit();
            break;
     case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

     case 'n':write_horz_str(-300,-210,"                          ",31);
            break;
     case 'N':write_horz_str(-300,-210,"                          ",31);
            break;
     default : break;
     }
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */

}
```

```
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.         */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
```

```
/* PROGRAM  : cq2.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "coloring a
graph."

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
C compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*********************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         *********************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /************************************************
                        FUNCTION DEFINITIONS
         ************************************************/

void wait(char title[]);
void question_2  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /************************************************
                          MAIN PROGRAM
         ************************************************/
main()
{
     cls (1);
     adapt = getAdapter();
     if (adapt == 'V')
        {
          setMode(0x12);
          cls(1);
        }
    if (adapt == 'E')
      {
          setMode(16);
          cls(1);
      }
    if (adapt == 'C')
      {
          setMode(4);
          setCGAPalette(0);
          cls(1);
      }
  question_2();
  setMode(3);
}
```

```
/*****************************************************************************/
static void question_2(void)
{
  LINEWIDTH = 3;
  cls(1);
  drawRect(-309,210,310,-200,11);
  write_horz_str(-300,160," Find the chromatic number for the following"
                          " graph.",31);

  drawLine(0,100,100,100,11);
  drawLine(0,100,100,50,11);
  drawLine(0,100,100,0,11);
  drawLine(100,100,0,50,11);
  drawLine(100,100,0,0,11);
  drawLine(0,50,100,50,11);
  drawLine(0,50,100,0,11);
  drawLine(100,50,0,0,11);
  drawLine(0,0,100,0,11);

  aspect = 1.0;
  fillOval(0,100,3,16,aspect);
  fillOval(100,100,3,16,aspect);
  fillOval(100,50,3,16,aspect);
  fillOval(0,50,3,16,aspect);
  fillOval(0,0,3,16,aspect);
  fillOval(100,0,3,16,aspect);

  write_horz_str(-300,50," Choose one of the followings:",31);
  write_horz_str(-300,30," a)  3",31);
  write_horz_str(-300,10," b)  5",31);
  write_horz_str(-300,-10," c)  2",31);
  write_horz_str(-300,-30," d)  4",31);
  write_horz_str(-300,-60," Type the letter of which you believe"
                          " is true--->",31);
  Ch = waitkey();
  if(Ch==ESC) confirm_graph_exit();
```

```
 while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd') )) {
    write_horz_str(0,-80," Please type one of a, b, c, or d",31);
    Ch = getch ();
    if ((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
    write_horz_str(0,-80,"                                  ",31);
 }

switch (Ch)        {
 case 'c' :write_horz_str(90,-60," c ",31);
        write_horz_str(-300,-120," Yes, because the graph can be"
                " colored with 2 colors but not with a",31);
        write_horz_str(-300,-140," smaller number of colors. Press"
                " a key to see how.",31);
        break;
 case 'b' :write_horz_str(90,-60," b ",31);
        write_horz_str(-300,-120," No, the answer must be 2 because"
                " the graph can be colored with 2 colors",31);
        write_horz_str(-300,-140," but not with a smaller number of"
                " colors. Press a key to see how.",31);
        break;
 case 'a' :write_horz_str(90,-60," a ",31);
        write_horz_str(-300,-120," No, the answer must be 2 because"
                " the graph can be colored with 2 colors",31);
        write_horz_str(-300,-140," but not with a smaller number of"
                " colors. Press a key to see how.",31);
        break;

 case 'd' :write_horz_str(90,-60," d ",31);
        write_horz_str(-300,-120," No, the answer must be 2 because"
                " the graph can be colored with 2 colors",31);
        write_horz_str(-300,-140," but not with a smaller number of"
                " colors. Press a key to see how.",31);
        break;

 default  : break;   }
 getch();
```

```
  fillOval(0,100,3,20,aspect);
  fillOval(100,100,3,18,aspect);
  fillOval(100,50,3,18,aspect);
  fillOval(0,50,3,20,aspect);
  fillOval(0,0,3,20,aspect);
  fillOval(100,0,3,18,aspect);

  wait("");
  cls(0);
}
/*****************************************************************************/
void wait(char title[])
{
    int tab,width,line,mode;
    mode = getMode(&width);
    if ((mode == 0x11) || (mode == 0x12))
      line = 29;
    else
      line = 24;
    tab = (width - strlen(title))/2;
    gotoxy(tab,0);
    writString(title,WHITE,0);
    tab = (width - 33)/2;
    gotoxy(tab,line);
    writString("Press any key to continue ...", WHITE,0);
    if(waitkey()==ESC) confirm_graph_exit();
}

/*****************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
```

```
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
  write_horz_str(30,-210," Please type y or n",79);
     ch = getch ();
      write_horz_str(30,-210,"                ",31);
  }
  switch (ch)        {
   case 'y': setMode(3);
          videoinit();
          normal_exit();
          break;
   case 'Y': setMode(3);
          videoinit();
          normal_exit();
          break;
   case 'n':write_horz_str(-300,-210,"                          ",31);
          break;
   case 'N':write_horz_str(-300,-210,"                          ",31);
          break;
   default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : cq3.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "coloring a
graph."

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /*************************************************************
          GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
         ************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
        *************************************************/

void wait(char title[]);
void question_3  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                          MAIN PROGRAM
        *************************************************/

main()
{
      adapt = getAdapter();
      if (adapt == 'V')
        {
          setMode(0x12);
          cls(1);
        }
      if (adapt == 'E')
        {
          setMode(16);
          cls(1);
        }
      if (adapt == 'C')
        {
          setMode(4);
          setCGAPalette(0);
          cls(1);
        }
  question_3();
  setMode(3);
}
```

```
/*************************************************************************/
static void question_3(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,160," Color the graph using the Welsh and Powell"
                            " algorithm.",31);
    drawLine (-100,40,-120,-30,11);   /* AD */
    drawLine (-120,-30,-70,10,11);    /* DC */
    drawLine (-70,10,-130,10,11);     /* CB */
    drawLine (-130,10,-80,-30,11);    /* BE */
    drawLine (-80,-30,-100,40,11);    /* EA */

    aspect = 1.0;
    fillOval(-100,40,3,16,aspect);
    fillOval(-120,-30,3,16,aspect);
    fillOval(-70,10,3,16,aspect);
    fillOval(-130,10,3,16,aspect);
    fillOval(-80,-30,3,16,aspect);

    write_horz_str(-300,-100," Press a key to see the coloring.",31);
    if(waitkey()==ESC) confirm_graph_exit();

    fillOval(-100,40,3,20,aspect);
    fillOval(-120,-30,3,30,aspect);
    fillOval(-70,10,3,20,aspect);
    fillOval(-130,10,3,30,aspect);
    fillOval(-80,-30,3,18,aspect);

  wait("");
  cls(0);
}
```

```
/*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
        line = 29;
     else
       line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
}

/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
           videoinit();
           normal_exit();
```

```
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

    case 'n':write_horz_str(-300,-210,"                    ",31);
            break;
    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}

/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor  */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.          */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : cq4.c
   AUTHOR   : Yavuz BAS
   DATE     : Feb. 4, 1990
   REVISED  : Mar. 5, 1990

   DESCRIPTION : This program contains an exercise for the "coloring a
graph."

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo C
compiler Version 2.0.
*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include "gtools.h"
#include "gdraws.h"
#include "colors.h"
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

        /************************************************************
           GLOBAL VARIABLES USED BY GRAPHICS FUNCTIONS
        ************************************************************/


char adapt, Ch;
int color = 2;
int LINEWIDTH=1;
unsigned long int PATTERN=0xFFFFFFFF;
double aspect;
static int *savescrn,crow,ccol;
```

```
        /*************************************************
                        FUNCTION DEFINITIONS
         *************************************************/

void wait(char title[]);
void question_4  (void);
static void confirm_graph_exit(void);
static void normal_exit(void);

        /*************************************************
                        MAIN PROGRAM
         *************************************************/

main()
{
      cls (1);
      adapt = getAdapter();
      if (adapt == 'V')
        {
            setMode(0x12);
            cls(1);
         }
     if (adapt == 'E')
        {
            setMode(16);
            cls(1);
        }
    if (adapt == 'C')
       {
            setMode(4);
            setCGAPalette(0);
            cls(1);
        }
   question_4();
   setMode(3);
}
```

```
/*************************************************************************/
static void question_4(void)
{
    LINEWIDTH = 3;
    cls(1);
    drawRect(-309,210,310,-200,11);
    write_horz_str(-300,160," Color the graph using the Welsh and Powell"
                            " algorithm.",31);
    drawRect(-50,130,50,50,11);
    drawLine (50,130,80,100,11);
    drawLine (80,100,50,50,11);
    drawLine (-50,130,80,100,11);

    aspect = 1.0;
    fillOval(-50,130,3,16,aspect);
    fillOval(-50,50,3,16,aspect);
    fillOval(50,130,3,16,aspect);
    fillOval(50,50,3,16,aspect);
    fillOval(80,100,3,16,aspect);

    write_horz_str(-300,-100," Press a key to see the coloring.",31);
    Ch = waitkey();
    if(Ch==ESC) confirm_graph_exit();
    fillOval(-50,130,3,20,aspect);
    fillOval(-50,50,3,18,aspect);
    fillOval(50,130,3,18,aspect);
    fillOval(50,50,3,20,aspect);
    fillOval(80,100,3,30,aspect);

    wait("");
    cls(0);
}
```

```
/*************************************************************************/
void wait(char title[])
{
     int tab,width,line,mode;
     mode = getMode(&width);
     if ((mode == 0x11) || (mode == 0x12))
        line = 29;
     else
        line = 24;
     tab = (width - strlen(title))/2;
     gotoxy(tab,0);
     writString(title,WHITE,0);
     tab = (width - 33)/2;
     gotoxy(tab,line);
     writString("Press any key to continue ...", WHITE,0);
     if(waitkey()==ESC) confirm_graph_exit();
}

/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  write_horz_str(-300,-210,"Quit process, are you sure (y/n)?",79);
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      write_horz_str(30,-210," Please type y or n",79);
      ch = getch ();
      write_horz_str(30,-210,"                ",31);
   }
   switch (ch)        {
    case 'y': setMode(3);
           videoinit();
           normal_exit();
```

```
            break;
    case 'Y': setMode(3);
            videoinit();
            normal_exit();
            break;

    case 'n':write_horz_str(-300,-210,"                    ",31);
            break;
    case 'N':write_horz_str(-300,-210,"                    ",31);
            break;
    default : break;
    }
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */

}

/*************************************************************************/
/* this function handles normal termination.  The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
```

```
/* PROGRAM  : props.c
   AUTHOR   : Atilla BAKAN
   DATE     : Feb. 4, 1990
   REVISED  : Apr. 16, 1990

   DESCRIPTION : This program contains the tutorial for the basic tree
                 properties.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/



/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
```

```
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk          FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif

#define _GRAPH_T_DEFINED



/* function prototypes */

/*  Utility functions          */
static void add_shadow    (void);
static void confirm_quit  (void);
static void disp_sure_msg (void);
static void error_exit    (int errnum);
static void initialize    (void);
static void move_window   (int nsrow, int scol);
static void normal_exit   (void);
static void Pageup        (void);
static void Pagedown      (void);
static void press_a_key   (int wrow);
static void pre_help      (void);
static void quit_window   (void);
static void restore_cursor(void);
static void short_delay   (void);
static void size_window   (int nerow,int necol);



/* Tutorial procedures            */
```

```
static void prop_of_trees (void);
static void definition_4_1(void);
static void example_4_1   (void);
static void theorem_4_1   (void);
static void proof_4_1     (void);
static void theorem_4_2   (void);
static void proof_4_2     (void);
static void theorem_4_3   (void);
static void proof_4_3     (void);
static void theorem_4_4   (void);
static void proof_4_4     (void);
static void theorem_4_5   (void);
static void proof_4_5     (void);
static void P1            (void);
static void P2            (void);
static void P3            (void);
static void P4            (void);
static void P5            (void);
static void P6            (void);
static void P7            (void);
static void P8            (void);
static void P9            (void);
static void P10           (void);
static void P11           (void);
static void P12           (void);
static void P13           (void);
static void exercises     (void);
static void exer1         (void);
static void exer2         (void);
static void exer3         (void);
static void exer4         (void);
```

```
/*****************************************************************/
```

```
/* miscellaneous global variables                                          */
/*************************************************************************/
static int *savescrn,crow,ccol;
static WINDOW w[21];
static char ssan[10];


/*************************************************************************/
/* error message table                                                     */
/*************************************************************************/
static char *error_text[]= {
   NULL,  /* errnum = 0, no error   */
   NULL,  /* errnum == 1, windowing error */
   "Syntax:  CXLDEMO [-switches]\n\n"
      "\t -c = CGA snow elimination\n"
      "\t -b = BIOS screen writing\n"
      "\t -m = force monochrome text attributes",
   "Memory allocation error"
};



/*************************************************************************/
/* miscellaneous defines                                                   */
/*************************************************************************/
#define SHORT_DELAY 18
#define H_WINTITLE  33

/* this function will add a shadow to the active window */

static void add_shadow(void)
{
   wshadow(LGREY|_BLACK);
}

/*************************************************************************/
```

```
/* this function pops open a window and confirms that the user really          */
/* wants to quit the demo.  If so, it terminates the demo program.             */
/*****************************************************************************/
static void confirm_quit(void)
{
  struct _onkey_t *kblist;

  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
  add_shadow();
  wputs("\n Quit demo, are you sure? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y') normal_exit();
  wclose();
  hidecur();
  if(_ ouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
}

/*****************************************************************************/
/* this function is called by the pull-down demo for a prompt                */
/*****************************************************************************/
static void disp_sure_msg(void)
{
  wprints(0,2,WHITE|_BLUE,"Are you sure?");
}
/
```

```
/*************************************************************************/
/* this function handles abnormal termination. If it is passed an        */
/* error code of 1, then it is a windowing system error. Otherwise        */
/* the error message is looked up in the error message table.             */
/*************************************************************************/
static void error_exit(int errnum)
{
   if(errnum) {
      printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
               exit(errnum);
   }
}
/*************************************************************************/
/* this function initializes CXL's video, mouse, keyboard, and help systems */
/*************************************************************************/
static void initialize(void)
{
   /* initialize the CXL video system and save current screen info */
   videoinit();
   readcur(&crow,&ccol);
   if((savescrn=ssave())==NULL) error_exit(3);
   if(msinit()) {
      mssupport(MS_FULL);
      msgotoxy(12,49);
   }
   /* attach [Alt-X] to the confirm_quit() function */
   setonkey(0x2d00,confirm_quit,0);
   /* attach [Ctrl Pageup] to the Pageup() function */
   setonkey(0x8400,Pageup,0);
   /* attach [Ctrl Pagedown] to the Pagedown() function */
   setonkey(0x7600,Pagedown,0);
   /* initialize help system, help key = [F1] */
   whelpef("CXLDEMO.HLP",0x3b00,YELLOW|_RED,LRED|_RED,
            WHITE|_RED,RED|_LGREY ,pre_help);
}
```

```
/*************************************************************************/
static void pre_help(void)
{
   add_shadow();
   setonkey(0x2d00,confirm_quit,0);
}

/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.        */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescrn);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}
/*************************************************************************/
/* this function displays a pause message then pauses for a keypress      */
/*************************************************************************/
static void press_a_key(int wrow)
{
   register int attr1;
   register int attr2;

   attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
   attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
   wcenters(wrow,attr1,"Press a key");
   wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
   hidecur();
   if(waitkey()==ESC) confirm_quit();
   wcenters(wrow,attr1,"          ");
   wprints(wrow,0,attr2,"        ");
}
```

```
/*************************************************************************/
/* This routine causes short dealys during execution                     */
/*************************************************************************/
static void short_delay(void)
{
  delay_(SHORT_DELAY);
}

/*************************************************************************/
/* this function is called by the pull-down menu demo anytime            */
/* the  selection bar moves on or off the [Q]uit menu items.             */
/*************************************************************************/
static void quit_window(void)
{
  static WINDOW handle=0;

  if(handle) {
    wactiv(handle);
    wclose();
    handle=0;
  }
  else {
    handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
    wputs(" Quit takes you back to the\n demo program's main menu.");
  }
}
/*************************************************************************/
/* this function is called anytime to switch back to previous window.    */
/*************************************************************************/
static void Pageup(void)
{
  static WINDOW handle;

  handle = whandle();
  wactiv(handle - 1);
}
```

```
/*************************************************************************/
/* this function is called anytime to switch back to next window.        */
/*************************************************************************/
static void Pagedown(void)
{
   static WINDOW handle;

   handle = whandle();
   wactiv(handle + 1);
}



/*************************************************************************/
/* shows the cursor again if it has been hidden                          */
/*************************************************************************/
static void restore_cursor(void)
{
   wtextattr(WHITEl_MAGENTA);
   showcur();
}



/*************************************************************************/
/* enlarges or shrinks the windows                                       */
/*************************************************************************/
static void size_window(int nerow,int necol)
{
   wsize(nerow,necol);
   short_delay();
}
```

```
/*************************************************************************/
/* moves the active window to a given screen coordinates                 */
/*************************************************************************/
static void move_window(int nsrow,int nscol)
{
  if(wmove(nsrow,nscol)) error_exit(1);
  short_delay();
}
/*************************************************************************/
/* main routine that calls basic properties of trees tutorial            */
/*************************************************************************/
void main()
{
  initialize();
  prop_of_trees();
}

/*************************************************************************/
/* this routine that calls definition_4_1 routine whenever Pageup or Pagedown  */
/* keys are pressed.                                                     */
/*************************************************************************/
void P1()
{
  wcloseall();
  definition_4_1();
}

/*************************************************************************/
/* this routine that calls prop_of_trees  routine whenever Pageup or Pagedown  */
/* keys are pressed.                                                     */
/*************************************************************************/
void P2()
{
  wcloseall();
  prop_of_trees();
}
```

```
/*************************************************************************/
/* this routine that calls example_4_1  routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                             */
/*************************************************************************/
void P3()
{
   wcloseall();
   example_4_1();
}
/*************************************************************************/
/* this routine that calls theorem_4_1  routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                             */
/*************************************************************************/
void P4()
{
   wcloseall();
   theorem_4_1();
}
/*************************************************************************/
/* this routine that calls theorem_4_2  routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                             */
/*************************************************************************/
void P5()
{
   wcloseall();
   theorem_4_2();
}
/*************************************************************************/
/* this routine that calls theorem_4_3  routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                             */
/*************************************************************************/
void P6()
{
   wcloseall();
   theorem_4_3();
}
```

```
/*************************************************************************/
/* this routine that calls theorem_4_7  routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                             */
/*************************************************************************/
void P7()
{
   wcloseall();
   theorem_4_4();
}
/*************************************************************************/
/* this routine that calls theorem_4_8  routine whenever Pageup or Pagedown      */
/* keys are pressed.                                                             */
/*************************************************************************/
void P8()
{
   wcloseall();
   theorem_4_5();
}
/*************************************************************************/
/* this routine that calls exercises  routine whenever Pageup or Pagedown        */
/* keys are pressed.                                                             */
/*************************************************************************/
void P9()
{
   wcloseall();
   exercises();
}
/*************************************************************************/
/* this routine that calls exercises  routine whenever Pageup or Pagedown        */
/* keys are pressed.                                                             */
/*************************************************************************/
void P10()
{
   wcloseall(),
   exer1();
}
```

```
/*************************************************************************/
/* this routine that calls exercises  routine whenever Pageup or Pagedown  */
/* keys are pressed.                                                        */
/*************************************************************************/
void P11()
{
   wcloseall();
   exer2();
}

/*************************************************************************/
/* this routine that calls exercises  routine whenever Pageup or Pagedown  */
/* keys are pressed.                                                        */
/*************************************************************************/
void P12()
{
   wcloseall();
   exer3();
}

/*************************************************************************/
/* this routine that calls exercises  routine whenever Pageup or Pagedown  */
/* keys are pressed.                                                        */
/*************************************************************************/
void P13()
{
   wcloseall();
   exer4();
}
```

```
/*************************************************************************/
/* This routine tells about the basic properties of the trees. It gives the */
/* definitions, examples and theorems about the trees.                    */
/*************************************************************************/
static void prop_of_trees(void)
{
  register int *scrn;

  if((scrn=ssave())==NULL) error_exit(3);
  cclrscrn(LGREY|_BLUE);
 /*************************************************************************/
 /* attach [Pagedown] to the definition_4_1() function */
  setonkey(0x5100,P1,0);
 /*************************************************************************/
 if((w[1]=wopen(5,15,17,65,3,LCYAN|_GREEN,WHITE|_CYAN))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" Trees are very important class of graphs. They have a very clean"
         " set of properties, which are summarized in Theorem 4.5. In terms"
         " of applications, tree structures appear naturally as a representation"
         " vehicle for expression evaluation in sorting and searching problems,"
         " and in numerous situations where hierarchical representations are"
         " needed. The definition of a tree is as follows.");
  press_a_key(10);
  wslide(0,0);
  short_delay();
 /*************************************************************************/
  definition_4_1();
  srestore(scrn);

}
```

```
/*************************************************************************/
/* Dummy function to call the actual definition 4.1                      */
/*************************************************************************/
void definition_4_1(void)
{
  /***********************************************************************/
   /* attach [Pageup] to the prop_of_trees() function */
   setonkey(0x4900,P2,0);
  /***********************************************************************/
   /* attach [Pagedown] to the example_4_1() function */
   setonkey(0x5100,P3,0);
  /***********************************************************************/
   if((w[2]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
                                                        error_exit(1);
   wtitle("[Properties of Trees - Definition_4_1]",TCENTER,_LGREY|BROWN);
   whelpcat(H_WINTITLE);
   add_shadow();
   wputs("\n");
   wputsw(" Def : A connected graph T is a tree if T has no cycles");
   press_a_key(3);
   short_delay();
   wcloseall();
  /***********************************************************************/
   example_4_1();
}
```

```
/*************************************************************************/
/* Dummy function to call the actual example 4.1                         */
/*************************************************************************/
void example_4_1(void)
{

 /*************************************************************************/
  /* attach [Pageup] to the definition_4_1() function */
  setonkey(0x4900,P1,0);
 /*************************************************************************/
  /* attach [Pagedown] to the theorem_4_1() function */
  setonkey(0x5100,P4,0);
 /*************************************************************************/
  if((w[3]=wopen(5,15,11,65,3,WHITE|_BLACK,WHITE|_LGREY))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees - Example_4_1]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" The following figure shows some graphs that are trees "
         " and also graphs that do not have tree properties");
  press_a_key(4);
  short_delay();
  wclose();
  spawnl(P_WAIT,"examp41.exe",NULL);
  cclrscm(LGREY|_BLUE);
  theorem_4_1();
}
```

```
/*************************************************************************/
/* This routine gives the first theorem about the trees. Besides, if the user   */
/* wants, it gives the proof for this theorem.                                   */
/*************************************************************************/
void theorem_4_1(void)
 {
  struct _onkey_t *kblist;

  /*************************************************************************/
  /* attach [Pageup] to the example_4_1() function */
  setonkey(0x4900,P3,0);
  /*************************************************************************/
  /* attach [Pagedown] to the theorem_4_2() function */
  setonkey(0x5100,P5,0);
  /*************************************************************************/
  if((w[4]=wopen(2,4,9,71,3,LCYAN|_GREEN,WHITE|_LGREY))==-0)
                                                        error_exit(1);
  wtitle("[Properties of Trees - Theorem_4_1]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\nTheorem_4_1\n");
  wputsw("In a tree is exactly one simple path between any two vertices.");
  press_a_key(5);
  /**   *************************************************************/
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,20,13,55,0,BROWN|_CYAN,RED|_BLACK)) error_exit(1);
  add_shadow();
  wputs("\n Do you want to see the proof? \0.    \156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y')  {
    wclose();
    proof_4_1();
  }
  else wclose();
```

```
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
  /*************************************************************************/
  short_delay();
  wcloseall();
  theorem_4_2();
 }
/*************************************************************************/
/* This routine gives the proof to the theorem_4_1.                      */
/*************************************************************************/
static void proof_4_1(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the example_4_1() function */
  setonkey(0x4900,P3,0);
  /*************************************************************************/
  /* attach [Pagedown] to the theorem_4_2() function */
  setonkey(0x5100,P5,0);
  /*************************************************************************/
  if((w[5]=wopen(9,4,19,71,3,LCYAN|_RED,WHITE|_GREEN))==0) error_exit(1);
  wtitle("[Theorem_4_1 - Proof]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" Suppose T is a tree. Thus, by Theorem 3.3 there is a simple path"
         " between any two vertices. We will make our proof by way of"
         " contradiction. Now suppose that there exist 2 distinct simple paths");
  wputs("\n");
  wputs("          U, U1, U2,...,Uk, V\n");
  wputs("          U, V1, V2,...,Vm, V\n");
  wputsw(" Now consider the following graph.");
  press_a_key(8);
  spawnl(P_WAIT,"examp42.exe",NULL);
  cclrscrn(LGREY|_BLUE);
}
```

```
/*************************************************************************/
/* This routine gives the theorem_4_2 about the trees. Besides, if the user      */
/* wants, it gives the proof for this theorem.                                   */
/*************************************************************************/
void theorem_4_2(void)
 {
  struct _onkey_t *kblist;
  /*************************************************************************/
  /* attach [Pageup] to the theorem_4_1() function */
  setonkey(0x4900,P4,0);
  /*************************************************************************/
  /* attach [Pagedown] to the theorem_4_3() function */
  setonkey(0x5100,P6,0);
  /*************************************************************************/
  if((w[6]=wopen(3,4,10,71,3,LCYAN|_GREEN,WHITE|_LGREY))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees - Theorem_4_2]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\nTheorem_4_2\n");
  wputsw(" In a tree T with more than one vertex there are at least two vertices"
         " of degree 1. ");
  press_a_key(5);
  /*************************************************************************/
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,20,13,55,0,BROWN|_CYAN,RED|_BLACK)) error_exit(1);
  add_shadow();
  wputs("\n Do you want to see the proof? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y') {
    wclose();
    proof_4_2();
   }
  else wclose();
```

```
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
  /*************************************************************************/
  short_delay();
  wcloseall();
  theorem_4_3();
 }
/*************************************************************************/
/* This routine gives the proof to the theorem_4_2.                      */
/*************************************************************************/
 static void proof_4_2(void)
 {
  /*************************************************************************/
  /* attach [Pageup] to the theorem_4_1() function */
  setonkey(0x4900,P4,0);
  /*************************************************************************/
  /* attach [Pagedown] to the theorem_4_3() function */
  setonkey(0x5100,P6,0);
  /*************************************************************************/
  if((w[7]=wopen(11,4,21,71,3,LCYAN|_RED,WHITE|_GREEN))==0)
                                                        error_exit(1);
  wtitle("[Theorem_4_2 - Proof]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" Since T is a connected graph with at least two vertices, there is a"
         " simple path with at least two distinct vertices. Suppose the simple path"
        " with the maximal number of edges, is say from U to V, where U and V are"
        " distinct. If U has degree more than 1, then since T has no  cycles,"
         " a longer simple path would exist; we can make the same argument"
         " for V too. Thus U and V have  degree 1.");
  press_a_key(8);
  short_delay();
  wclose();
 }
```

```
/*************************************************************************/
/* This routine gives the theorem_4_3 about the trees. Besides, if the user      */
/* wants, it gives the proof for this theorem.                                   */
/*************************************************************************/
static  void theorem_4_3(void)
 {
  struct _onkey_t *kblist;

  /*************************************************************************/
  /* attach [Pageup] to the theorem_4_2() function */
  setonkey(0x4900,P5,0);
  /*************************************************************************/
  /* attach [Pagedown] to the theorem_4_4() function */
  setonkey(0x5100,P7,0);
  /*************************************************************************/
  if((w[8]=wopen(2,4,7,71,3,LCYAN|_GREEN,WHITE|_LGREY))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees - Theorem_4_3]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n  Theorem_4_3\n");
  wputsw("   A tree with n vertices has exactly  n - 1 edges.");
  press_a_key(3);
  /*************************************************************************/
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,20,13,55,0,BROWN|_CYAN,RED|_BLACK)) error_exit(1);
  add_shadow();
  wputs("\n Do you want to see the proof? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y')  {
   wclose();
   proof_4_3();
  }
  else  wclose();
```

```
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
  /*************************************************************************/
  short_delay();
  wcloseall();
  theorem_4_4();
}
```

```
/*************************************************************************/
/* This routine gives the proof to the theorem_4_3.                      */
/*************************************************************************/
static void proof_4_3(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the theorem_4_2() function */
  setonkey(0x4900,P5,0);
  /***********************************************************************/
  /* attach [Pagedown] to the theorem_4_4() function */
  setonkey(0x5100,P7,0);
  /***********************************************************************/
  if((w[9]=wopen(8,4,23,71,3,LCYAN|_RED,WHITE|_GREEN))==0) error_exit(1);
  wtitle("[Theorem_4_3 - Proof]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw("  Suppose T is a tree. The proof will be done by induction on  n,"
         " the number of vertices.  We will first consider the base  case,"
         " where n = 1. Because since this graph is a tree  there may not"
         " be any loops in the graph. Hence there can b    o edges in a tree"
         " with only one vertex,  the theorem holds when n = 1.");
  wputs("\n\n");
  wputsw("  Now assume t' , theorem holds for a tree which have k vertices."
         " This  is our induction hypothesis. Now for the inductive step"
         " we   will prove that the theorem holds for a tree T with k + 1"
         " vertices.  We know that by Theorem 4.2 there is a vertex V which"
         " has degree 1. Now suppose we remove the vertex V and the edge on"
         " V from the graph T to obtain a new graph T'. To see the graphs...");
  press_a_key(13);
  short_delay();
  wcloseall();
  /***********************************************************************/
  spawnl(P_WAIT,"examp43.exe",NULL);
  cclrscrn(LGREY|_BLUE);
}
```

```
/*************************************************************************/
/* This routine gives the theorem_4_4 about the trees. Besides, if the user      */
/* wants, it gives the proof for this theorem.                                   */
/*************************************************************************/
static void theorem_4_4(void)
 {
  struct _onkey_t *kblist;

  /*************************************************************************/
  /* attach [Pageup] to the theorem_4_3() function */
  setonkey(0x4900,P6,0);
  /*************************************************************************/
  /* attach [Pagedown] to the theorem_4_5() function */
  setonkey(0x5100,P8,0);
  /*************************************************************************/
  if((w[10]=wopen(3,4,12,71,3,LCYAN|_GREEN,WHITE|_LGREY))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees - Theorem_4_4]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("Theorem_4_4\n");
  wputsw(" (a) When an edge is removed from a tree (leaving all the vertices),"
         " the resulting graph is not connected and hence is not a tree.");
  wputs("\n");
  wputsw("(b) When an edge is added to a tree (without adding additional vertices),"
         " the resulting graph has a cycle and hence is not a tree.");
  press_a_key(7);
  /*************************************************************************/
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,20,13,55,0,BROWN|_CYAN,RED|_BLACK)) error_exit(1);
  add_shadow();
  wputs("\n Do you want to see the proof? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y')  {
```

```
    wclose();
    proof_4_4();
  }
  else  wclose();
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
  /*****************************************************************/
  short_delay();
  wcloseall();
  theorem_4_5();
}
```

```
/*************************************************************************/
/* This routine gives the proof to the theorem_4_4.                      */
/*************************************************************************/
static void proof_4_4(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the theorem_4_3() function */
  setonkey(0x4900,P6,0);
  /***********************************************************************/
  /* attach [Pagedown] to the theorem_4_5() function */
  setonkey(0x5100,P8,0);
  /***********************************************************************/
  if((w[11]=wopen(13,4,21,71,3,LCYAN|_RED,WHITE|_GREEN))==0)
                                                        error_exit(1);
  wtitle("[Theorem_4_4 - Proof]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" If an edge is added or removed from a tree, the resulting new graph"
         " can  no longer be a tree by Theorem 4.3. Since removing an edge cannot"
         " create a cycle nor adding an edge disconnect the graph, both parts"
         " of the theorem follow.");
  press_a_key(6);
  short_delay();
  wclose();
}
```

```
/*************************************************************************/
/* This routine gives the theorem_4_5 about the trees. Besides, if the user      */
/* wants, it gives the proof for this theorem.                                   */
/*************************************************************************/
void theorem_4_5(void)
 {
  struct _onkey_t *kblist;

  /*************************************************************************/
  /* attach [Pageup] to the theorem_4_4() function */
  setonkey(0x4900,P7,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exercises() function */
  setonkey(0x5100,P9,0);
  /*************************************************************************/
        if((w[12]=wopen(0,0,19,35,3,LCYAN|_GREEN,WHITE|_LGREY))==0)   er-
ror_exit(1);
  wtitle("[Properties of Trees - Theorem_4_5]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\nTheorem_4_5\n");
  wputs("(a) T is a tree\n");
  wputsw(" (b) There is exactly one simple path between each two vertices in T.");
  wputs("\n");
  wputsw(" (c) T is connected and the number of vertices is one more than"
         " the number of edges.");
  wputs("\n");
  wputsw(" (d) T is connected and removing any edge disconnects T.");
  wputs("\n");
  wputsw(" (e) T has no cycles and the number of vertices is one more than"
         " the number of edges.");
  wputs("\n");
  wputsw(" (f) T has no cycles and adding any edge creates a cycle");
  press_a_key(17);
  /*************************************************************************/
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
```

```
 if(_mouse&MS_CURS) mshidecur();
 if(!wopen(9,20,13,55,0,BROWN|_CYAN,RED|_BLACK)) error_exit(1);
 add_shadow();
 wputs("\n Do you want to see the proofs? \033A\156Y\b");
 clearkeys();
 showcur();
 if(wgetchf("YN",'Y')=='Y') {
  wclose();
  proof_4_5();
 }
 else  wclose();
 hidecur();
 if(_mouse&MS_CURS) msshowcur();
 chgonkey(kblist);      /* restore any hidden hot keys */
 /*************************************************************************/
 short_delay();
 wcloseall();
 exercises();
}
```

```
/*************************************************************************/
/* This routine gives the proof to the theorem_4_5.                      */
/*************************************************************************/
static void proof_4_5(void)
{
  register int *scrn;


  /*************************************************************************/
  /* attach [Pageup] to the theorem_4_4() function */
  setonkey(0x4900,P7,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exercises() function */
  setonkey(0x5100,P9,0);
  /*************************************************************************/
  if((scrn=ssave())==NULL) error_exit(3);
  /*************************************************************************/
  if((w[13]=wopen(0,36,9,79,3,LCYAN|_RED,WHITE|_CYAN))==0) error_exit(1);
  wtitle("[Theorem_4_3]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" We will prove only  three of the required implications, and"
         " wait you to prove the others. The important thing to get from"
         " the theorem is an intuitive feel for the various equivalent"
         " statements. Drawing several graphs will be helpful.");
  press_a_key(7);
  wclose();
  /*************************************************************************/
          if((w[14]=wopen(0,36,15,79,3,LCYAN|_RED,WHITE|_CYAN))==0)    er-
ror_exit(1);
  wtitle("[Theorem_4_5 - Proof]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n(a) => (b)\n");
  wputsw(" This implication follows from two seperate statements about"
```

```
        " connected graphs and acyclic graphs. The first is that T is"
        " connected if and only if there exists at least one path between"
        " any pair of vertices in T. The second is that T contains a cycle"
        " if and only if there exist vertices U and V with two paths between"
        " them. The following figure illustrates the latter statement.");
press_a_key(13);
short_delay();
wclose();
/*****************************************************************/
spawnl(P_WAIT,"examp44.exe",NULL);
cclrscrn(LGREY|_BLUE);
srestore(scrn);
/*****************************************************************/
if((scrn=ssave())==NULL) error_exit(3);
/*****************************************************************/
if((w[15]=wopen(0,36,24,79,3,WHITE|_RED,RED|_LGREY))==0) error_exit(1);
wtitle("[Theorem_4_5 - Proof]",TCENTER,_MAGENTA|WHITE);
add_shadow();
whelpcat(H_WINTITLE);
wputs("(b) => (c)\n");
wputsw(" If there is exactly one path between any pair of vertices in T,"
        " then T is connected. To show that the number of vertices is one"
        " more than the number of edges, we will use induction argument on"
        " the number of vertices in T, and we will denote it by n. To  do"
        " so, we let P(n) be the following assertion, for n >= 1;");
wputs("\n\n");
wputsw(" P(n) : If T has k vertices, 1 <= k <= n, and if there is"
        " exactly one path between any pair of vertices, then the  number"
        " of edges in T is k - 1.");
wputs("\n\n");
wputsw(" For the base step, observe that if n = 1, T has 0 edges, P(1)"
        " holds. For the induction step, assume that P(n) is true for "
        " some n >= 1. Let T be a graph with  n + 1 vertices and suppose"
        " there exists exactly one path between any pair of vertices. "
        " Now consider the next graph...");
press_a_key(22);
```

```
  short_delay();
  wclose();
  /*************************************************************************/
  spawnl(P_WAIT,"examp45.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  srestore(scrn);
  /*************************************************************************/
  if((w[16]=wopen(0,36,12,79,3,WHITE|_RED,RED|_LGREY))==0) error_exit(1);
  wtitle("[Theoren _ 1_5 - Proof]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n(f) => (a)\n");
  wputsw(" Assume that (f) is true. We need to show that T is connected. "
         " Suppose U and V are vertices. If we add the edge { U, V } to "
         " graph, our hypothesis implies that we have created a cycle."
         " Because, since T is a tree and U and V are in the T, there"
         " must have been a path from U to V already. Hence, G is connected");
  press_a_key(10);
  short_delay();
  wcloseall();
}
```

```
/*************************************************************************/
/* This routine makes a small quiz about the basic properties of trees.  */
/*************************************************************************/
void exercises(void)
 {
  register int *screen;
  /*************************************************************************/
  /* attach [Pageup] to the theorem_4_5() function */
  setonkey(0x4900,P8,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exer1() function */
  setonkey(0x5100,P10,0);
  /*************************************************************************/
  if((w[17]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw(" We have completed our presentation of this section. Are"
         " you ready for a pop quiz ? ");
  press_a_key(3);
  short_delay();
  wclose();
  if((screen=ssave())==NULL) error_exit(3); {
  /*************************************************************************/
  exer1();
  /* if mouse exists, turn on full mouse support */
   if(msinit()) {
     mssupport(MS_FULL);
     msgotoxy(12,49);
     }
  }
  /*************************************************************************/
  srestore(screen);
}
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise 4.1.1                      */
/*************************************************************************/
static void exer1(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the theorem_4_5() function */
  setonkey(0x4900,P8,0);
  /***********************************************************************/
  /* attach [Pagedown] to the exer2() function */
  setonkey(0x5100,P11,0);
  /***********************************************************************/
  if((w[18]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the first question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q411.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer2();
}
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise 4.1.2                      */
/*************************************************************************/
static void exer2(void)
{
  /*************** ********************************************************/
  /* attach [Pageup] to the exer1() function */
  setonkey(0x4900,P10,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exer3() function */
  setonkey(0x5100,P12,0);
  /*************************************************************************/
  cclrscrn(LGREY|_BLUE);
  if((w[19]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Second question.");
  press_a_key(3);
  short_delay();
  wclose();
  spawnl(P_WAIT,"q412.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer3();
}
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise 4.1.3                      */
/*************************************************************************/
static void exer3(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the exer2() function */
  setonkey(0x4900,P11,0);
  /***********************************************************************/
  /* attach [Pagedown] to the exer4() function */
  setonkey(0x5100,P13,0);
  /***********************************************************************/
  cclrscm(LGREY|_BLUE);
  if((w[20]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Third question.");
  press_a_key(3);
  short_delay();
  wclose();
  spawnl(P_WAIT,"q413.exe",NULL);
  cclrscm(LGREY|_BLUE);
  exer4();
}
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise 4.1.4                      */
/*************************************************************************/
static void exer4(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the exer3() function */
  setonkey(0x4900,P12,0);
  /***********************************************************************/
  cclrscrn(LGREY|_BLUE);
  if((w[21]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
                                                        error_exit(1);
  wtitle("[Properties of Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Fourth question.");
  press_a_key(3);
  short_delay();
  wclose();
  spawnl(P_WAIT,"q414.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  normal_exit();
}
```

```
/*PROGRAM : examp41.c
  AUTHOR   : Atilla BAKAN
  DATE     : Apr. 15, 1990
  REVISED  : Apr. 15, 1990

  DESCRIPTION : This routine contains examples for graphs which are
                trees/not trees.


  MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                     C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                    /* Turbo C */
  #include <dir.h>
#else
  #include <direct.h>                   /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)        /* MSC/QuickC */
  #define bioskey(a)      _bios_keybrd(a)
  #define findfirst(a,b,c) _dos_findfirst(a,c,b)
  #define findnext(a)     _dos_findnext(a)
  #define ffblk          find_t
  #define ff_name         name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
  #define ffblk          FIND
  #define ff_name         name
  #define ff_attrib       attribute
```

```
#endif
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions       */
static void init_graph   (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void example_4_1   (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;

/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
   settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
      }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
      }
   forecolor = WHITE;
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
  case 'y': closegraph();
          videoinit();
          exit(0);
          break;

  case 'Y': closegraph();
          videoinit();
          exit(0);
          break;

  case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
```

```
            break;

     case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;

     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}

/*-------------------------------------------------------------------*/
/* This function sets the text default values                        */
/*-------------------------------------------------------------------*/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}


/*-------------------------------------------------------------------*/
/* Equivalent of press_a_key function for graphics screen            */
/*-------------------------------------------------------------------*/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>PRESS A KEY TO CONTINUE<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
```

```
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  example_4_1();
}

/*************************************************************************/
/* This routine gives examples of trees and some graphs that are not trees.  */
/*************************************************************************/
 void example_4_1(void)
 {
  /************************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-6-1");
  /************************************************************************/
  outtextxy(5*x,5*y," Graphs that are trees");
  pieslice(7*x,10*y,0,359,2);
  pieslice(10*x,13*y,0,359,2);
  pieslice(15*x,13*y,0,359,2);
  pieslice(18*x,10*y,0,359,2);
  pieslice(18*x,16*y,0,359,2);
  pieslice(7*x,16*y,0,359,2);
  pieslice(32*x,10*y,0,359,2);
  pieslice(37*x,10*y,0,359,2);
  pieslice(35*x,16*y,0,359,2);
  pieslice(39*x,16*y,0,359,2);
  pieslice(42*x,10*y,0,359,2);
  moveto(7*x,10*y);  lineto(10*x,13*y);
  lineto(15*x,13*y); lineto(18*x,10*y);
  moveto(10*x,13*y); lineto(7*x,16*y);
  moveto(15*x,13*y); lineto(18*x,16*y);
```

```
moveto(32*x,10*y); lineto(35*x,16*y);
lineto(37*x,10*y); lineto(39*x,16*y);   lineto(42*x,10*y);
outtextxy(11*x,20*y,"(a)");
outtextxy(35*x,20*y,"(b)");
Pause(30*x,24*y);
/*************************************************************************/
outtextxy(MaxX/2+6*x,5*y," Graphs that are not trees");
pieslice(MaxX/2+ 5*x,10*y,0,359,2);
pieslice(MaxX/2+10*x,12*y,0,359,2);
pieslice(MaxX/2+10*x,14*y,0,359,2);
pieslice(MaxX/2+15*x,10*y,0,359,2);
pieslice(MaxX/2+5*x,16*y,0,359,2);
pieslice(MaxX/2+15*x,16*y,0,359,2);
pieslice(MaxX/2+30*x,10*y,0,359,2);
pieslice(MaxX/2+30*x,16*y,0,359,2);
pieslice(MaxX/2+37*x,10*y,0,359,2);
pieslice(MaxX/2+42*x,16*y,0,359,2);
pieslice(MaxX/2+27*x,12*y,0,359,2);
pieslice(MaxX/2+35*x,16*y,0,359,2);
moveto(MaxX/2+5*x,10*y);  lineto(MaxX/2+10*x,12*y);
lineto(MaxX/2+15*x,10*y);
moveto(MaxX/2+5*x,16*y);  lineto(MaxX/2+10*x,14*y);
lineto(MaxX/2+15*x,16*y);
moveto(MaxX/2+30*x,10*y); lineto(MaxX/2+27*x,12*y);
lineto(MaxX/2+30*x,16*y); lineto(MaxX/2+30*x,10*y);
lineto(MaxX/2+37*x,10*y); lineto(MaxX/2+35*x,16*y);
lineto(MaxX/2+42*x,16*y);
settextstyle(0,0,0);
outtextxy(MaxX/2+11*x,20*y,"(c)");
outtextxy(MaxX/2+30*x,20*y,"(d)");
outtextxy(MaxX/2+x,22*y,"  Not connected");  /* reasons why the graphs */
outtextxy(MaxX/2+25*x,22*y," Has a cycle");    /* that are not trees.   */
/*************************************************************************/
Pause(30*x,24*y);
closegraph();
}
```

```
/* PROGRAM   : examp42.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 16, 1990
   REVISED   : Apr. 16, 1990

   DESCRIPTION : This routine contains examples for graphs which are
                 trees/not trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/
/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph   (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void setc...      (void);

/* tutorial functions    */
static void example_4_2  (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor,
int quitcolor;
int x, y, MaxX, MaxY;


/*-----------------------------------------------------------------*/
/* This function is used for including drivers to the executable code */
/*-----------------------------------------------------------------*/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This function initializes the necessary graphical routines            */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
   settext();
   if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
     }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
     }
   forecolor = WHITE;
   }
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,47*y/2,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,47*y/2,69*x,97*y/4);
     setcolor(quitcolor);
   }
   switch (ch)        {
    case 'y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,47*y/2,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
```

```
    case 'N': setcolor(backcolor);
              bar(4*x/3,47*y/2,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}



/*-------------------------------------------------------------------*/
/* This function sets the text default values                        */
/*-------------------------------------------------------------------*/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/*-------------------------------------------------------------------*/
/* Equivalent of press_a_key function for graphics screen            */
/*-------------------------------------------------------------------*/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>PRESS A KEY TO CONTINUE<<");
  if(waitkey()==ESC) confirm_graph_exit();
 }
```

```
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  example_4_2();
}

/*-----------------------------------------------------------------------*/
/* This routine gives examples of trees and some graphs that are not trees.  */
/*-----------------------------------------------------------------------*/
 void example_4_2(void)
 {
  /*************************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-2");
  /*************************************************************************/
  pieslice(25*x,5*y,0,359,2);  /* U  */
  pieslice(35*x,3*y,0,359,2);  /* U1 */
  pieslice(40*x,3*y,0,359,2);
  pieslice(50*x,3*y,0,359,2);
  pieslice(55*x,3*y,0,359,2);  /* Uk */
  pieslice(65*x,5*y,0,359,2);
  pieslice(35*x,7*y,0,359,2);  /* V1 */
  pieslice(50*x,7*y,0,359,2);
  pieslice(55*x,7*y,0,359,2);  /* Vm */
  pieslice(65*x,5*y,0,359,2);  /* V  */
  moveto(25*x,5*y); lineto(35*x,3*y); lineto(40*x,3*y);
  putpixel(42*x,3*y,2);
  putpixel(45*x,3*y,2);
  putpixel(48*x,3*y,2);
  moveto(50*x,3*y); lineto(55*x,3*y); lineto(65*x,5*y);
  moveto(25*x,5*y); lineto(35*x,7*y); lineto(40*x,7*y);
```

```
putpixel(42*x,7*y,2);
putpixel(45*x,7*y,2);
putpixel(48*x,7*y,2);
moveto(50*x,7*y); lineto(55*x,7*y); lineto(65*x,5*y);
outtextxy(23*x,5*y,"U");
outtextxy(35*x,3*y/2,"U1");
outtextxy(55*x,3*y/2,"Uk");
outtextxy(35*x,8*y,"V1");
outtextxy(57*x,8*y,"Vm");
outtextxy(67*x,5*y,"V");
/*************************************************************************/
outtextxy(2*x,10*y," If all Ui = Vj then U, U1, ..., Uk, V , Vm, ..., V1, U is a cycle
                    which is a");
outtextxy(2*x,11*y,"  contradiction to the definition of a tree.");
outtextxy(2*x,12*y," Now think about the other case, where existence of an
                     vertice such that this");
outtextxy(2*x,13*y," vertice is not equal to U or V, ");
/*************************************************************************/
pieslice(20*x,17*y,0,359,2);   /* U  */
pieslice(30*x,15*y,0,359,2);   /* U1 */
pieslice(30*x,19*y,0,359,2);   /* Vi */
pieslice(75*x/2,17*y,0,359,2); /* V  */
pieslice(60*x,15*y,0,359,2);   /* Uk */
pieslice(60*x,19*y,0,359,2);   /* Vm */
pieslice(70*x,17*y,0,359,2);
pieslice(105*x/2,17*y,0,359,2);
pieslice(45*x,19*y,0,359,2);
pieslice(45*x,15*y,0,359,2);
moveto(20*x,17*y); lineto(30*x,15*y);
moveto(20*x,17*y); lineto(30*x,19*y);
setlinestyle(3,0,1);
moveto(30*x,15*y); lineto(45*x,19*y); lineto(60*x,15*y);
moveto(30*x,19*y); lineto(45*x,15*y); lineto(60*x,19*y);
setlinestyle(0,0,3);
moveto(60*x,19*y); lineto(70*x,17*y);
moveto(60*x,15*y); lineto(70*x,17*y);
```

```
 outtextxy(18*x,17*y,"U");
 outtextxy(27*x,29*y/2,"U1");
 outtextxy(63*x,29*y/2,"Uk");
 outtextxy(33*x,17*y,"Ui");
 outtextxy(41*x,17*y,"Vj");
 outtextxy(27*x,19*y,"V1");
 outtextxy(63*x,19*y,"Vm");
 outtextxy(72*x,17*y,"V");
 /*************************************************************************/
 outtextxy(2*x,21*y," Let Ui = Vj  as shown above. Then again U, U1, ..., Ui, Vj-1,
                          ... V1, U  is");
 outtextxy(2*x,22*y," a cycle. This contradicts with the definition of the tree.");
 outtextxy(2*x,23*y," Therefore there exist exactly one path between U & V. ");
 /*************************************************************************/
 Pause(30*x,24*y);
 closegraph();
 videoinit();
}
```

```
/* PROGRAM  : example43.c
   AUTHOR       : Atilla BAKAN
   DATE         : Apr. 16, 1990
   REVISED      : Apr. 16, 1990

   DESCRIPTION : This program draws the example graph for the proof of
                 theorem 4.3.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"


#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions        */
static void init_graph   (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void graph        (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;



/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /***********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
     }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
     }
  forecolor = WHITE;
  }
```

```
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>PRESS A KEY TO CONTINUE<<");
   if(waitkey()==ESC) {
      closegraph();
      videoinit();
      exit(0);
    }
  }



/*************************************************************************/
/* main routine that calls  graph routine                                */
/*************************************************************************/
void main()
{
  graph();
}
```

```
/*************************************************************************/
/* This routine gives examples of trees and some graphs that are not trees.      */
/*************************************************************************/
 void graph(void)
 {
  /************************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-3");
  /************************************************************************/
  pieslice(25*x,6*y,0,359,2);
  pieslice(30*x,8*y,0,359,2);
  pieslice(30*x,12*y,0,359,2);
  pieslice(25*x,14*y,0,359,2);
  pieslice(25*x,10*y,0,359,2);
  pieslice(35*x,10*y,0,359,2);
  pieslice(30*x,10*y,0,359,2);
  pieslice(60*x,10*y,0,359,2);
  pieslice(35*x,6*y,0,359,2);
  pieslice(55*x,6*y,0,359,2);
  pieslice(60*x,8*y,0,359,2);
  pieslice(60*x,12*y,0,359,2);
  pieslice(55*x,10*y,0,359,2);
  pieslice(65*x,10*y,0,359,2);
  pieslice(65*x,6*y,0,359,2);
  moveto(25*x,6*y);  lineto(30*x,8*y);
  lineto(30*x,12*y); lineto(25*x,14*y);
  moveto(25*x,10*y); lineto(35*x,10*y);
  moveto(30*x,8*y);  lineto(35*x,6*y);
  moveto(55*x,6*y);  lineto(60*x,8*y);  lineto(60*x,12*y);
  moveto(55*x,10*y); lineto(65*x,10*y);
  moveto(60*x,8*y);  lineto(65*x,6*y);
  outtextxy(30*x,5*y,"T");
  outtextxy(60*x,5*y,"T'");
```

```
/*************************************************************************/
outtextxy(4*x,16*y,"Now consider these graphs. The graph T' has k vertices and
                    still is a tree.");
outtextxy(4*x,17*y,"By the induction assumption T' has  k - 1 edges. But then
                    T has k edges.");
outtextxy(4*x,19*y,"Therefore, by induction the theorem holds for all positive
                    integers n.");
/*************************************************************************/
Pause(30*x,24*y);
closegraph();
videoinit();
}
```

```
/* PROGRAM   : examp44.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 16, 1990
   REVISED   : Apr. 16, 1990

   DESCRIPTION : This routine draws the example graph for the proof of
                 the first part of theorem 4.4.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"


#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph   (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void graph        (void);


/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************/
/* This fuction initializes the necessary graphical routines */
/*************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /*************************************************************/
  settext();
  /*************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
     }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
     }
  forecolor = WHITE;
  }
```

```
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>PRESS A KEY TO CONTINUE<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
   }
 }



/*************************************************************************/
/* main routine that calls graph routine                                 */
/*************************************************************************/
void main()
{
  graph();
}
```

```
/*************************************************************************/
/* This routine gives examples of trees and some graphs that are not trees.      */
/*************************************************************************/
 void graph(void)
 {
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-4");
  /*************************************************************************/
  pieslice(20*x,6*y,0,359,2);
  pieslice(25*x,6*y,0,359,2);
  pieslice(35*x,4*y,0,359,2);
  pieslice(55*x,4*y,0,359,2);
  pieslice(65*x,6*y,0,359,2);
  pieslice(70*x,6*y,0,359,2);
  pieslice(35*x,8*y,0,359,2);
  pieslice(55*x,8*y,0,359,2);
  pieslice(65*x,6*y,0,359,2);
  moveto(20*x,6*y); lineto(25*x,6*y);
  lineto(35*x,4*y); lineto(55*x,4*y);
  lineto(65*x,6*y); lineto(70*x,6*y);
  moveto(25*x,6*y); lineto(35*x,8*y);
  lineto(55*x,8*y); lineto(65*x,6*y);
  moveto(35*x,4*y); lineto(55*x,8*y);
  outtextxy(18*x,6*y,"U");
  outtextxy(72*x,6*y,"V");
  outtextxy(2*x,15*y," Observe that U and V need not be part of the cycle. Hence,
                      if T is connected");
  outtextxy(2*x,16*y," and acyclic, then there will be exactly one path between any
                       pair of vertices.");
  Pause(30*x,24*y);
  closegraph();
  videoinit();
 }
```

```
/* PROGRAM   : examp45.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 16, 1990
   REVISED   : Apr. 16, 1990

   DESCRIPTION : This routine draws the example graph for the second
                 part of the proof for the theorem 4.5.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cximou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)        /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name         name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
   #define ffblk          FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void graph         (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
     }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
     }
  forecolor = WHITE;
  }
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,47*y/2,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,47*y/2,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,47*y/2,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,47*y/2,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*---------------------------------------------------------------------*/
/* This function sets the text default values                          */
/*---------------------------------------------------------------------*/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*---------------------------------------------------------------------*/
/* Equivalent of press_a_key function for graphics screen              */
/*---------------------------------------------------------------------*/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
 }
/*---------------------------------------------------------------------*/
/* main routine that calls exer routine                                */
/*---------------------------------------------------------------------*/
void main()
{
  graph();
}
```

```
/*************************************************************************/
/* This routine gives examples of trees and some graphs that are not trees.      */
/*************************************************************************/
 void graph(void)
 {
  /************************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-5");
  /************************************************************************/
  pieslice(20*x,4*y,0,359,2);
  pieslice(25*x,4*y,0,359,2);
  pieslice(35*x,6*y,0,359,2);
  pieslice(45*x,6*y,0,359,2);
  pieslice(50*x,4*y,0,359,2);
  pieslice(55*x,3*y,0,359,2);
  pieslice(58*x,6*y,0,359,2);
  pieslice(52*x,8*y,0,359,2);
  pieslice(20*x,8*y,0,359,2);
  pieslice(25*x,8*y,0,359,2);
  moveto(20*x,4*y); lineto(25*x,4*y);
  lineto(35*x,6*y); lineto(45*x,6*y);
  lineto(50*x,4*y); lineto(55*x,3*y);
  moveto(20*x,8*y); lineto(25*x,8*y); lineto(35*x,6*y);
  moveto(45*x,6*y); lineto(52*x,8*y);
  moveto(50*x,4*y); lineto(58*x,6*y);
  outtextxy(33*x,6*y,"U");
  outtextxy(47*x,6*y,"V");
  /************************************************************************/
  outtextxy(2*x,10*y,"Let e = { U, V } be an edge in T.");
  /************************************************************************/
  Pause(30*x,24*y);
  setcolor(backcolor);
  bar(29*x,23*y,70*x,49*y/2);
```

```
setcolor(forecolor);
outtextxy(40*x,5*y,"e");
outtextxy(2*x,10*y,"Let e = { U, V } be an edge in T. Since there is only one path
                    from U tu V,");
outtextxy(2*x,11*y,"removing e from the graph T disconnects T (as you see)");
/*************************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
moveto(35*x,6*y); lineto(45*x,6*y); /* delete e */
setcolor(forecolor);
setlinestyle(3,0,1);
moveto(35*x,6*y); lineto(45*x,6*y);
/*************************************************************************/
outtextxy(2*x,11*y,"removing e from the graph T disconnects T (as you see),
                    creating a graph T'");
outtextxy(2*x,12*y,"with two components and one less edge. Let T1 and T2 these
                    componets.");
/*************************************************************************/
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
outtextxy(22*x,9*y,"T1");
outtextxy(50*x,9*y,"T2");
outtextxy(2*x,13*y,"Suppose T1 has n1 vertices and m1 edges, and suppose T2
                    has n2 vertices and");
outtextxy(2*x,14*y,"m2 edges  The number of vertices in T' is n = n1 + n2, and the
                    number of edges");
outtextxy(2*x,15*y,"in T' is m1 + m2. Both T1 and T2 are subgraphs of T, so any
                    pair of vertices in");
outtextxy(2*x,16*y,"each connected by exactly one path. Applying the inductive
                    hypothesis P(n), we");
outtextxy(2*x,17*y,"conclude that  m1 = n1 - 1  and  m2 = n2  - 1. Now T' has one
                     less edge than T.");
outtextxy(2*x,18*y,"Then ");
```

```
 outtextxy(14*x,20*y,"m = ( the number of edges in T') + 1");
 outtextxy(14*x,21*y," = m1 + m2 + 1 = n1 - 1 + (n2 - 1) + 1");
 outtextxy(14*x,22*y," = n1 + n2 - 1 = n - 1");
 outtextxy(2*x,23*y,"which completes the proof.");
 /*************************************************************************/
 Pause(30*x,24*y);
 closegraph();
 videoinit();
}
```

```
/* PROGRAM      : q411.c
   AUTHOR       : Atilla BAKAN
   DATE         : Mar. 20, 1990
   REVISED      : Apr. 16, 1990

   DESCRIPTION : This program contains the first exercise about properties
                 of trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer          (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
   settext();
   if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
     }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
     }
   forecolor = WHITE;
   }
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
```

```
    case 'N': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}


/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>PRESS A KEY TO CONTINUE<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
   }
 }
```

```
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}


/*************************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                          */
/*************************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  1");
   /*************************************************************************/
   pieslice(10*x,7*y,0,359,2);
   pieslice(25*x,5*y,0,359,2);
   pieslice(35*x,5*y,0,359,2);
   pieslice(25*x,9*y,0,359,2);
   pieslice(35*x,9*y,0,359,2);
   pieslice(45*x,7*y,0,359,2);
   pieslice(55*x,7*y,0,359,2);
   pieslice(68*x,5*y,0,359,2);
   pieslice(75*x,7*y,0,359,2);
   pieslice(68*x,9*y,0,359,2);
   pieslice(80*x,7*y,0,359,2);
   moveto(25*x,5*y);  lineto(35*x,5*y);
   lineto(35*x,9*y); lineto(25*x,9*y);  lineto(25*x,5*y);
```

```
moveto(68*x,5*y);  lineto(75*x,7*y); lineto(68*x,9*y);
outtextxy(9*x,12*y,"(a)");
outtextxy(29*x,12*y,"(b)");
outtextxy(49*x,12*y,"(c)");
outtextxy(74*x,12*y,"(d)");
/*************************************************************************/
outtextxy(18*x,2*y,"Which one of the following graphs is a tree ?");
outtextxy(18*x,1[illegible],"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,18*y,"    Please type a,b,c, or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
   setcolor(backcolor);
   bar(50*x,17*y,88*x,19*y);
   setcolor(forecolor);
}
switch (Ch)        {
 case 'a': outtextxy(50*x,18*y,"a");
      outtextxy(55*x,18*y,"Correct. Congratulations.");
      break;
 case 'b': outtextxy(50*x,18*y,"b");
      outtextxy(55*x,18*y,"Sorry! It cannot be a tree,");
      outtextxy(55*x,19*y,"because it has a cycle.");
      outtextxy(55*x,20*y,"The correct answer was 'a'.");
      break;

 case 'c': outtextxy(50*x,18*y,"c");
       outtextxy(55*x,18*y,"Sorry! It cannot be a tree,");
       outtextxy(55*x,19*y,"because it is not connected.");
       outtextxy(55*x,20*y,"The correct answer was 'a'.");
       break;

 case 'd': outtextxy(50*x,18*y,"d");
```

```
            outtextxy(55*x,18*y,"Sorry! It cannot be a tree,");
            outtextxy(55*x,19*y,"because it is not connected.");
            outtextxy(55*x,20*y,"The correct answer was 'a'.");
            break;

        default  : break;
        }
        Pause(31*x,24*y);
        closegraph();
}
```

```
/* PROGRAM  : q412.c
   AUTHOR   : Atilla BAKAN
   DATE     : Mar. 20, 1990
   REVISED  : Mar. 20, 1990

   DESCRIPTION : This program contains the second exercise about properties
                 of trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"


#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer          (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
  }
```

```
/**:  :*****************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
```

```
    case 'N': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}

/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}

/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
 }
```

```
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
/*************************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                          */
/*************************************************************************/
static void exer(void)
{
   int backcolor = 7;
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  2");
   /*********************************************************************/
   pieslice(30*x,5*y,0,359,2);
   pieslice(40*x,5*y,0,359,2);
   pieslice(50*x,5*y,0,359,2);
   pieslice(60*x,5*y,0,359,2);
   pieslice(30*x,9*y,0,359,2);
   pieslice(40*x,9*y,0,359,2);
   pieslice(50*x,9*y,0,359,2);
   pieslice(60*x,9*y,0,359,2);
   moveto(30*x,5*y);  lineto(40*x,9*y); lineto(50*x,5*y); lineto(60*x,9*y);
   moveto(30*x,9*y);  lineto(40*x,5*y); lineto(50*x,9*y); lineto(60*x,5*y);
   /*********************************************************************/
   outtextxy(32*x,2*y,"Is this graph a tree ?");
   outtextxy(18*x,18*y,"Enter your choice here --->");
   Ch = getch ();
```

```
if(Ch==ESC) confirm_graph_exit();
 while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
    outtextxy(48*x,18*y,"   Please type y or n");
    Ch = getch ();
    if(Ch==ESC) confirm_graph_exit();
    if((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))
    setcolor(backcolor);
    bar(50*x,17*y,88*x,19*y);
    setcolor(forecolor);
 }
 switch (Ch)        {
  case 'y': outtextxy(50*x,18*y,"y");
       outtextxy(55*x,18*y,"No. If you examine the graph");
       outtextxy(55*x,19*y,"carefully, you'll see that it");
       outtextxy(55*x,20*y,"is not connected.");
       break;
  case 'Y'. outtextxy(50*x,18*y,"Y");
       outtextxy(55*x,18*y,"No. If you examine the graph");
       outtextxy(55*x,19*y,"carefully, you'll see that it");
       outtextxy(55*x,20*y,"is not connected.");
       break;

  case 'n': outtextxy(50*x,18*y,"n");
       outtextxy(55*x,18*y,"You are right! Conratulations.");
       break;

  case 'N': outtextxy(50*x,18*y,"N");
       outtextxy(55*x,18*y,"You are right! Conratulations.");
       break;

  default  : break;
  }
  Pause(30*x,24*y);
  closegraph();
}
```

```
/* PROGRAM  : q413.c
   AUTHOR   : Atilla BAKAN
   DATE     : Mar. 20, 1990
   REVISED  : Mar. 20, 1990

   DESCRIPTION : This program contains the third exercise about properties
                 of trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions    */
static void exer          (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;

/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
   settext();
   if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
  }
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
```

```
    case 'N': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}

/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
 }
```

```
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}


/*************************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                          */
/*************************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE  3");
   /*********************************************************************/
   pieslice(35*x,5*y,0,359,2);
   pieslice(55*x,5*y,0,359,2);
   pieslice(35*x,9*y,0,359,2);
   pieslice(55*x,9*y,0,359,2);
   moveto(35*x,5*y);  lineto(55*x,5*y); lineto(55*x,9*y); lineto(35*x,5*y);
   moveto(35*x,9*y);  lineto(55*x,5*y); lineto(55*x,9*y); lineto(35*x,9*y);
   /*********************************************************************/
   outtextxy(32*x,2*y,"Is this graph a tree ?");
   outtextxy(18*x,18*y,"Enter your choice here --->");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
      outtextxy(48*x,18*y,"   Please type y or n");
```

```
    Ch = getch ();
    if(Ch==ESC) confirm_graph_ex t();
    if((Ch == 'y') || (Ch == 'n') || (Ch  == 'Y') || (Ch == 'N'))
    setcolor(backcolor);
    bar(50*x,17*y,88*x,19*y);
    setcolor(forecolor);
  }
  switch (Ch)        {
   case 'y': outtextxy(50*x,18*y,"y");
        outtextxy(55*x,18*y,"No. If you examine the graph");
        outtextxy(55*x,19*y,"carefully, you'll see that it");
        outtextxy(55*x,20*y,"has a cycle.");
        break;
   case 'Y': outtextxy(50*x,18*y,"Y");
        outtextxy(55*x,18*y,"No. If you examine the graph");
        outtextxy(55*x,19*y,"carefully, you'll see that it");
        outtextxy(55*x,20*y,"has a cycle.");
        break;

   case 'n': outtextxy(50*x,18*y,"n");
        outtextxy(55*x,18*y,"You are right! Conratulations.");
        break;

   case 'N': outtextxy(50*x,18*y,"N");
        outtextxy(55*x,18*y,"You are right! Conratulations.");
        break;

   default  : break;
   }
   Pause(30*x,24*y);
   closegraph();
}
```

```
/* PROGRAM  : q414.c
   AUTHOR   : Atilla BAKAN
   DATE     : Mar. 20, 1990
   REVISED  : Mar. 20, 1990

   DESCRIPTION : This program contains the fourth exercise about properties
                 of trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlmou.h"
#include "cxlkey.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer          (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;

/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
   settext();
   if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
      }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
      }
  forecolor = WHITE;
  }
```

```
/*********************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
```

```
    case 'N': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}




/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
 }
```

```
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}


/*************************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                          */
/*************************************************************************/
static void exer(void)
{
   char Ch;

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXERCISE 4");
   /*************************************************************************/
   pieslice(15*x,7*y,0,359,2);
   pieslice(20*x,7*y,0,359,2);
   pieslice(25*x,7*y,0,359,2);
   pieslice(45*x,7*y,0,359,2);
   pieslice(50*x,5*y,0,359,2);
   pieslice(50*x,7*y,0,359,2);
   pieslice(70*x,7*y,0,359,2);
   pieslice(75*x,7*y,0,359,2);
   pieslice(80*x,7*y,0,359,2);
   moveto(15*x,7*y); lineto(25*x,7*y);
   moveto(45*x,7*y); lineto(50*x,7*y); lineto(50*x,5*y);
   moveto(70*x,7*y); lineto(80*x,7*y);
   outtextxy(29*x/2,13*y/2,"A");
```

```
outtextxy(39*x/2,13*y/2,"B");
outtextxy(49*x/2,13*y/2,"C");
outtextxy(43*x,7*y,"C");
outtextxy(51*x,7*y,"B");
outtextxy(49*x,9*y/2,"A");
outtextxy(139*x/2,13*y/2,"C");
outtextxy(149*x/2,13*y/2,"A");
outtextxy(159*x/2,13*y/2,"B");
outtextxy(19*x,9*y,"(I)");
outtextxy(46*x,9*y,"(II)");
outtextxy(72*x,9*y,"(III)");
/*************************************************************************/
outtextxy(18*x,11*y,"Which one of the following statements is  true ?");
outtextxy(20*x,13*y,"a)  (I) and (III) are the same.");
outtextxy(20*x,14*y,"b)  (I) and (II) are the same.");
outtextxy(20*x,15*y,"c)  All three are the same.");
outtextxy(20*x,16*y,"d)  All three are distinct.");
outtextxy(18*x,18*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,18*y,"   Please type a,b,c, or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
   setcolor(backcolor);
   bar(50*x,17*y,88*x,19*y);
   setcolor(forecolor);
}
switch (Ch)        {
 case 'a': outtextxy(50*x,18*y,"a");
      outtextxy(55*x,18*y,"Sorry, that's not true!");
      outtextxy(55*x,19*y,"because, they do not have");
      outtextxy(55*x,20*y,"the same set of edges. For");
      outtextxy(55*x,21*y,"example {A,C} is an edge of");
      outtextxy(55*x,22*y,"(III) but not of the tree (I)");
```

```
        outtextxy(55*x,23*y,"The answer is 'b'.");
        break;

    case 'b': outtextxy(50*x,18*y,"b");
        outtextxy(55*x,18*y,"Correct. You are doing fine!");
        break;

    case 'c': outtextxy(50*x,18*y,"c");
        outtextxy(55*x,18*y,"No. Because (I) and (III)");
        outtextxy(55*x,19*y,"do not have the same set of");
        outtextxy(55*x,20*y,"edges. For example {A,C} is");
        outtextxy(55*x,21*y,"an edge of (III) but not of (I)");
        outtextxy(55*x,22*y,"The answer is 'b'.");
        break;

    case 'd': outtextxy(50*x,18*y,"d");
        outtextxy(55*x,18*y,"No. Because, although their");
        outtextxy(55*x,19*y,"topology are different, (I)");
        outtextxy(55*x,20*y,"and (II) both have the same");
        outtextxy(55*x,21*y,"set of edges, namely, {A, B}");
        outtextxy(55*x,22*y,"and {B, C}.");
        outtextxy(55*x,23*y," So, the answer is 'b'");
        break;

    default : break;
    }
    Pause(30*x,24*y);
    closegraph();
}
```

```
/* PROGRAM  : root.c
   AUTHOR       : Atilla BAKAN
   DATE         : Feb. 4, 1990
   REVISED      : Apr. 17, 1990

   DESCRIPTION : This program contains the tutorial for rooted trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/
/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions        */
static void add_shadow   (void);
static void confirm_quit  (void);
static void disp_sure_msg (void);
static void error_exit   (int errnum);
static void initialize   (void);
static void move_window   (int nsrow, int scol);
static void normal_exit   (void);
static void Pageup       (void);
static void Pagedown     (void);
static void press_a_key   (int wrow);
static void pre_help     (void);
static void quit_window   (void);
static void restore_cursor(void);
static void short_delay   (void);
static void size_window   (int nerow,int necol);

/* Tutorial procedures             */
static void rooted_trees  (void);
static void definition_4_2_1(void);
static void definition_4_2_2(void);
static void example_4_2_1 (void);
static void example_4_2_2 (void);
static void example_4_2_3 (void);
static void P1           (void);
static void P2           (void);
static void P3           (void);
static void P4           (void);
static void P5           (void);
static void P6           (void);
static void P7           (void);
static void P8           (void);
```

```
static void P9          (void);
static void P10         (void);
static void P11         (void);
static void theorem_4_6  (void);
static void proof_4_6    (void);
static void exercises    (void);
static void exer1        (void);
static void exer2        (void);
static void exer3        (void);
static void exer4        (void);
```

```
/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
static int *savescrn,crow,ccol;
static WINDOW w[10];
static char ssan[10];
```

```
/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* error message table                                                   */
/*************************************************************************/
static char *error_text[]= {
   NULL,  /* errnum = 0, no error   */
   NULL,  /* errnum == 1, windowing error */
   "Syntax:  CXLDEMO [-switches]\n\n"
      "\t -c = CGA snow elimination\n"
      "\t -b = BIOS screen writing\n"
      "\t -m = force monochrome text attributes",
   "Memory allocation error"
};
```

```
/*************************************************************************/
/* miscellaneous defines                                                 */
/*************************************************************************/
#define SHORT_DELAY 18
#define H_WINTITLE  33
```

```
/*************************************************************************/
/* this function will add a shadow to the active window                  */
/*************************************************************************/
static void add_shadow(void)
{
   wshadow(LGREY|_BLACK);
}
```

```
/*************************************************************************/
/* this function pops open a window and confirms that the user really    */
/* wants to quit the demo.  If so, it terminates the demo program.       */
/*************************************************************************/
static void confirm_quit(void)
{
  struct _onkey_t *kblist;

  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
  add_shadow();
  wputs("\n Quit demo, are you sure? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y') normal_exit();
  wclose();
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
}
```

```
/*************************************************************************/
/* this function is called by the pull-down demo for a prompt            */
/*************************************************************************/
static void disp_sure_msg(void)
{
  wprints(0,2,WHITE|_BLUE,"Are you sure?");
}
```

```
/*************************************************************************/
/* this function handles abnormal termination.  If it is passed an       */
/* error code of 1, then it is a windowing system error.  Otherwise      */
/* the error message is looked up in the error message table.            */
/*************************************************************************/
static void error_exit(int errnum)
{
  if(errnum) {
    printf("\n%s\n",(errnum==1)?werrmsg():error_text[errnum]);
          exit(errnum);
  }
}
/*************************************************************************/
/* this function initializes CXL's video, mouse, keyboard, and help systems */
/*************************************************************************/
static void initialize(void)
{
  /* initialize the CXL video system and save current screen info */
  videoinit();
  readcur(&crow,&ccol);
  if((savescrn=ssave())==NULL) error_exit(3);
  /* if mouse exists, turn on full mouse support */
  if(msinit()) {
    mssupport(MS_FULL);
    msgotoxy(12,49);
  }
  /* attach [Alt-X] to the confirm_quit() function */
  setonkey(0x2d00,confirm_quit,0);
  /* attach [Ctrl Pageup] to the Pageup() function */
  setonkey(0x8400,Pageup,0);
  /* attach [Ctrl Pagedown] to the Pagedown() function */
  setonkey(0x7600,Pagedown,0);
  /* initialize help system, help key = [F1] */
   whelp def("CXLDEMO.HLP",0x3b00,YELLOW|_RED,LRED|_RED,
          WHITE|_RED,RED|_LGREY ,pre_help);
}
```

```
/*************************************************************************/
/* this function is called anytime to switch back to previous window.    */
/*************************************************************************/
static void Pageup(void)
{
   static WINDOW handle;

   handle = whandle();
   wactiv(handle - 1);
}




/*************************************************************************/
/* this function is called anytime to switch back to next window.        */
/*************************************************************************/
static void Pagedown(void)
{
   static WINDOW handle;

   handle = whandle();
   wactiv(handle + 1);
}




/*************************************************************************/
 static void pre_help(void)
{
   add_shadow();
   setonkey(0x2d00,confirm_quit,0);
}
```

```
/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
  srestore(savescrn);
  gotoxy_(crow,ccol);
  if(_mouse) mshidecur();
  showcur();
  exit(0);
}
/*************************************************************************/
/* this function displays a pause message then pauses for a keypress     */
/*************************************************************************/
static void press_a_key(int wrow)
{
  register int attr1;
  register int attr2;

  attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
  attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
  wcenters(wrow,attr1,"Press a key");
  wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
  hidecur();
  if(waitkey()==ESC) confirm_quit();
  wcenters(wrow,attr1,"          ");
  wprints(wrow,0,attr2,"         ");
}
/*************************************************************************/
/* This routine causes short dealys during execution                     */
/*************************************************************************/
static void short_delay(void)
{
  delay_(SHORT_DELAY);
}
```

```
/*************************************************************************/
/* this function is called by the pull-down menu demo anytime            */
/* the  selection bar moves on or off the [Q]uit menu items.              */
/*************************************************************************/
static void quit_window(void)
{
   static WINDOW handle=0;

   if(handle) {
      wactiv(handle);
      wclose();
      handle=0;
   }
   else {
      handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
      wputs(" Quit takes you back to the\n demo program's main menu.");
   }
}

/*************************************************************************/
/* shows the cursor again if it has been hidden                           */
/*************************************************************************/
static void restore_cursor(void)
{
   wtextattr(WHITE|_MAGENTA);
   showcur();
}

/*************************************************************************/
/* enlarges or shrinks the windows                                        */
/*************************************************************************/
static void size_window(int nerow,int necol)
{
   wsize(nerow,necol);
   short_delay();
}
```

```
/*************************************************************************/
/* moves the active window to a given screen coordinates                 */
/*************************************************************************/
static void move_window(int nsrow,int nscol)
{
  if(wmove(nsrow,nscol)) error_exit(1);
  short_delay();
}
/*************************************************************************/
/* this routine that calls rooted_trees routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void P1()
{
  wcloseall();
  rooted_trees();
}
/*************************************************************************/
/* this routine that calls definition 4-2-1 routine whenever Pageup or   */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P2()
{
  wcloseall();
  definition_4_2_1();
}

/*************************************************************************/
/* this routine that calls definition 4-2-2  routine whenever Pageup or  */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P3()
{
  wcloseall();
  example_4_2_1();
}
```

```
/*************************************************************************/
/* this routine that calls definition 4-2-2 routine whenever Pageup or   */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P4()
{
  wcloseall();
  definition_4_2_2();
}
/*************************************************************************/
/* this routine that calls theorem_4_6  routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void P5()
{
  wcloseall();
  theorem_4_6();
}
/*************************************************************************/
/* this routine that calls example 4-2-3 routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void P6()
{
  wcloseall();
  example_4_2_3();
}
/*************************************************************************/
/* this routine that calls exercises  routine whenever Pageup or Pagedown */
/* keys are pressed.                                                     */
/*************************************************************************/
void P7()
{
  wcloseall();
  exercises();
}
```

```
/*************************************************************************/
/* this routine that calls exer1  routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void P8()
{
   wcloseall();
   exer1();
}
/*************************************************************************/
/* this routine that calls exer2  routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void P9()
{
   wcloseall();
   exer2();
}
/*************************************************************************/
/* this routine that calls exer3  routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void P10()
{
   wcloseall();
   exer3();
}
/*************************************************************************/
/* this routine that calls exer4  routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                     */
/*************************************************************************/
void P11()
{
   wcloseall();
   exer4();
}
```

```
/*************************************************************************/
/* this routine that calls example 4-2-2 routine whenever Pageup or Pagedown    */
/* keys are pressed.                                                            */
/*************************************************************************/
void P12()
{
   wcloseall();
   example_4_2_2();
}
/*************************************************************************/
/* main routine that calls rooted trees tutorial                                */
/*************************************************************************/
void main()
{
  initialize();
  rooted_trees();
}
```

```
/*************************************************************************/
/* This routine tells about the rooted trees. It gives the definitions,  */
/*  examples and theorems about the trees.                               */
/*************************************************************************/
static void rooted_trees(void)
{
  cclrscrn(LGREY|_BLUE);
  /*************************************************************************/
  /* attach [Pagedown] to the definition_4_2_1() function */
  setonkey(0x5100,P2,0);
  /*************************************************************************/
  if((w[1]=wopen(5,15,19,60,3,LCYAN|_BLACK,WHITE|_RED))==0)
       error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  In previous chapter we introduced  the special type of relation"
         ", tree, which is exceptionally useful in a variety of computer science"
         "  applications, and which is usually represented by its digraph. "
         " These relations are essential for the construction of data bases"
         " and language compilers, to name just two important areas. Now we"
         " will talk about a type of tree, which is called rooted tree, because"
         " of its appearance. Let's illustrate this with an example. ");
  press_a_key(12);
  wslide(0,30);
  wslide(0,0);
  /*************************************************************************/
  if((w[2]=wopen(13,10,22,70,3,LCYAN|_BROWN,WHITE|_GREEN))==0)
       error_exit(1);
  wtitle("[Rooted Trees - Example]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw("  Although we are talking about its use in computer science"
         " applications, for the sake of simplicity our example will be "
         " from daily life. We all know that people have always been "
         " interested in learning about the descendants of historically"
```

```
            " important individuals. To assist in these investigations a"
            " genealogical chart is often drawn. Here is an example.");
  press_a_key(7);
  short_delay();
  wcloseall();
  /*************************************************************************/
  spawnl(P_WAIT,"examp421.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  short_delay();
  definition_4_2_1();
}
```

```
/*************************************************************************/
/* This routine gives the definitions of rooted trees                    */
/*************************************************************************/
void definition_4_2_1(void)
 {
  /************************************************************************/
  /* attach [Pageup] to the rooted_trees() function */
  setonkey(0x4900,P1,0);
  /************************************************************************/
  /* attach [Pagedown] to the example_4_2_1() function */
  setonkey(0x5100,P3,0);
 /************************************************************************/
  if((w[1]=wopen(5,20,9,60,3,LCYAN|_RED,WHITE|_GREEN))==0) error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n It is time to see some definitions... ");
  press_a_key(2);
  short_delay();
  wclose();
   if((w[2]=wopen(1,20,25,58,3,LCYAN|_BLACK,RED|_LGREY))==0)
        error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_CYAN|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n Definition_4_2_1 : Rooted tree\n\n");
  wputsw(" Given a digraph T, is a rooted tree satisfying the following "
      " conditions :");
  wputs("\n\n 1. The underlying graph of T is connected\n\n");
  wputsw(" 2. There exist exactly one vertex R of T such that the indegree"
      "  of R is 0 and the indegree of any other vertex is 1.");
  wputs("\n\n This vertex R is called the root of the rooted tree.");
  press_a_key(22);
  wslide(0,0);
  example_4_2_1();
}
```

```
/*************************************************************************/
/* This routine gives an example about converting a suitable graph into a */
/* rooted tree.                                                           */
/*************************************************************************/
void example_4_2_1(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the definition_4_2_1() function */
  setonkey(0x4900,P2,0);
  /***********************************************************************/
  /* attach [Pagedown] to the definition_4_2_2() function */
  setonkey(0x5100,P4,0);
 /************************************************************************/
  if((w[3]=wopen(5,20,9,60,3,LCYAN|_RED,RED|_LGREY))==0) error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n And now, how about an example ?");
  press_a_key(2);
  wslide(0,39);
  short_delay();
  wcloseall();
  /***********************************************************************/
  spawnl(P_WAIT,"examp422.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  /***********************************************************************/
  if((w[4]=wopen(10,20,14,60,3,LCYAN|_RED,RED|_LGREY))==0) error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n     Some more definitions ? ");
  press_a_key(2);
  wclose();
  short_delay();
  definition_4_2_2();
}
```

```
/*************************************************************************/
/*  This routine gives the definitions of concepts related with rooted trees     */
/*  (i.e. parent, child, ancestor, descendent, etc.                              */
/*************************************************************************/
 void definition_4_2_2(void)
  {
   /*************************************************************************/
   /* attach [Pageup] to the example_4_2_1() function */
   setonkey(0x4900,P3,0);
   /*************************************************************************/
   /* attach [Pagedown] to the example_4_2_2() function */
   setonkey(0x5100,P12,0);
   /*************************************************************************/
   if((w[5]=wopen(1,20,25,58,3,LCYAN|_BLACK,RED|_LGREY))==0)
              error_exit(1);
   wtitle("[Rooted Trees - Definition_4_2_2]",TCENTER,_CYAN|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputs("\n Definition_4_2_2 : Rooted tree\n\n");
   wputs(" In a rooted tree : ");
   wputs("\n\n 1. If (u,v) is an edge then\n\n");
   wputs("   a) u is the parent of v.\n\n");
   wputs("   b) v is the child  of u.\n\n");
   wputs(" 2. If there exist a simple directed path from u to v, then :\n\n");
   wputs("   a) u is ancestor   of v.\n\n");
   wputs("   b) v is descendent of u.\n\n");
   wputs(" 3. A terminal vertex ( or leaf ) has no children. \n\n");
   wputs(" 4. An internal vertex has children.");
   press_a_key(22);
   wslide(0,39);
   short_delay();
   example_4_2_2();
}
```

```
/*************************************************************************/
/* This routine gives an example graph to show  the relations that are in  */
/* in the definition_4_2_2.                                                */
/*************************************************************************/
void example_4_2_2(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the definition 4-2-2 function */
  setonkey(0x4900,P4,0);
  /*************************************************************************/
  /* attach [Pagedown] to the theorem_4_6() function */
  setonkey(0x5100,P5,0);
 /*************************************************************************/
  if((w[6]=wopen(10,20,14,60,3,LCYAN|_RED,RED|_LGREY))==0) error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n      To see an example ");
  add_shadow();
  press_a_key(2);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"examp423.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  theorem_4_6();
}
```

```
/*************************************************************************/
/* This routine gives the theorem about the rooted trees. Besides, if the */
/* user wants, it gives the proof for this theorem.                        */
/*************************************************************************/
 void theorem_4_6(void)
 {
  struct _onkey_t *kblist;

  /*************************************************************************/
  /* attach [Pageup] to the example 4-2-2 function */
  setonkey(0x4900,P12,0);
  /*************************************************************************/
  /* attach [Pagedown] to the example_4_2_3() function */
  setonkey(0x5100,P6,0);
 /*************************************************************************/
  if((w[1]=wopen(1,4,13,71,3,LCYAN|_BLACK,RED|_LGREY))==0) error_exit(1);
  wtitle("[Rooted Trees - Theorem_4_6]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("Theorem_4_6\n\n");
  wputs("In a rooted tree with root R\n\n");
  wputs(" (a) The number of vertices = number of arcs + 1.\n\n");
  wputs(" (b) There are no directed cycles in T.\n\n");
  wputsw(" (c) There is exactly one simple directed path from R to every "
       " other vertex.");
  press_a_key(10);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,20,13,55,0,BROWN|_CYAN,RED|_BLACK)) error_exit(1);
  add_shadow();
  wputs("\n Do you want to see the proof? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y') {
    wclose();
    proof_4_6();
```

```
  }
  else wclose();
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
  wclose();
  /*************************************************************************/
  if((w[5]=wopen(10,20,14,60,3,LCYAN|_RED,RED|_LGREY))==0) error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n    One more example... ");
  press_a_key(2);
  wclose();
  short_delay();
  /*************************************************************************/
  example_4_2_3();
}
```

```
/*************************************************************************/
/* This routine gives the proof to the theorem_4_6.                      */
/*************************************************************************/
static void proof_4_6(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the example 4-2-2 function */
  setonkey(0x4900,P12,0);
  /***********************************************************************/
  /* attach [Pagedown] to the example_4_2_3() function */
  setonkey(0x5100,P6,0);
 /************************************************************************/
  if((w[2]=wopen(14,4,24,71,3,LCYAN|_RED,WHITE|_CYAN))==0) error_exit(1);
  wtitle("[Theorem_4_6 - Proof]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw("  When the  directions on the directed edges are ignored and"
         "  T becomes a tree, the proofs of (a) and (b) follow immediately."
         "  Next we will show that there is a direct path from R to any other"
         "  vertex ");
  press_a_key(8);
  wclose();
}
```

```
/*************************************************************************/
/* This routine gives an example about given a graph finding out if it's */
/* a rooted tree.                                                        */
/*************************************************************************/
 void example_4_2_3(void)
  {
   register int *scrn;


   /*************************************************************************/
   /* attach [Pageup] to the theorem 4-6 function */
   setonkey(0x4900,P5,0);
   /*************************************************************************/
   /* attach [Pagedown] to the exercises() function */
   setonkey(0x5100,P7,0);
  /*************************************************************************/
   if((w[6]=wopen(1,10,8,70,3,LCYAN|_BLACK,RED|_LGREY))==0) error_exit(1);
   wtitle("[Example_4_2_3 ]",TCENTER,_CYAN|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputs("\n");
   wputsw(" Let A = { a, b, c, d, e, f, g, h, i, j } and let T = { (b, c),"
            " (b, a), (d, e), (d, f), (e, h), (f, g), (d, b), (g, i),"
            " (g, j) }. Show that T is a rooted tree, and identify the root.");
   short_delay();
   /*************************************************************************/
   if((w[6]=wopen(9,20,13,60,3,LCYAN|_RED,BLACK|_LGREY))==0)
             error_exit(1);
   wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputs("\n      To see the solution ");
   press_a_key(2);
   wclose();
   short_delay();
   /*************************************************************************/
```

```
  if((w[7]=wopen(9,10,23,70,3,LCYAN|_BLACK,RED|_LGREY))==0)
        error_exit(1);
  wtitle("[Example_4_2_3 - Solution]",TCENTER,_CYAN|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n");
  wputsw(" Since no paths begin at vertices a, c, h, i, and j, these vertices"
        " cannot be roots of a tree. There are no paths from vertices f, g, "
        " b, and e to vertex d, so we must eliminate these vertices as possible"
        " roots. Thus if T is a rooted tree, its root must be vertex d. It is "
        " easy to show that there is a path from d to every other vertex."
        " For example, the path d, f, g, i leads from d to i, since (d, f),"
        " (f, g), (g, i) are all in T. We draw the digraph of T, beginning"
        " with vertex d, and with edges shown downward. You will see the result"
        " in the following figure.  To see the figure ");
  press_a_key(12);
  wcloseall();
  /*************************************************************************/
  spawnl(P_WAIT,"examp424.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exercises();
}
```

```
/*************************************************************************/
/* This routine makes a small quiz about the rooted trees.               */
/*************************************************************************/
void exercises(void)
 {
  register int *screen;

  /*************************************************************************/
  /* attach [Pageup] to the example_4_2_3() function */
  setonkey(0x4900,P6,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exer1() function */
  setonkey(0x5100,P8,0);
  /*************************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw(" We have completed our presentation of this section. Are"
         " you ready for a pop quiz ? ");
  press_a_key(3);
  short_delay();
  wclose();
  if((screen=ssave())==NULL) error_exit(3); {
  /*************************************************************************/
   exer1();
  /* if mouse exists, turn on full mouse support */
  if(msinit()) {
    mssupport(MS_FULL);
    msgotoxy(12,49);
    }
   }
  srestore(screen);
 }
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise 4.2.1                      */
/*************************************************************************/
static void exer1(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the example_4_2_3() function */
  setonkey(0x4900,P6,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exer2() function */
  setonkey(0x5100,P9,0);
  /*************************************************************************/
  if((w[18]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("      Here is the first question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q421.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer2();
}
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise 4.2.2                      */
/*************************************************************************/
static void exer2(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the exer1() function */
  setonkey(0x4900,P8,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exer3() function */
  setonkey(0x5100,P10,0);
  /*************************************************************************/
  if((w[18]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("      Here is the second question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q422.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer3();
}
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise 4.2.3                      */
/*************************************************************************/
static void exer3(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the exer2() function */
  setonkey(0x4900,P9,0);
  /***********************************************************************/
  /* attach [Pagedown] to the exer4() function */
  setonkey(0x5100,P11,0);
  /***********************************************************************/
  if((w[18]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("      Here is the third question. "),
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q423.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  exer4();
}
```

```
/*****************************************************************************/
/* Dummy function to call the actual exercise 4.2.4                          */
/*********************** ***************************************************/
static void exer4(void)
{
  /*****************************************************************************/
  /* attach [Pageup] to the exer3() function */
  setonkey(0x4900,P10,0);
  /*****************************************************************************/
  if((w[18]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
              error_exit(1);
  wtitle("[Rooted Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("        Here is the forth question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"q424.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  wcloseall();
  normal_exit();
}
```

```
/* PROGRAM  : examp421.c
   AUTHOR   : Atilla BAKAN
   DATE     : Apr. 16, 1990
   REVISED  : Apr. 17, 1990

   DESCRIPTION : This routine draws the example graph for



   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer           (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error,
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherromsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
      }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
      }
  forecolor = WHITE;
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
   case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* This routine gives example of rooted trees.                           */
/*************************************************************************/
 void exer(void)
 {
  /************************************************************************/
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-2-1");
  /************************************************************************/
  moveto(16*x,4*y); lineto(6*x,6*y);
  moveto(18*x,4*y); lineto(18*x,6*y);
  moveto(20*x,4*y); lineto(30*x,6*y);
  moveto(4*x,8*y);  lineto(2*x,10*y);
  moveto(7*x,8*y);  lineto(9*x,10*y);
  moveto(18*x,8*y); lineto(18*x,10*y);
  moveto(30*x,8*y); lineto(28*x,10*y);
  moveto(33*x,8*y); lineto(39*x,10*y);
  moveto(16*x,12*y); lineto(12*x,14*y);
  moveto(20*x,12*y); lineto(24*x,14*y);
  outtextxy(15*x,5*y/2,"John");
  outtextxy(3*x,13*y/2,"Ed");
  outtextxy(15*x,13*y/2,"Judy");
  outtextxy(28*x,13*y/2,"Mary");
  outtextxy(2*x,21*y/2,"Sam");
  outtextxy(7*x,21*y/2,"Hal");
  outtextxy(16*x,21*y/2,"Liz");
  outtextxy(25*x,21*y/2,"Tom");
  outtextxy(36*x,21*y/2,"Albert");
  outtextxy(9*x,29*y/2,"Don");
  outtextxy(21*x,29*y/2,"Denise");
  /************************************************************************/
  outtextxy(3*x,17*y,"An example of these charts is drawn above, where for simplic-
                      ity only");
```

```
outtextxy(3*x,18*y,"first names are used. It is understood that the downward lines
                    lines");
outtextxy(3*x,19*y,"lines represent the ''is a parent of'' relationship.");
/*************************************************************************/
Pause(55*x,15*y);
setcolor(backcolor);
bar(54*x,14*y,MaxX-3*x/2,16*y);
setcolor(forecolor);
/*************************************************************************/
outtextxy(3*x,22*y,"This chart can also be represented by the next graph in which
                    vertices");
outtextxy(3*x,23*y,"represent individuals and edges begin at a parent and end at a
                    child.");
/*************************************************************************/
pieslice(71*x,4*y,0,359,2);      /* John  */
pieslice(59*x,6*y,0,359,2);      /* Ed    */
pieslice(71*x,6*y,0,359,2);      /* Judy  */
pieslice(83*x,6*y,0,359,2);      /* Mary  */
pieslice(52*x,10*y,0,359,2);     /* Sam   */
pieslice(62*x,10*y,0,359,2);     /* Hal   */
pieslice(71*x,10*y,0,359,2);     /* Liz   */
pieslice(81*x,10*y,0,359,2);     /* Tom   */
pieslice(89*x,10*y,0,359,2);     /* Albert */
pieslice(65*x,14*y,0,359,2);     /* Don   */
pieslice(77*x,14*y,0,359,2);     /* Denise */
moveto(71*x,4*y);  lineto(59*x,6*y);
moveto(71*x,4*y);  lineto(71*x,6*y);
moveto(71*x,4*y);  lineto(83*x,6*y);
moveto(59*x,6*y);  lineto(52*x,10*y);
moveto(59*x,6*y);  lineto(62*x,10*y);
moveto(71*x,6*y);  lineto(71*x,10*y);
moveto(83*x,6*y);  lineto(81*x,10*y);
moveto(83*x,6*y);  lineto(89*x,10*y);
moveto(71*x,10*y); lineto(65*x,14*y);
moveto(71*x,10*y); lineto(77*x,14*y);
outtextxy(69*x,5*y/2,"John");
```

```
    outtextxy(55*x,6*y,"Ed");
    outtextxy(65*x,6*y,"Judy");
    outtextxy(84*x,6*y,"Mary");
    outtextxy(49*x,21*y/2,"Sam");
    outtextxy(60*x,21*y/2,"Hal");
    outtextxy(66*x,10*y,"Liz");
    outtextxy(77*x,21*y/2,"Tom");
    outtextxy(83*x,21*y/2,"Albert");
    outtextxy(62*x,29*y/2,"Don");
    outtextxy(74*x,29*y/2,"Denise");
    /*************************************************************/
    Pause(30*x, 24*y);
    closegraph():
    videoinit();
}
```

```
/* PROGRAM  : examp422.c
   AUTHOR   : Atilla BAKAN
   DATE     : Apr. 16, 1990
   REVISED  : Apr. 17, 1990

   DESCRIPTION : This routine draws the example graph for



   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                     /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)          /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                     /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer           (void);


/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;

/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
      }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
      }
  forecolor = WHITE;
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
  case 'y': closegraph();
          videoinit();
          exit(0);
          break;
  case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
  case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
  case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* This routine gives example of rooted trees.                           */
/*************************************************************************/
 void exer(void)
 {
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE 4-2-2");
  /*************************************************************************/
  pieslice(15*x,4*y,0,359,2);   /* B */
  pieslice(25*x,7*y,0,359,2);   /* A */
  pieslice(15*x,10*y,0,359,2);  /* C */
  pieslice(27*x,10*y,0,359,2);  /* D */
  pieslice(35*x,7*y,0,359,2);   /* F */
  pieslice(40*x,4*y,0,359,2);   /* G */
  pieslice(30*x,4*y,0,359,2);   /* E */
  pieslice(40*x,10*y,0,359,2);  /* H */
  outtextxy(13*x,4*y,"B");
  outtextxy(26*x,13*y/2,"A");
  outtextxy(33*x,13*y/2,"F");
  outtextxy(15*x,11*y,"C");
  outtextxy(27*x,11*y,"D");
  outtextxy(42*x,10*y,"H");
  outtextxy(42*x,4*y,"G");
  outtextxy(28*x,4*y,"E");
  moveto(15*x,4*y); lineto(25*x,7*y); lineto(35*x,7*y);
  lineto(40*x,4*y); lineto(30*x,4*y);
  moveto(25*x,7*y); lineto(15*x,10*y); lineto(27*x,10*y);
  moveto(35*x,7*y); lineto(40*x,10*y);
  /*************************************************************************/
  outtextxy(7*x,14*y,"The graph above is a rooted tree with root A since");
  outtextxy(7*x,16*y,"(1) when the directions on the edges are ignored, the");
  outtextxy(7*x,17*y,"    resulting is a tree; and");
  outtextxy(7*x,19*y,"(2) A has indegree 0, and all the other vertices have");
```

```
outtextxy(7*x,20*y,"    indegree 1.");
outtextxy(7*x,22*y,"Now press any key to follow how we draw the usual tree.");
Pause(30*x, 24*y);
delay_(18);
/*************************************************************************/
pieslice(70*x,4*y,0,359,2);
outtextxy(68*x,7*y/2,"A");
delay_(18);
/*************************************************************************/
pieslice(60*x,7*y,0,359,2);
outtextxy(58*x,7*y,"C");
moveto(70*x,4*y); lineto(60*x,7*y);
delay_(18);
/*************************************************************************/
pieslice(70*x,7*y,0,359,2);
outtextxy(68*x,7*y,"F");
moveto(70*x,4*y); lineto(70*x,7*y);
delay_(18);
/*************************************************************************/
pieslice(80*x,7*y,0,359,2);
outtextxy(78*x,7*y,"B");
moveto(70*x,4*y); lineto(80*x,7*y);
delay_(18);
/*************************************************************************/
pieslice(55*x,9*y,0,359,2);
outtextxy(53*x,9*y,"D");
moveto(60*x,7*y); lineto(55*x,9*y);
delay_(18);
/*************************************************************************/
pieslice(65*x,9*y,0,359,2);
outtextxy(63*x,9*y,"H");
moveto(70*x,7*y); lineto(65*x,9*y);
delay_(18);
/*************************************************************************/
pieslice(75*x,9*y,0,359,2);
outtextxy(73*x,9*y,"G");
```

```
    moveto(70*x,7*y); lineto(75*x,9*y);
    delay_(18);
    /*************************************************************************/
    pieslice(70*x,11*y,0,359,2);
    outtextxy(68*x,11*y,"E");
    moveto(75*x,9*y); lineto(70*x,11*y);
    setcolor(forecolor);
    /*************************************************************************/
    Pause(30*x, 24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM   : examp423.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 16, 1990
   REVISED   : Apr. 17, 1990

   DESCRIPTION : This routine draws the example graph for the exercise 4_2_3



   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */
/*  Utility functions        */
static void init_graph   (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer          (void);
/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
 int in_the_exercise = 1;

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int x, y, MaxX, MaxY;

/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /***********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      }
  forecolor = WHITE;
}
```

```
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}



/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) {
     closegraph();
     videoinit();
     exit(0);
   }
  }



/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* This routine gives examples about the concepts in rooted trees        */
/*************************************************************************/
void exer()
{

   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE 4-2-3");
   /*************************************************************************/
   pieslice(45*x,4*y,0,359,2);   /* R */
   pieslice(35*x,7*y,0,359,2);   /* a */
   pieslice(55*x,7*y,0,359,2);   /* b */
   pieslice(45*x,10*y,0,359,2);  /* c */
   pieslice(65*x,10*y,0,359,2);  /* d */
   outtextxy(45*x,7*y/2,"R");
   outtextxy(32*x,7*y,"a");
   outtextxy(52*x,7*y,"b");
   outtextxy(42*x,10*y,"c");
   outtextxy(62*x,10*y,"d");
   moveto(35*x,7*y); lineto(45*x,4*y);
   lineto(55*x,7*y); lineto(45*x,10*y);
   moveto(55*x,7*y); lineto(65*x,10*y);
   /*************************************************************************/
   outtextxy(23*x,13*y,"b is the parent of c and d");
   outtextxy(23*x,15*y,"c and d are children of  b");
   outtextxy(23*x,17*y,"R is the ancestor");
   outtextxy(23*x,19*y,"c (or d) is the descendant of R");
   outtextxy(23*x,21*y,"a, c, d are leaves .");
   /*************************************************************************/
   Pause(30*x, 24*y);
   closegraph();
   videoinit();
}
```

```
/* PROGRAM  : examp424.c
   AUTHOR   : Atilla BAKAN
   DATE     : Apr. 16, 1990
   REVISED  : Apr. 16, 1990

   DESCRIPTION : This routine draws the example graph for exercise 4_2_4.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph   (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer          (void);
static void example       (void);
static void show_alg      (void);
static void step_solution (void);
static void compare_solutions (void);
static void confirm_exit  (void);


/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
 int in_the_exercise = 1;

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /***********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
```

```
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}



/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}

/*************************************************************************/
/* this routine illustrates constructing a rooted tree.                  */
/*************************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE 4-2-4");
   /*********************************************************************/
   pieslice(40*x,4*y,0,359,2);
   outtextxy(38*x,7*y/2,"d");
   Pause(30*x, 24*y);
   /*********************************************************************/
   pieslice(30*x,7*y,0,359,2);
```

```
outtextxy(28*x,7*y,"f");
moveto(40*x,4*y); lineto(30*x,7*y);
Pause(30*x, 24*y);
/*************************************************************************/
pieslice(40*x,7*y,0,359,2);
outtextxy(38*x,7*y,"b");
moveto(40*x,4*y); lineto(40*x,7*y);
Pause(30*x, 24*y);
/*************************************************************************/
pieslice(50*x,7*y,0,359,2);
outtextxy(48*x,7*y,"e");
moveto(40*x,4*y); lineto(50*x,7*y);
Pause(30*x, 24*y);
/*************************************************************************/
pieslice(55*x,9*y,0,359,2);
outtextxy(57*x,9*y,"h");
moveto(50*x,7*y); lineto(55*x,9*y);
Pause(30*x, 24*y);
/*************************************************************************/
pieslice(25*x,9*y,0,359,2);
outtextxy(23*x,9*y,"g");
moveto(30*x,7*y); lineto(25*x,9*y);
Pause(30*x, 24*y);
/*************************************************************************/
pieslice(20*x,11*y,0,359,2);
outtextxy(18*x,11*y,"i");
moveto(25*x,9*y); lineto(20*x,11*y);
Pause(30*x, 24*y);
/*************************************************************************/
pieslice(30*x,11*y,0,359,2);
outtextxy(28*x,11*y,"j");
moveto(25*x,9*y); lineto(30*x,11*y);
Pause(30*x, 24*y);
/*************************************************************************/
pieslice(35*x,9*y,0,359,2);
outtextxy(33*x,9*y,"a");
```

```
    moveto(40*x,7*y); lineto(35*x,9*y);
    Pause(30*x, 24*y);
    /*************************************************************************/
    pieslice(45*x,9*y,0,359,2),
    outtextxy(43*x,9*y,"c");
    moveto(40*x,7*y); lineto(45*x,9*y);
    Pause(30*x, 24*y);
    /*************************************************************************/
    outtextxy(4*x,18*y," A quick inspection of this graph shows that paths from vertex
                        d to every");
    outtextxy(4*x,19*y," other vertex are unique, and there are no paths from d to d.
                        Thus T is");
    outtextxy(4*x,20*y," a tree with root d.");
    /*************************************************************************/
    Pause(30*x, 24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM   : q421.c
   AUTHOR    : Atilla BAKAN
   DATE      : Mar. 21, 1990
   REVISED   : Apr. 17, 1990

   DESCRIPTION : This program contains the first exercise about rooted
                 trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example         (void);
static void show_alg        (void);
static void step_solution   (void);
static void compare_solutions (void);
static void confirm_exit    (void);

/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
 int in_the_exercise = 1;


/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /*************************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
```

```
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
     }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
     }
  forecolor = WHITE;
 }

/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}



/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                          */
/*************************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  1");
  /*************************************************************************/
  pieslice(35*x,5*y,0,359,2);
  pieslice(30*x,7*y,0,359,2);
  pieslice(40*x,7*y,0,359,2);
  pieslice(55*x,5*y,0,359,2);
  pieslice(50*x,7*y,0,359,2);
  pieslice(60*x,7*y,0,359,2);
  moveto(35*x,5*y); lineto(30*x,7*y);
  moveto(35*x,5*y); lineto(40*x,7*y);
  moveto(55*x,5*y); lineto(50*x,7*y);
  moveto(55*x,5*y); lineto(60*x,7*y);
  /*************************************************************************/
  outtextxy(29*x,2*y,"Is this graph a rooted tree ?");
  outtextxy(15*x,18*y,"Enter your choice here --->");
  Ch = getch ();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))) {
     outtextxy(48*x,18*y,"   Please type y or n");
     Ch = getch ();
     if(Ch==ESC) confirm_graph_exit();
     if((Ch == 'y') || (Ch == 'n') || (Ch == 'Y') || (Ch == 'N'))
     setcolor(backcolor);
     bar(50*x,17*y,88*x,19*y);
```

```
    setcolor(forecolor);
  }
  switch (Ch)        {
   case 'y': outtextxy(47*x,18*y,"y");
        outtextxy(52*x,18*y,"No. Because, it does not satis-");
        outtextxy(52*x,19*y,"fy any one of the properties of ");
        outtextxy(52*x,20*y,"a rooted tree. First it is not a");
        outtextxy(52*x,21*y,"tree (it is not connected) and");
        outtextxy(52*x,22*y,"and it has more than one vertex");
        outtextxy(52*x,23*y,"with indegree of 0.");
        break;
   case 'Y': outtextxy(47*x,18*y,"Y");
        outtextxy(52*x,18*y,"No. Because, it does not satis-");
        outtextxy(52*x,19*y,"fy any one of the properties of ");
        outtextxy(52*x,20*y,"a rooted tree. First it is not a");
        outtextxy(52*x,21*y,"tree (it is not connected) and");
        outtextxy(52*x,22*y,"and it has more than one vertex");
        outtextxy(52*x,23*y,"with indegree of 0.");
        break;

   case 'n': outtextxy(47*x,18*y,"n");
        outtextxy(52*x,18*y,"You are right! Conratulations.");
        break;

   case 'N': outtextxy(47*x,18*y,"N");
        outtextxy(52*x,18*y,"You are right! Conratulations.");
        break;

   default  : break;
   }
   Pause(30*x,24*y);
   closegraph();
}
```

```
/* PROGRAM  : q422.c
   AUTHOR   . Atilla BAKAN
   DATE     : Mar. 20, 1990
   REVISED  : Apr. 17, 1990

   DESCRIPTION : This program contains the second exercise about rooted
                 trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions       */
static void init_graph   (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer          (void);
static void example       (void);
static void show_alg      (void);
static void step_solution (void);
static void compare_solutions (void);
static void confirm_exit  (void);

/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
 int in_the_exercise = 1;


/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /***********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
```

```
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}



/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                          */
/*************************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE 2");
  /*************************************************************************/
  pieslice(40*x,3*y,0,359,2);
  pieslice(28*x,5*y,0,359,2);
  pieslice(40*x,5*y,0,359,2);
  pieslice(52*x,5*y,0,359,2);
  pieslice(20*x,7*y,0,359,2);
  pieslice(36*x,7*y,0,359,2);
  pieslice(44*x,7*y,0,359,2);
  pieslice(60*x,7*y,0,359,2);
  pieslice(62*x,9*y,0,359,2);
  moveto(20*x,7*y);  lineto(28*x,5*y); lineto(40*x,3*y);
  lineto(52*x,5*y);  lineto(60*x,7*y); lineto(62*x,9*y);
  moveto(28*x,5*y);  lineto(36*x,7*y);
  moveto(40*x,3*y);  lineto(40*x,5*y);
  moveto(52*x,5*y);  lineto(44*x,7*y);
  outtextxy(79*x/2,5*y/2,"C");
  outtextxy(26*x,5*y,"G");
  outtextxy(79*x/2,11*y/2,"H");
  outtextxy(54*x,5*y,"A");
  outtextxy(61*x,7*y,"B");
  outtextxy(20*x,15*y/2,"D");
  outtextxy(36*x,15*y/2,"E");
  outtextxy(44*x,15*y/2,"F");
```

```
outtextxy(62*x,19*y/2,"I");
/*************************************************************************/
outtextxy(18*x,11*y,"Which one of the following statements is  false ?");
outtextxy(20*x,13*y,"a)  F is left child of A.");
outtextxy(20*x,14*y,"b)  I is descendent of C and H and A.");
outtextxy(20*x,15*y,"c)  G is root of C's left subtree.");
outtextxy(20*x,16*y,"d)  H is a terminal vertice.");
outtextxy(18*x,18*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(48*x,18*y,"   Please type a,b,c, or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
   setcolor(backcolor);
   bar(50*x,17*y,88*x,19*y);
   setcolor(forecolor);
 }
 switch (Ch)        {
  case 'a': outtextxy(50*x,18*y,"a");
        outtextxy(55*x,18*y,"No. That's true, so you are");
        outtextxy(55*x,19*y,"wrong. The answer is 'b', ");
        outtextxy(55*x,20*y,"because I is descendent of");
        outtextxy(55*x,21*y,"C and A but not H.");
       break;
  case 'b': outtextxy(50*x,18*y,"b");
       outtextxy(55*x,18*y,"Correct. You found the false one.");
       break;
  case 'c': outtextxy(50*x,18*y,"c");
        outtextxy(55*x,18*y,"No. That's true, so you are");
        outtextxy(55*x,19*y,"wrong. The answer is 'b', ");
        outtextxy(55*x,20*y,"because I is descendent of");
        outtextxy(55*x,21*y,"C and A but not H.");
       break;
  case 'd': outtextxy(50*x,18*y,"d");
```

```
        outtextxy(55*x,18*y,"No. That's true, so you are");
        outtextxy(55*x,19*y,"wrong. The answer is 'b', ");
        outtextxy(55*x,20*y,"because I is descendent of");
        outtextxy(55*x,21*y,"C and A but not H.");
        break;
    default : break;
    }
    Pause(30*x,24*y);
    closegraph();
}
```

```
/* PROGRAM  : q423.c
   AUTHOR       : Atilla BAKAN
   DATE         : Mar. 20, 1990
   REVISED      : Apr. 17, 1990

   DESCRIPTION : This program contains the third exercise about properties
                 of trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                    /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)        /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk          find_t
   #define ff_name         name
#elif defined(__ZTC__)                    /* Zortech C/C++ */
   #define ffblk          FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions        */
static void init_graph   (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer          (void);
static void example       (void);
static void show_alg      (void);
static void step_solution (void);
static void compare_solutions (void);
static void confirm_exit  (void);

/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
 int in_the_exercise = 1;


/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /***********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
```

```
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
      }
   else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
      }
  forecolor = WHITE;
 }

/*************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
      case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
      case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
      default : break;
      }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}



/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*************************** ***********************************************/
/* Equivalent of press_a_key function for graphics screen                    */
/*****************************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/************************************ ***********************************/
/* main routine that calls exer routine                                      */
/*****************************************************************************/
void main()
{
  exer();
}
```

```
/*****************************************************************/
/* Routine that asks the question, then depending on the user's answer */
/* makes necessary explanations                                    */
/*****************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE 3");
  /*****************************************************************/
  pieslice(8*x,7*y,0,359,2);
  pieslice(13*x,5*y,0,359,2);
  pieslice(18*x,7*y,0,359,2);              /*  a  */
  pieslice(13*x,9*y,0,359,2);
  moveto(13*x,5*y); lineto(13*x,9*y);
  moveto(8*x,7*y); lineto(18*x,7*y);
  /*****************************************************************/
  pieslice(25*x,5*y,0,359,2);
  pieslice(30*x,5*y,0,359,2);
  pieslice(35*x,5*y,0,359,2);
  pieslice(40*x,5*y,0,359,2);
  pieslice(25*x,7*y,0,359,2);              /*  b  */
  pieslice(35*x,7*y,0,359,2);
  pieslice(40*x,7*y,0,359,2);
  pieslice(30*x,9*y,0,359,2);
  pieslice(35*x,9*y,0,359,2);
  pieslice(40*x,9*y,0,359,2);
  moveto(25*x,5*y); lineto(25*x,7*y); lineto(40*x,7*y); lineto(40*x,5*y);
  moveto(30*x,5*y); lineto(35*x,5*y); lineto(35*x,9*y); lineto(40*x,9*y);
  moveto(30*x,9*y); lineto(35*x,9*y);
  /*****************************************************************/
  pieslice(45*x,5*y,0,359,2);
```

```
pieslice(55*x,5*y,0,359,2);
pieslice(45*x,9*y,0,359,2);              /*  c  */
pieslice(55*x,9*y,0,359,2);
moveto(45*x,5*y); lineto(55*x,5*y); lineto(55*x,9*y);
lineto(45*x,9*y); lineto(45*x,5*y);
/*************************************************************************/
pieslice(65*x,5*y,0,359,2);
pieslice(75*x,5*y,0,359,2);
pieslice(85*x,5*y,0,359,2);
pieslice(65*x,9*y,0,359,2);              /*  d  */
pieslice(75*x,9*y,0,359,2);
pieslice(85*x,9*y,0,359,2);
moveto(65*x,5*y); lineto(85*x,5*y);
moveto(65*x,9*y); lineto(85*x,9*y);
moveto(75*x,5*y); lineto(75*x,9*y);
/*************************************************************************/
outtextxy(13*x,12*y,"(a)");
outtextxy(32*x,12*y,"(b)");
outtextxy(49*x,12*y,"(c)");
outtextxy(74*x,12*y,"(d)");
/*************************************************************************/
outtextxy(15*x,2*y,"Which one of the following graphs is a rooted tree ?");
outtextxy(15*x,18*y,"Enter your choice here --->");
Ch = getch ();
if(Ch==ESC) confirm_graph_exit();
while (!((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))) {
   outtextxy(45*x,18*y,"   Please type a,b,c, or d");
   Ch = getch ();
   if(Ch==ESC) confirm_graph_exit();
   if((Ch == 'a') || (Ch == 'b') || (Ch == 'c') || (Ch == 'd'))
   setcolor(backcolor);
   bar(45*x,17*y,88*x,19*y);
   setcolor(forecolor);
 }
 switch (Ch)        {
  case 'a': outtextxy(47*x,18*y,"a");
```

```
            outtextxy(52*x,18*y,"Sorry! It cannot be a rooted");
            outtextxy(52*x,19*y,"tree, because it has more than");
            outtextxy(52*x,20*y,"on vertex with indegree of 0,");
            outtextxy(52*x,21*y,"moreover, since it is not con-");
            outtextxy(52*x,22*y,"nected, it is not even a tree.");
            outtextxy(52*x,23*y,"The answer is 'd'.");
            break;
    case 'b': outtextxy(47*x,18*y,"b");
            outtextxy(52*x,18*y,"Sorry! It cannot be a rooted");
            outtextxy(52*x,19*y,"tree, because it has more than");
            outtextxy(52*x,20*y,"on vertex with indegree of 0.");
            outtextxy(52*x,21*y,"The answer is 'd'.");
            break;
    case 'c': outtextxy(47*x,18*y,"c");
            outtextxy(52*x,18*y,"Sorry! It cannot be a rooted");
            outtextxy(52*x,19*y,"tree, because it has a cycle.");
            outtextxy(52*x,20*y,"The answer is 'd'.");
            break;
    case 'd': outtextxy(47*x,18*y,"d");
            outtextxy(52*x,18*y,"That's right! Congratulations.");
            break;
    default  : break;
    }
    Pause(30*x,24*y);
    closegraph();
}
```

```
/* PROGRAM  : q424.c
   AUTHOR   : Atilla BAKAN
   DATE     : Mar. 20, 1990
   REVISED  : Apr. 17, 1990

   DESCRIPTION : This program contains the fourth exercise about rooted
                 trees.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/


/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer            (void);
static void example         (void);
static void show_alg        (void);
static void step_solution   (void);
static void compare_solutions (void);
static void confirm_exit    (void);

/*************************************************************************/
/* miscellaneous global variables                                        */
/*************************************************************************/
 int in_the_exercise = 1;


/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;
```

```
/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}

/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  /***********************************************************************/
  settext();
  /***********************************************************************/
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
```

```
    ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
      setfillstyle(SOLID_FILL,BLACK);
      backcolor = BLACK;
      quitcolor = WHITE;
     }
  else {
      setfillstyle(SOLID_FILL,BLUE);
      backcolor = BLUE;
      quitcolor = RED;
     }
 forecolor = WHITE;
}

/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
   }
   switch (ch)        {
    case 'y': closegraph();
```

```
            videoinit();
            exit(0);
            break;
    case 'Y': closegraph();
            videoinit();
            exit(0);
            break;
    case 'n': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    case 'N': setcolor(backcolor);
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
    default : break;
    }
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);     /* restore any hidden hot keys */
}



/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
```

```
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
   settext();
   outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
   if(waitkey()==ESC) confirm_graph_exit();
  }


/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* Routine that asks the question, then depending on the user's answer   */
/* makes necessary explanations                                          */
/*************************************************************************/
static void exer(void)
{
  char Ch;

  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXERCISE  4");
  /*************************************************************************/
  outtextxy(15*x,8*y,"In a rooted tree the level of a vertex is defined");
  outtextxy(15*x,9*y,"to be the length of the simple directed path from");
  outtextxy(15*x,10*y,"the root to that vertex. What is the level of the");
  outtextxy(15*x,11*y,"root ?");
  outtextxy(15*x,13*y," a)  1 ");
  outtextxy(15*x,14*y," b)  0 ");
  /*************************************************************************/
  outtextxy(18*x,18*y,"Enter your choice here --->");
  Ch = getch ();
  if(Ch==ESC) confirm_graph_exit();
  while (!((Ch == 'a') || (Ch == 'b'))) {
     outtextxy(48*x,18*y,"    Please type a or b");
     Ch = getch ();
     if(Ch==ESC) confirm_graph_exit();
     if((Ch == 'a') || (Ch == 'b'))
     setcolor(backcolor);
     bar(50*x,17*y,88*x,19*y);
     setcolor(forecolor);
   }
   switch (Ch)        {
    case 'a': outtextxy(50*x,18*y,"a");
         outtcxtxy(55*x,18*y,"No that's not true. Consider");
```

```
        outtextxy(55*x,19*y,"the following example...");
        Pause(30*x,24*y);
        setcolor(backcolor);
        bar(0,0,MaxX,MaxY);
        setcolor(forecolor);
        rectangle(x,y,MaxX-x,MaxY-y/2);
        example();
        break;
    case 'b': outtextxy(50*x,18*y,"b");
        outtextxy(55*x,18*y,"That's right!");
        break;
    default  : break;
    }
    Pause(30*x,24*y);
    closegraph();
}

/*************************************************************************/
/* This routine gives an example about the level of root in a rooted tree */
/*************************************************************************/
static void example(void)
{
  /***********************************************************************/
  pieslice(40*x,3*y,0,359,2);
  pieslice(28*x,5*y,0,359,2);
  pieslice(40*x,5*y,0,359,2);
  pieslice(52*x,5*y,0,359,2);
  pieslice(20*x,7*y,0,359,2);
  pieslice(36*x,7*y,0,359,2);
  pieslice(44*x,7*y,0,359,2);
  pieslice(60*x,7*y,0,359,2);
  pieslice(62*x,9*y,0,359,2);
  moveto(20*x,7*y); lineto(28*x,5*y); lineto(40*x,3*y);
  lineto(52*x,5*y); lineto(60*x,7*y); lineto(62*x,9*y);
  moveto(28*x,5*y); lineto(36*x,7*y);
  moveto(40*x,3*y); lineto(40*x,5*y);
```

```
  moveto(52*x,5*y); lineto(44*x,7*y);
  outtextxy(79*x/2,5*y/2,"C");
  outtextxy(26*x,5*y,"G");
  outtextxy(79*x/2,11*y/2,"H");
  outtextxy(54*x,5*y,"A");
  outtextxy(61*x,7*y,"B");
  outtextxy(20*x,15*y/2,"D");
  outtextxy(36*x,15*y/2,"E");
  outtextxy(44*x,15*y/2,"F");
  outtextxy(62*x,19*y/2,"I");
  /*************************************************************************/
  outtextxy(5*x,13*y,"As we said earlier we measure the level of a vertex with its
                      simple");
  outtextxy(5*x,14*y,"directed path from the root. And length of a path is determined
                      by");
  outtextxy(5*x,15*y,"the number of directed edges in the path. For example the
                      level of");
  outtextxy(5*x,16*y,"vertex F in the example graph is two, because its simple
                      directed");
  outtextxy(5*x,17*y,"path consists of two directed edges, namely (C, A) and
                      (A, F). So,");
  outtextxy(5*x,18*y,"if the level of vertex F is 2, then with similar approach, the
                      level");
  outtextxy(5*x,19*y,"of A is 1, and obviously, the level of root C is 0.");
}
```

```
/* PROGRAM  : span.c
   AUTHOR   : Atilla BAKAN
   DATE     : Mar. 18, 1990
   REVISED  : Apr. 18, 1990

   DESCRIPTION : This program contains the tutorial for spanning trees
                 It also covers the breadth-first search algorithm which
                 is used for finding spanning trees in graphs.

   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <process.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"
#include "cxlstr.h"
#include "cxlvid.h"
#include "cxlwin.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
```

```
   #define ff_name      name
   #define ff_attrib    attribute
#endif

#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions       */
static void add_shadow   (void);
static void confirm_quit  (void);
static void disp_sure_msg (void);
static void error_exit   (int errnum);
static void initialize   (void);
static void move_window   (int nsrow, int scol);
static void normal_exit   (void);
static void Pageup       (void);
static void Pagedown     (void);
static void press_a_key   (int wrow);
static void pre_help     (void);
static void quit_window   (void);
static void restore_cursor(void);
static void short_delay   (void);
static void size_window   (int nerow,int necol);

/* Tutorial procedures          */
static void spanning_trees   (void);
static void definition_4_2_3 (void);
static void ex_span_1        (void);
static void breadth_first    (void);
static void breadth_first_tree (void);
static void ex_span_2        (void);
static void ex_span_3        (void);
static void ex_span_4        (void);
static void ex_span_5        (void);
static void theorem_4_7      (void);
```

```
static void proof_4_7      (void);
static void exercises      (void);
static void exer1          (void);
static void exer2          (void);
static void P1             (void);
static void P2             (void);
static void P3             (void);
static void P4             (void);
static void P5             (void);
static void P6             (void);
static void P7             (void);
static void P8             (void);
static void P9             (void);
static void P10            (void);
static void P11            (void);
static void P12            (void);

/*****************************************************************************/
/* miscellaneous global variables                                            */
/*****************************************************************************/
static int *savescrn,crow,ccol;
static WINDOW w[10];


/*****************************************************************************/
/* error message table                                                       */
/*****************************************************************************/
static char *error_text[]= {
   NULL,  /* ermum =  0, no error    */
   NULL,  /* ermum == 1, windowing error */
   "Syntax:  CXLDEMO [-switches]\n\n"
      "\t -c = CGA snow elimination\n"
      "\t -b = BIOS screen writing\n"
      "\t -m = force monochrome text attributes",
   "Memory allocation error"
};
```

```
/*************************************************************************/
/* miscellaneous defines                                                 */
/*************************************************************************/
#define SHORT_DELAY 18
#define H_WINTITLE  33

/*************************************************************************/
/* this function will add a shadow to the active window                  */
/*************************************************************************/
static void add_shadow(void)
{
  wshadow(LGREY|_BLACK);
}


/*************************************************************************/
/* this function pops open a window and confirms that the user really    */
/* wants to quit the demo.  If so, it terminates the demo program.       */
/*************************************************************************/
static void confirm_quit(void)
{
  struct _onkey_t *kblist;

  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,26,13,55,0,WHITE|_BROWN,WHITE|_BROWN)) error_exit(1);
  add_shadow();
  wputs("\n Quit demo, are you sure? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y') normal_exit();
  wclose();
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
}
```

```
/*************************************************************************/
/* this function is called by the pull-down demo for a prompt            */
/*************************************************************************/
static void disp_sure_msg(void)
{
  wprints(0,2,WHITE|_BLUE,"Are you sure?");
}

/*************************************************************************/
/* this function handles abnormal termination.  If it is passed an       */
/* error code of 1, then it is a windowing system error.  Otherwise       */
/* the error message is looked up in the error message table.             */
/*************************************************************************/
static void error_exit(int ermum)
{
  if(ermum) {
    printf("\n%s\n",(ermum==1)?wermmsg():error_text[ermum]);
          exit(ermum);
  }
}

/*************************************************************************/
/* this function initializes CXL's video, mouse, keyboard, and help systems */
/*************************************************************************/
static void initialize(void)
{
  /* initialize the CXL video system and save current screen info */
  videoinit();
  readcur(&crow,&ccol);
  if((savescrn=ssave())==NULL) error_exit(3);

  /* if mouse exists, turn on full mouse support */
  if(msinit()) {
    mssupport(MS_FULL);
    msgotoxy(12,49);
  }
```

```
   /* attach [Alt-X] to the confirm_quit() function */
   setonkey(0x2d00,confirm_quit,0);

   /* attach [Ctrl Pageup] to the Pageup() function */
   setonkey(0x8400,Pageup,0);

   /* attach [Ctrl Pagedown] to the Pagedown() function */
   setonkey(0x7600,Pagedown,0);

   /* initialize help system, help key = [F1] */
                                                                    whelp-
def("CXLDEMO.HLP",0x3b00,YELLOW|_RED,LRED|_RED,WHITE|_RED,RED|_
LGREY
      ,pre_help);
}
/*************************************************************************/
/* this function is called anytime to switch back to previous window.    */
/*************************************************************************/
static void Pageup(void)
{
   static WINDOW handle;

   handle = whandle();
   wactiv(handle - 1);
}

/*************************************************************************/
/* this function is called anytime to switch back to next window.        */
/*************************************************************************/
static void Pagedown(void)
{
   static WINDOW handle;

   handle = whandle();
   wactiv(handle + 1);
}
```

```
static void pre_help(void)
{
   add_shadow();
   setonkey(0x2d00,confirm_quit,0);
}

/*************************************************************************/
/* this function handles normal termination. The original screen and cursor */
/* coordinates are restored before exiting to DOS with ERRORLEVEL 0.      */
/*************************************************************************/
static void normal_exit(void)
{
   srestore(savescm);
   gotoxy_(crow,ccol);
   if(_mouse) mshidecur();
   showcur();
   exit(0);
}

/*************************************************************************/
/* this function displays a pause message then pauses for a keypress     */
/*************************************************************************/
static void press_a_key(int wrow)
{
   register int attr1;
   register int attr2;

   attr1=(YELLOW)|((_winfo.active->wattr>>4)<<4);
   attr2=(LGREY)|((_winfo.active->wattr>>4)<<4);
   wcenters(wrow,attr1,"Press a key");
   wprints(wrow,0,LGREY|_RED,"Pgup/Pgdn");
   hidecur();
   if(waitkey()==ESC) confirm_quit();
   wcenters(wrow,attr1,"          ");
   wprints(wrow,0,attr2,"        ");
}
```

```
/*************************************************************************/
/* This routine causes short dealys during execution                     */
/*************************************************************************/
static void short_delay(void)
{
  delay_(SHORT_DELAY);
}

/*************************************************************************/
/* this function is called by the pull-down menu demo anytime            */
/* the  selection bar moves on or off the [Q]uit menu items.             */
/*************************************************************************/
static void quit_window(void)
{
  static WINDOW handle=0;

  if(handle) {
    wactiv(handle);
    wclose();
    handle=0;
  }
  else {
    handle=wopen(14,41,17,70,0,YELLOW|_RED,WHITE|_RED);
    wputs(" Quit takes you back to the\n demo program's main menu.");
  }
}


/*************************************************************************/
/* shows the cursor again if it has been hidden                          */
/*************************************************************************/
static void restore_cursor(void)
{
  wtextattr(WHITE|_MAGENTA);
  showcur();
}
```

```
/*************************************************************************/
/* enlarges or shrinks the windows                                       */
/*************************************************************************/
static void size_window(int nerow,int necol)
{
  wsize(nerow,necol);
  short_delay();
}
/*************************************************************************/
/* moves the active window to a given screen coordinates                 */
/*************************************************************************/
static void move_window(int nsrow,int nscol)
{
  if(wmove(nsrow,nscol)) error_exit(1);
  short_delay();
}

/*************************************************************************/
/* this routine  calling spanning_trees() routine whenever Pageup or Pagedown  */
/* keys are pressed.                                                     */
/*************************************************************************/
void P1()
{
  wcloseall();
  spanning_trees();
}

/*************************************************************************/
/* this routine calling ex_span_1() routine whenever Pageup or           */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P2()
{
  wcloseall();
  ex_span_1();
}
```

```
/*************************************************************************/
/* this routine  calling definition_4_2_3 routine whenever Pageup or     */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P3()
{
  wcloseall();
  definition_4_2_3();
}
/*************************************************************************/
/* this routine calling ex_span_2() routine whenever Pageup or           */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P4()
{
  wcloseall();
  ex_span_2();
}
/*************************************************************************/
/* this routine calling breadth_first() routine whenever Pageup or       */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P5()
{
  wcloseall();
  breadth_first();
}
/*************************************************************************/
/* this routine calling breadth_first_tree routine whenever Pageup or    */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P6()
{
  wcloseall();
  breadth_first_tree();
}
```

```
/*************************************************************************/
/* this routine calling ex_span_5  routine whenever Pageup or            */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P7()
{
   wcloseall();
   ex_span_4();
}
/*************************************************************************/
/* this routine calling ex_span_6 routine whenever Pageup or             */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P8()
{
   wcloseall();
   ex_span_5();
}
/*************************************************************************/
/* this routine calling theorem_4_7 routine whenever Pageup or           */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P9()
{
   wcloseall();
   theorem_4_7();
}
/*************************************************************************/
/* this routine calling exercises routine whenever Pageup or             */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P10()
{
   wcloseall();
   exercises();
}
```

```
/*************************************************************************/
/* this routine calling exer1 routine whenever Pageup or                 */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P11()
{
   wcloseall();
   exer1();
}



/*************************************************************************/
/* this routine calling exer2 routine whenever Pageup or                 */
/* Pagedown keys are pressed.                                            */
/*************************************************************************/
void P12()
{
   wcloseall();
   exer2();
}



/*************************************************************************/
/* main routine which calls  spanning trees tutorial                     */
/*************************************************************************/
void main()
{
  initialize();
  spanning_trees();
}
```

```
/*************************************************************************/
/* Routine which calls definition, example and algorithm routines about  */
/* spanning trees.                                                       */
/*************************************************************************/
static void spanning_trees(void)
{
  register int *scrn;

  if((scrn=ssave())==NULL) error_exit(3);
  cclrscrn(LGREY|_BLUE);
  /*************************************************************************/
  /* attach [Pagedown] to the ex_span_1() function */
  setonkey(0x5100,P2,0);
  /*************************************************************************/
  if((w[1]=wopen(5,15,17,65,3,LCYAN|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
  wtitle("[Spanning Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" Suppose an oil company wants to build a series of pipelines"
         " between six storage facilities in order to be able to move"
         " oil from one storage facility to any of the other five."
         " Because the construction of pipeline is very expensive, the"
         " company wants to construct as few pipelines as possible."
         " For environmental reasons it is not possible to build a"
         " pipeline between each pair of storage facilities.");
  press_a_key(10);
  short_delay();
  wslide(1,15);
  short_delay();
  ex_span_1();
  srestore(scrn);
}
```

```
/*************************************************************************/
/* This routine gives an example about a spanning tree                   */
/*************************************************************************/
static void ex_span_1 (void)
{
  /***********************************************************************/
  /* attach [Pageup] to the spanning_trees() function */
  setonkey(0x4900,P1,0);
  /***********************************************************************/
  /* attach [Pagedown] to the definition_4_2_3() function */
  setonkey(0x5100,P3,0);
  /***********************************************************************/
  if((w[2]=wopen(14,15,18,65,3,LCYAN|_BLACK,BLACK|_LGREY))==0)
          error_exit(1);
  wtitle("[Spanning Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" The following graph shows the pipelines that can be built.");
  press_a_key(2);
  wcloseall();
  spawnl(P_WAIT,"exspan1.exe",NULL);
  cclrscm(LGREY|_BLUE);
  definition_4_2_3();
}
```

```
/*****************************************************************/
/* This routine gives the definition of a spanning tree.         */
/*****************************************************************/
static void definition_4_2_3(void)
{
  /*****************************************************************/
  /* attach [Pageup] to the ex_span_1() function */
  setonkey(0x4900,P2,0);
  /*****************************************************************/
  /* attach [Pagedown] to the ex_span_2() function */
  setonkey(0x5100,P4,0);
  /*****************************************************************/
  if((w[1]=wopen(5,15,9,65,3,LCYAN|_BLACK,BLACK|_LGREY))==0)
          error_exit(1);
  wtitle("[Spanning Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" The later graph in the example takes us to the idea of"
      " spanning tree. What is a spanning tree ?");
  press_a_key(2);
  wslide(1,15);
  short_delay();
  if((w[2]=wopen(6,15,11,65,3,LCYAN|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
  wtitle("[Definition - Spanning Trees]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" A spanning tree of a  graph G is a tree (formed by using edges"
      " and vertices of G) containing all the vertices of G.");
  press_a_key(3);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"exspan2.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  ex_span_2();
}
```

```
/*************************************************************************/
/* This routine gives an example illustration of obtaining a spanning tree */
/* from a given graph                                                      */
/*************************************************************************/
static void ex_span_2 (void)
{
  /*************************************************************************/
  /* attach [Pageup] to the definition_4_2_3() function */
  setonkey(0x4900,P3,0);
  /*************************************************************************/
  /* attach [Pagedown] to the breadth_first() function */
  setonkey(0x5100,P5,0);
  /*************************************************************************/
  if((w[1]=wopen(5,15,13,65,3,LCYAN|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
  wtitle("[Spanning Trees - Example_3]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" There are several ways to find a spanning tree for a graph"
         " One way is to  remove an edge from each cycle. This method"
         " is used to find the fundamental system of circuits in an"
         " electrical network. This  process is illustrated in the "
         " following example.");
  press_a_key(6);
  short_delay();
  wclose();
  spawnl(P_WAIT,"exspan3.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  breadth_first();
}
```

```
/*********** ***********************************************************/
/* This routine explains and illustrates the implementation of the breadth          */
/* first search algorithm.                                                          */
/*********** ***********************************************************/
static void breadth_first(void)
{

  /*********** ***********************************************************/
  /* attach [Pageup] to the ex_span_2() function */
  setonkey(0x4900,P4,0);
  /*********** ***********************************************************/
  /* attach [Pagedo  n] to the breadth_first_tree() function */
  setonkey(0x5100,P6,0);
  /*********** ***********************************************************/
  if((w[1]=wopen(5,15,16,54,3,LCYAN|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
  wtitle("[Breadth First Search]",TCENTER,_LGREY|BROWN);
  add_shadc v();
  whelpcat(H_WINTITLE);
  wputsw(" The method we described above is not the only way to find"
          " a spanning tree. There are many others, and some of these"
          " are easier to program on a computer because they do not"
          " require that cycles be found. One of these methods is"
          " based on the idea of breadth-first search, which was"
          " discussed in earlier sections.");
  press_a_key(9);
  wslide(0,0);
  /*********** ***********************************************************/
        if((w[2]=wopen(2,15,21,54,3,LCYAN|_BLACK,BLACK|_GREEN))==0)  er-
ror_exit(1);
  wtitle("[Breadth First Search]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" Recall that in breadth-first search algorithm we start"
          " with a vertex S. Then we find the vertices adjacent to S and"
          " assign them the label 1. Next we look at the unlabeled"
```

```
            " vertices that are adjacent to those with label 1. These"
            " newly found vertices are then given the label 2. We continue"
            " by finding unlabeled vertices adjacent to the vertices with"
            " label 3. This process is repeated until there are no more"
            " unlabeled vertices adjacent to labeled vertices. In this"
            " procedure there are edges that lead from a vertex with label"
            " k to vertex with label k + 1 that form a spanning tree for"
            " the original graph.");
press_a_key(17);
wslide(0,40);
/*************************************************************************/
if((w[3]=wopen(5,15,10,65,3,LCYAN|_BLACK,BLACK|_RED))==0)
        error_exit(1);
wtitle("[Depth First Search]",TCENTER,_LGREY|BROWN);
add_shadow();
whelpcat(H_WINTITLE);
wputs("\n");
wputsw(" The formal specification of the breadth_first search algorithm"
        " is as follows :");
press_a_key(3);
short_delay();
wcloseall();
/*************************************************************************/
if((w[1]=wopen(0,15,24,65,3,BLACK|_GREEN,BLACK|_CYAN))==0)
        error_exit(1);
wtitle("[Breadth First Search Algorithm]",TCENTER,BLUE|_LGREY);
add_shadow();
wputsw(" This algorithm will find a spanning tree, if it exists,"
        " for a graph G with n vertices. In the algorithm, L is the set"
        " of vertices with labels and T is the set of edges connecting"
        " the vertices in L.");
wputs("\n");
wputsw(" Step 1 (start with a vertex). Pick a vertex U and assign"
        " U the label 0. Let  L = { x }, T = 0, and k = 0");
wputs("\n");
wputsw(" Step 2 (L has n vertices). If L contains all the vertices"
```

```
            " of G, then stop; the edges in T and the vertices in L form"
            " a spanning tree for G.");
    wputs("\n");
    wputs("   Step 3 (L has fewer than n vertices). If L does not contain"
          " all the vertices of G, find the vertices not in L that are"
            " adjacent to the vertices in L with largest label number k."
            " If there are no such vertices, G has no spanning tree. "
            " Otherwise, assign these newly found vertices the label k + 1"
            " and put them in L. For each new vertex with label k + 1,"
            " place in T one edge connecting this vertex to a vertex with"
            " label k. If there is more than one such edge, choose one"
            " arbitrarily. Return to Step 2.");
    press_a_key(22);
    short_delay();
    /*************************************************************************/
    if((w[2]=wopen(5,15,10,65,3,LCYAN|_BLACK,BLACK|_RED))==0)
            error_exit(1);
    wtitle("[Breadth First Search - Example_4]",TCENTER,_LGREY|BROWN);
    add_shadow();
    whelpcat(H_WINTITLE);
    wputs("\n      We need to show an example !");
    press_a_key(3);
    short_delay();
    wcloseall();
    spawnl(P_WAIT,"exspan4.exe",NULL);
    cclrscrn(LGREY|_BLUE);
    breadth_first_tree();
}
```

```
/*************************************************************************/
/* Another example about a Breadth First Search Algorithm implementation */
/*************************************************************************/
static void breadth_first_tree(void)
{

  /*************************************************************************/
  /* attach [Pageup] to the breadth_first() function */
  setonkey(0x4900,P5,0);
  /*************************************************************************/
  /* attach [Pagedown] to the ex_span_4() function */
  setonkey(0x5100,P7,0);
  /*************************************************************************/
  if((w[1]=wopen(2,15,14,65,3,LCYAN|_BLACK,BLACK|_GREEN))==0)
          error_exit(1);
  wtitle("[Breadth First Search]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" The construction process in Step 3 of this algorithm"
          " guarantees that the edges in T and the vertices in L form"
          " a connected graph. Furthermore, each edge in T joins"
          " two vertices labeled with consecutive integers, and no"
          " vertex in L is connected by an edge in T to more than one"
         " vertex with smaller label.Therefore, no collection of edges"
          " in T forms a cycle. Thus, after each iteration of Step 3"
         " the edges in T and the vertices in L form a tree.");
  press_a_key(10);
  short_delay();
  ex_span_4();
}
```

```
/*************************************************************************/
/* Another example about a Breadth First Search Algorithm implementation */
/*************************************************************************/
static void ex_span_4 (void)
{
  /*************************************************************************/
  /* attach [Pageup] to the breadth_first_tree() function */
  setonkey(0x4900,P6,0);
  /*************************************************************************/
  /* attach [Pagedown] to the ex_span_5() function */
  setonkey(0x5100,P8,0);
  /*************************************************************************/
  if((w[2]=wopen(13,15,18,65,3,LCYAN|_BLACK,BLACK|_RED))==0)
          error_exit(1);
  wtitle("[Breadth First Search]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n     Now we will show you one more example.");
  wputs("\n       But a little bit complicated !");
  press_a_key(3);
  short_delay();
  wcloseall();
  spawnl(P_WAIT,"exspan5.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  /*************************************************************************/
  if((w[1]=wopen(5,15,12,65,3,LCYAN|_BLACK,BLACK|_CYAN))==0)
          error_exit(1);
  wtitle("[Breadth First Search]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" A spanning tree constructed by means of the breadth-"
         " first algorithm is sometimes called a 'shortest path tree'"
         " because of the path from U to any other vertex using"
         " edges in the spanning tree is a shortest path in the"
        " original graph.");
  press_a_key(5);
```

```
   short_delay();
   wclose();
   ex_span_5();
}



/*************************************************************************/
/* This routine gives an example illustration of a graph which does not have  */
/* a spanning tree                                                            */
/*************************************************************************/
static void ex_span_5 (void)
{
   /*************************************************************************/
   /* attach [Pageup] to the ex_span_5() function */
   setonkey(0x4900,P7,0);
   /*************************************************************************/
   /* attach [Pagedown] to the theorem_4_7() function */
   setonkey(0x5100,P9,0);
   /*************************************************************************/
   if((w[1]=wopen(8,15,13,65,3,LCYAN|_BLACK,WHITE|_BLUE))==0)
            error_exit(1);
   wtitle("[Breadth First Search]",TCENTER,_LGREY|BRO..N);
   add_shadow();
   whelpcat(H_WINTITLE);
   wputsw(" In the examples so far, the graphs have had spanning trees."
          " However, this is not always the case, as you will see in "
          " the next example.");
   press_a_key(3);
   wclose();
   spawnl(P_WAIT,"exspan6.exe",NULL);
   cclrscm(LGREY|_BLUE);
   theorem_4_7();
}
```

```
/*************************************************************************/
/* This routine gives the theorem_4_7 about the spanning trees. Besides,  */
/* if the user wants, it gives the proof for this theorem.                */
/*************************************************************************/
static void theorem_4_7(void)
 {
  struct _onkey_t *kblist;
  /*************************************************************************/
  /* attach [Pageup] to the ex_span_6() function */
  setonkey(0x4900,P8,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exercises() function */
  setonkey(0x5100,P10,0);
  /*************************************************************************/
  if((w[1]=wopen(5,15,11,65,3,LCYAN|_BLACK,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Breadth First Search]",TCENTER,_LGREY|BROWN);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputsw(" In the examples we have seen that the existance of a"
         " spanning tree is related to the connectedness of the graph."
        " This relationship is made explicit in the following theorem.");
  press_a_key(4);
  short_delay();
  wclose();
  /*************************************************************************/
  if((w[1]=wopen(2,4,7,71,3,LCYAN|_GREEN,WHITE|_LGREY))==0)
          error_exit(1);
  wtitle("[Spanning Trees - Theorem_4_7]",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
  wputs("\n  Theorem_4_7\n");
  wputsw("   A graph G is connected if and only if G has a spanning tree.");
  press_a_key(3);
  /*************************************************************************/
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
```

```
  if(_mouse&MS_CURS) mshidecur();
  if(!wopen(9,20,13,55,0,BROWN|_CYAN,RED|_BLACK)) error_exit(1);
  add_shadow();
  wputs("\n Do you want to see the proof? \033A\156Y\b");
  clearkeys();
  showcur();
  if(wgetchf("YN",'Y')=='Y') {
    wclose();
    proof_4_7();
  }
  else wclose();
  hidecur();
  if(_mouse&MS_CURS) msshowcur();
  chgonkey(kblist);      /* restore any hidden hot keys */
  /*************************************************************************/
  wclose();
  exercises();
}


/*************************************************************************/
/* This routine gives the proof to the theorem_4_7                       */
/*************************************************************************/
static void proof_4_7(void)
{
  /*************************************************************************/
  /* attach [Pageup] to the ex_span_6() function */
  setonkey(0x4900,P8,0);
  /*************************************************************************/
  /* attach [Pagedown] to the exercises() function */
  setonkey(0x5100,P10,0);
  /*************************************************************************/
  if((w[2]=wopen(8,4,20,71,3,LCYAN|_RED,WHITE|_GREEN))==0) error_exit(1);
  wtitle("|Theorem_4_7 - Proof|",TCENTER,_MAGENTA|WHITE);
  add_shadow();
  whelpcat(H_WINTITLE);
```

```
    wputsw("  Since this is an if and only if statement we have make our"
           " proof in both ways. We will first prove the statement :");
    wputs("\n   A graph G is connected  =>  G has a spanning tree.");
    wputs("\n\nProof :");
    wputsw("  Suppose that the graph G has a spanning tree T. Since T is"
           " connected graph containing all the vertices in G, for any two"
           " vertices U and V in G there is a path between U and V using"
           " edges from T. But since the edges of T are also edges of G,"
           " we have a path between U and V using edges in G, Hence, G"
           " is connected.");
    press_a_key(10);
    short_delay();
    wslide(0,4);
    short_delay();
    /*************************************************************************/
    if((w[3]=wopen(12,4,24,71,3,LCYAN|_RED,WHITE|_GREEN))==0)
            error_exit(1);
    wtitle("[Theorem_4_7 - Proof]",TCENTER,_MAGENTA|WHITE);
    add_shadow();
    whelpcat(H_WINTITLE);
    wputs("\n");
    wputsw("  Now we will go the other way around, and give a proof to"
           " the statement :");
    wputs("\n   G has a spanning tree  =>  A graph G is connected.");
    wputs("\n\nProof :");
    wputsw("  Suppose G is connected. Applying the bradth-first search"
           " spanning tree algorithm to G yields a set L of vertices with"
           " labels and a set T of edges connecting the vertices in L."
           " Moreover, T is a tree. Since G is connected, each vertex of"
           " G is labeled. Thus, L contains all the vertices of G and"
           " T is a spanning tree for G. This completes our proof for"
           " the theorem 4.7.");
    press_a_key(10);
    wclose();
    wclose();
}
```

```
/*************************************************************************/
/* This routine makes a small quiz about the spanning trees.             */
/*************************************************************************/
void exercises(void)
 {
  register int *screen;
  /***********************************************************************/
  /* attach [Pageup] to the theorem_4_7() function */
  setonkey(0x4900,P9,0);
  /***********************************************************************/
  /* attach [Pagedown] to the exer1() function */
  setonkey(0x5100,P11,0);
  /***********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
            error_exit(1);
  wtitle("[Spanning Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw(" We have completed our presentation of this section. Are"
         " you ready for a pop quiz ? ");
  press_a_key(3);
  short_delay();
  wclose();
  if((screen=ssave())==NULL) error_exit(3); {
  exer1();
  /* if mouse exists, turn on full mouse support */
   if(msinit()) {
     mssupport(MS_FULL);
     msgotoxy(12,49);
     }
   /* attach [Alt-X] to the confirm_quit() function */
   setonkey(0x2d00,confirm_quit,0);
   srestore(screen);
  }
 }
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise qs_4_2_1                   */
/*************************************************************************/
static void exer1(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the theorem_4_7() function          */
  setonkey(0x4900,P9,0);
  /***********************************************************************/
  /* attach [Pagedown] to the exer2() function */
  setonkey(0x5100,P12,0);
  /***********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Spanning Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("      Here is the first question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"qs421.exe",NULL);
  cclrscm(LGREY|_BLUE);
  exer2();
}
```

```
/*************************************************************************/
/* Dummy function to call the actual exercise qs_4_2_2                   */
/*************************************************************************/
static void exer2(void)
{
  /***********************************************************************/
  /* attach [Pageup] to the exer1() function          */
  setonkey(0x4900,P11,0);
  /***********************************************************************/
  if((w[1]=wopen(5,15,10,65,3,LCYAN|_GREEN,WHITE|_RED))==0)
          error_exit(1);
  wtitle("[Spanning Trees]",TCENTER,_LGREY|BROWN);
  whelpcat(H_WINTITLE);
  add_shadow();
  wputs("\n");
  wputsw("      Here is the second question. ");
  press_a_key(3);
  wclose();
  spawnl(P_WAIT,"qs422.exe",NULL);
  cclrscrn(LGREY|_BLUE);
  normal_exit();
}
```

```
/* PROGRAM  : exspan1.c
   AUTHOR   : Atilla BAKAN
   DATE     : Apr. 18, 1990
   REVISED  : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for spanning trees.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions       */
static void init_graph   (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_di   ers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer          (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
       }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
       }
  forecolor = WHITE;
}
```

```
/*****************************************************************************/
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)         {
    case 'y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'Y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'n': setcolor(backcolor);
              bar(1*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    case 'N': setcolor(backcolor),
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* This routine illustrates an spanning tree.                            */
/*************************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE SPAN_1");
   /*********************************************************************/
   pieslice(3*x,14*y,0,359,2);
   pieslice(33*x,12*y,0,359,2);
   pieslice(43*x,18*y,0,359,2);
   pieslice(8*x,18*y,0,359,2);
   pieslice(23*x,51*y/4,0,359,2);
   pieslice(28*x,18*y,0,359,2);
   moveto(3*x,14*y);  lineto(23*x,51*y/4); lineto(33*x,12*y);
   lineto(43*x,18*y); lineto(8*x,18*y);    lineto(3*x,14*y);
   moveto(8*x,18*y);  lineto(23*x,51*y/4); lineto(28*x,18*y);
   outtextxy(11*x,14*y,"a");
   outtextxy(27*x,13*y,"b");
   outtextxy(36*x,16*y,"c");
   outtextxy(34*x,35*y/2,"d");
   outtextxy(15*x,35*y/2,"e");
   outtextxy(15*x,16*y,"f");
   outtextxy(7*x,16*y,"h");
   outtextxy(28*x,16*y,"g");
   /*********************************************************************/
   outtextxy(2*x,2*y,"The task is to find a set of edges which, together with vertices
                      incident on");
   outtextxy(2*x,3*y,"these edges, form a connected graph containing all the vertices
                      and having no");
   outtextxy(2*x,4*y,"cycles.This will allow oil to go from any storage facility to any
                      other with-");
```

```
outtextxy(2*x,5*y,"out unnecessary duplication of routes and, hence, unnecessary
                    building cost.");
Pause(30*x,24*y);
/*************************************************************************/
pieslice(48*x,14*y,0,359,2);
pieslice(78*x,12*y,0,359,2);
pieslice(88*x,18*y,0,359,2);
pieslice(53*x,18*y,0,359,2);
pieslice(68*x,51*y/4,0,359,2);
pieslice(73*x,18*y,0,359,2);
moveto(48*x,14*y);  lineto(68*x,51*y/4); lineto(78*x,12*y);
lineto(88*x,18*y); lineto(53*x,18*y);    lineto(48*x,14*y);
moveto(53*x,18*y);  lineto(68*x,51*y/4); lineto(73*x,18*y);
outtextxy(56*x,14*y,"a");
outtextxy(72*x,13*y,"b");
outtextxy(81*x,16*y,"c");
outtextxy(79*x,35*y/2,"d");
outtextxy(60*x,35*y/2,"e");
outtextxy(60*x,16*y,"f");
outtextxy(52*x,16*y,"h");
outtextxy(73*x,16*y,"g");
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
moveto(53*x,18*y); lineto(48*x,14*y);
moveto(53*x,18*y); lineto(68*x,51*y/4);
moveto(53*x,18*y); lineto(73*x,18*y);
moveto(73*x,18*y); lineto(88*x,18*y);
moveto(68*x,51*y/4); lineto(78*x,12*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(53*x,18*y); lineto(48*x,14*y);
moveto(53*x,18*y); lineto(68*x,51*y/4);
moveto(53*x,18*y); lineto(73*x,18*y);
moveto(73*x,18*y); lineto(88*x,18*y);
moveto(68*x,51*y/4); lineto(78*x,12*y);
setlinestyle(0,0,3);
```

```
  /*************************************************************************/
  outtextxy(2*x,7*y,"Now in the next graph you see an example of this type of a
                    graph. The edges");
  outtextxy(2*x,8*y,"with dashed lines, namely h, e, f, b, d may serve our
                    purposes.");
  /*************************************************************************/
  Pause(30*x,24*y);
  closegraph();
  videoinit();
}
```

```
/* PROGRAM   : exspan2.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 18, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for spanning trees.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions        */
static void init_graph   (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer          (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
       }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
       }
  forecolor = WHITE;
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
   }
   switch (ch)        {
    case 'y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'Y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'n': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* This routine illustrates spanning trees.                              */
/*************************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE SPAN_2");
   /*********************************************************************/
   pieslice(3*x,14*y,0,359,2);
   pieslice(33*x,12*y,0,359,2);
   pieslice(43*x,18*y,0,359,2);
   pieslice(8*x,18*y,0,359,2);
   pieslice(23*x,51*y/4,0,359,2);
   pieslice(28*x,18*y,0,359,2);
   moveto(3*x,14*y); lineto(23*x,51*y/4); lineto(33*x,12*y);
   lineto(43*x,18*y); lineto(8*x,18*y);    lineto(3*x,14*y);
   moveto(8*x,18*y); lineto(23*x,51*y/4); lineto(28*x,18*y);
   outtextxy(11*x,14*y,"a");
   outtextxy(27*x,13*y,"b");
   outtextxy(36*x,16*y,"c");
   outtextxy(34*x,35*y/2,"d");
   outtextxy(15*x,35*y/2,"e");
   outtextxy(15*x,16*y,"f");
   outtextxy(7*x,16*y,"h");
   outtextxy(28*x,16*y,"g");
   /*********************************************************************/
   pieslice(48*x,14*y,0,359,2);
   pieslice(78*x,12*y,0,359,2);
   pieslice(88*x,18*y,0,359,2);
   pieslice(53*x,18*y,0,359,2);
   pieslice(68*x,51*y/4,0,359,2);
   pieslice(73*x,18*y,0,359,2);
   moveto(48*x,14*y); lineto(68*x,51*y/4); lineto(78*x,12*y);
```

```
lineto(88*x,18*y); lineto(53*x,18*y);    lineto(48*x,14*y);
moveto(53*x,18*y);  lineto(68*x,51*y/4); lineto(73*x,18*y);
outtextxy(56*x,14*y,"a");
outtextxy(72*x,13*y,"b");
outtextxy(81*x,16*y,"c");
outtextxy(79*x,35*y/2,"d");
outtextxy(60*x,35*y/2,"e");
outtextxy(60*x,16*y,"f");
outtextxy(52*x,16*y,"h");
outtextxy(73*x,16*y,"g");
setcolor(backcolor);
moveto(53*x,18*y); lineto(48*x,14*y);
moveto(53*x,18*y); lineto(68*x,51*y/4);
moveto(53*x,18*y); lineto(73*x,18*y);
moveto(73*x,18*y); lineto(88*x,18*y);
moveto(68*x,51*y/4); lineto(78*x,12*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(53*x,18*y); lineto(48*x,14*y);
moveto(53*x,18*y); lineto(68*x,51*y/4);
moveto(53*x,18*y); lineto(73*x,18*y);
moveto(73*x,18*y); lineto(88*x,18*y);
moveto(68*x,51*y/4); lineto(78*x,12*y);
setlinestyle(0,0,3);
/*************************************************************************/
outtextxy(2*x,2*y,"In this example the dashed edges in the graph on the  right
                   hande side forms a");
outtextxy(2*x,3*y,"a spanning tree.");
Pause(30*x,24*y);
outtextxy(2*x,5*y,"If the graph G is a tree, then its only spanning  tree is G itself
                  . As you ");
outtextxy(2*x,6*y,"see in the following graphs.");
Pause(30*x,24*y);
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
moveto(48*x,14*y); lineto(68*x,51*y/4);
```

```
moveto(78*x,12*y); lineto(88*x,18*y);
moveto(68*x,51*y/4); lineto(73*x,18*y);
moveto(3*x,14*y); lineto(23*x,51*y/4);
moveto(33*x,12*y); lineto(43*x,18*y);
moveto(23*x,51*y/4); lineto(28*x,18*y);
setcolor(forecolor);
Pause(30*x,24*y);
moveto(53*x,18*y); lineto(68*x,51*y/4);
moveto(48*x,14*y); lineto(68*x,51*y/4);
moveto(78*x,12*y); lineto(88*x,18*y);
moveto(68*x,51*y/4); lineto(73*x,18*y);
moveto(3*x,14*y); lineto(23*x,51*y/4);
moveto(33*x,12*y); lineto(43*x,18*y);
moveto(23*x,51*y/4); lineto(28*x,18*y);
outtextxy(2*x,8*y,"A graph may have more than one spanning tree. For example,
                    the edges ");
outtextxy(2*x,9*y,"h, a, e, b, d also form  a spanning tree.");
setcolor(backcolor);
moveto(53*x,18*y); lineto(48*x,14*y);
moveto(48*x,14*y); lineto(68*x,51*y/4);
moveto(53*x,18*y); lineto(73*x,18*y);
moveto(73*x,18*y); lineto(88*x,18*y);
moveto(68*x,51*y/4); lineto(78*x,12*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(53*x,18*y); lineto(48*x,14*y);
moveto(48*x,14*y); lineto(68*x,51*y/4);
moveto(53*x,18*y); lineto(73*x,18*y);
moveto(73*x,18*y); lineto(88*x,18*y);
moveto(68*x,51*y/4); lineto(78*x,12*y);
setlinestyle(0,0,3);
/*************************************************************************/
Pause(30*x,24*y);
closegraph();
videoinit();
}
```

```
/* PROGRAM   : exspan3.c
   AUTHOR    : Atilla BAKAN
   DATE      : Apr. 18, 1990
   REVISED   : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for spanning trees.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer           (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
      }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
      }
  forecolor = WHITE;
}
```

```
/*************************************************************************/.
static void confirm_graph_exit(void)
{
   struct _onkey_t *kblist;
   char ch;

   setcolor(backcolor);
   bar(3*x/2,23*y,179*x/2,97*y/4);
   setcolor(quitcolor);
   kblist=chgonkey(NULL);  /* hide any existing hot keys */
   if(_mouse&MS_CURS) mshidecur();
   outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
   ch = getch ();
   while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
      outtextxy(32*x,24*y," Please type y or n");
      ch = getch ();
      if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
      setcolor(backcolor);
      bar(31*x,23*y,69*x,97*y/4);
      setcolor(quitcolor);
    }
    switch (ch)        {
     case 'y': closegraph();
               videoinit();
               exit(0);
               break;
     case 'Y': closegraph();
               videoinit();
               exit(0);
               break;
     case 'n': setcolor(backcolor);
               bar(4*x/3,23*y,30*x,97*y/4);
               bar(31*x,23*y,69*x,97*y/4);
               setcolor(forecolor);
               break;
     case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* This routine illustrates a spanning tree.                             */
/*************************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE SPAN_3");
   /*********************************************************************/
   pieslice(10*x,14*y,0,359,2);
   pieslice(20*x,14*y,0,359,2);
   pieslice(20*x,18*y,0,359,2);
   pieslice(10*x,18*y,0,359,2);
   moveto(10*x,14*y);  lineto(20*x,14*y); lineto(20*x,18*y);
   lineto(10*x,18*y); lineto(10*x,14*y);  lineto(20*x,18*y);
   outtextxy(15*x,27*y/2,"a");
   outtextxy(22*x,16*y,"b");
   outtextxy(15*x,37*y/2,"c");
   outtextxy(8*x,16*y,"d");
   outtextxy(13*x,33*y/2,"e");
   /*********************************************************************/
   outtextxy(2*x,2*y,"The graph in this figure is not a tree because it contains cycles
                      such as");
   outtextxy(2*x,3*y,"a, b, c, d. In order to obtain a tree our procedure will be to
                      delete an edge");
   outtextxy(2*x,4*y,"in each cycle. For example : deleting b from the cycle a, b, c, d
                      gives the");
   outtextxy(2*x,5*y,"following graph.");
   Pause(30*x,24*y);
   /*********************************************************************/
   setcolor(backcolor);
   bar(29*x,23*y,70*x,49*y/2);
   setcolor(forecolor);
   pieslice(40*x,14*y,0,359,2);
```

```
pieslice(50*x,14*y,0,359,2);
pieslice(50*x,18*y,0,359,2);
pieslice(40*x,18*y,0,359,2);
moveto(40*x,14*y); lineto(50*x,14*y);
moveto(50*x,18*y); lineto(40*x,18*y);
lineto(40*x,14*y); lineto(50*x,18*y);
outtextxy(45*x,27*y/2,"a");
outtextxy(45*x,37*y/2,"c");
outtextxy(38*x,16*y,"d");
outtextxy(43*x,33*y/2,"e");
/*************************************************************************/
outtextxy(2*x,7*y,"The resultant graph in the new figure is still not a tree because
                   it the  ");
outtextxy(2*x,8*y,"cycle c, d, e. So we delete an edge in this cycle, say  c.");
Pause(30*x,24*y);
/*************************************************************************/
setcolor(backcolor);
bar(29*x,23*y,70*x,49*y/2);
setcolor(forecolor);
pieslice(70*x,14*y,0,359,2);
pieslice(80*x,14*y,0,359,2);
pieslice(80*x,18*y,0,359,2);
pieslice(70*x,18*y,0,359,2);
moveto(70*x,14*y); lineto(80*x,14*y);
moveto(70*x,18*y); lineto(70*x,14*y); lineto(80*x,18*y);
outtextxy(75*x,27*y/2,"a");
outtextxy(68*x,16*y,"d");
outtextxy(73*x,33*y/2,"e");
outtextxy(2*x,10*y,"As you see the resultant graph in the new figure is a tree.
                    This, then, is");
outtextxy(2*x,11*y,"a spanning  tree for the original graph.");
/*************************************************************************/
Pause(30*x,24*y);
closegraph();
videoinit();
}
```

```
/* PROGRAM  : exspan4.c
   AUTHOR   : Atilla BAKAN
   DATE     : Apr. 18, 1990
   REVISED  : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for an breadth first search
                 implementation.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions        */
static void init_graph   (void);
static void confirm_graph_exit (void);
static void Pause        (int i, int j);
static void register_drivers (void);
extern void settext      (void);

/* tutorial functions    */
static void exer          (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /***********************************************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /***********************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /***********************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
      }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
      }
  forecolor = WHITE;
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
  }
  switch (ch)        {
   case 'y': closegraph();
          videoinit();
          exit(0);
          break;
   case 'Y': closegraph();
          videoinit();
          exit(0);
          break;
  case 'n': setcolor(backcolor);
          bar(4*x/3,23*y,30*x,97*y/4);
          bar(31*x,23*y,69*x,97*y/4);
          setcolor(forecolor);
          break;
  case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* This routine illustrates an implementation of breadth first search.   */
/*************************************************************************/
void exer()
{
   init_graph();
   setcolor(forecolor);
   bar(0,0,MaxX,MaxY);
   rectangle(x,y,MaxX-x,MaxY-y/2);
   outtextxy(38*x,y/2,"EXAMPLE SPAN_4");
   /*********************************************************************/
   pieslice(5*x,7*y,0,359,2);      /* A */
   pieslice(10*x,7*y,0,359,2);     /* B */
   pieslice(35*x,7*y,0,359,2);     /* G */
   pieslice(20*x,4*y,0,359,2);     /* D */
   pieslice(20*x,10*y,0,359,2);    /* E */
   pieslice(20*x,3*y/2,0,359,2);   /* C */
   pieslice(20*x,25*y/2,0,359,2);  /* F */
   outtextxy(3*x,7*y,"A");
   outtextxy(36*x,7*y,"G");
   outtextxy(20*x/2,15*y/2,"B");
   outtextxy(20*x,9*y/2,"D");
   outtextxy(21*x,3*y/2,"C");
   outtextxy(20*x,19*y/2,"E");
   outtextxy(21*x,25*y/2,"F");
   moveto(5*x,7*y); lineto(10*x,7*y); lineto(35*x,7*y);
   moveto(5*x,7*y); lineto(20*x,4*y);   lineto(10*x,7*y);
   moveto(5*x,7*y); lineto(20*x,3*y/2); lineto(20*x,4*y);
   moveto(5*x,7*y); lineto(20*x,10*y);  lineto(10*x,7*y);
   moveto(5*x,7*y); lineto(20*x,25*y/2); lineto(20*x,10*y);
   moveto(20*x,3*y/2); lineto(35*x,7*y);
   moveto(20*x,4*y);   lineto(35*x,7*y);
   moveto(20*x,10*y);  lineto(35*x,7*y);
   moveto(20*x,25*y/2); lineto(35*x,7*y);
   /*********************************************************************/
   outtextxy(45*x,2*y,"k");
```

```
outtextxy(52*x,2*y,"L");
outtextxy(58*x,2*y,"Label");
outtextxy(74*x,2*y,"T");
moveto(44*x,5*y/2); lineto(47*x,5*y/2);
moveto(50*x,5*y/2); lineto(55*x,5*y/2);
moveto(57*x,5*y/2); lineto(67*x,5*y/2);
moveto(70*x,5*y/2); lineto(78*x,5*y/2);
outtextxy(2*x,14*y,"THE WAY WE APPLIED THE ALGORITHM");
moveto(3*x/2,29*y/2); lineto(48*x,29*y/2);
/*************************************************************************/
outtextxy(2*x,15*y,". Initially pick (arbitrarily) A. ");
Pause(7*x,24*y);
outtextxy(45*x,3*y,"0"); /* initialize k */
/*************************************************************************/
outtextxy(2*x,17*y,". Label A with 0 (As you see, we will");
outtextxy(2*x,18*y,"  show the label in paranthesis near");
outtextxy(2*x,19*y,"  the vertex).");
Pause(7*x,24*y);
outtextxy(58*x,3*y,"A <- 0");
outtextxy(2*x,13*y/2,"(0)");
/*************************************************************************/
outtextxy(2*x,20*y,". Put A in L.");
Pause(7*x,24*y);
outtextxy(52*x,3*y,"A");
/*************************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,50*x,24*y);
setcolor(forecolor);
/*************************************************************************/
outtextxy(2*x,15*y,". Now put all the vertices adjacent to");
outtextxy(2*x,16*y,"  A (currently largest labeled) in L");
outtextxy(2*x,17*y,"  and label them with k + 1 = 1");
Pause(7*x,24*y);
outtextxy(52*x,4*y,"C");
outtextxy(52*x,5*y,"D");
```

```
outtextxy(52*x,6*y,"B");
outtextxy(52*x,7*y,"E");
outtextxy(52*x,8*y,"F");
/*************************************************************************/
outtextxy(58*x,4*y,"C <- 1");
outtextxy(58*x,5*y,"D <- 1");
outtextxy(58*x,6*y,"B <- 1");
outtextxy(58*x,7*y,"E <- 1");
outtextxy(58*x,8*y,"F <- 1");
outtextxy(19*x,13*y,"(1)");
outtextxy(22*x,3*y/2,"(1)");
outtextxy(19*x,5*y,"(1)");
outtextxy(8*x,27*y/4,"(1)");
outtextxy(19*x,9*y,"(1)");
/*************************************************** ***************/
outtextxy(2*x,18*y,". Put the edges connecting these ");
outtextxy(2*x,19*y,"  vertices to A in the tree T.");
Pause(7*x,24*y);
outtextxy(72*x,4*y,"(A,C)");
outtextxy(72*x,5*y,"(A,D)");
outtextxy(72*x,6*y,"(A,B)");
outtextxy(72*x,7*y,"(A,E)");
outtextxy(72*x,8*y,"(A,F)");
setcolor(backcolor);
moveto(5*x,7*y); lineto(20*x,3*y/2);
moveto(5*x,7*y); lineto(20*x,4*y);
moveto(5*x,7*y); lineto(10*x,7*y);
moveto(5*x,7*y); lineto(20*x,10*y);
moveto(5*x,7*y); lineto(20*x,25*y/2);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(5*x,7*y); lineto(20*x,3*y/2);  /* add (A,C) to T */
moveto(5*x,7*y); lineto(20*x,4*y);    /* add (A,D) to T */
moveto(5*x,7*y); lineto(10*x,7*y);    /* add (A,B) to T */
moveto(5*x,7*y); lineto(20*x,10*y);   /* add (A,E) to T */
moveto(5*x,7*y); lineto(20*x,25*y/2); /* add (A,F) to T */
```

```
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,20*y,". Increment k and go to Step 2.");
Pause(7*x,24*y);
outtextxy(45*x,4*y,"1");
/*****************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,55*x,24*y);
setcolor(forecolor);
/*****************************************************************/
outtextxy(2*x,15*y,". Now pick G since it is unlabeled and");
outtextxy(2*x,16*y,"  adjacent to the (currently) largest");
outtextxy(2*x,17*y,"  labeled vertex C.");
Pause(7*x,24*y);
outtextxy(52*x,9*y,"G");
/*****************************************************************/
outtextxy(2*x,18*y,". Label G with k + 1 = 2.");
Pause(7*x,24*y);
outtextxy(58*x,9*y,"G <- 1");
outtextxy(37*x,7*y,"(2)");
/*****************************************************************/
outtextxy(2*x,19*y,". Put the edge connecting G to C ");
outtextxy(2*x,20*y,"  in the tree T.");
Pause(7*x,24*y);
outtextxy(72*x,9*y,"(C,G)");
setcolor(backcolor);
moveto(20*x,3*y/2);  lineto(35*x,7*y);
setcolor(forecolor);
setlinestyle(3,0,3);
moveto(20*x,3*y/2);  lineto(35*x,7*y);  /* add (C,G) to T */
setlinestyle(0,0,3);
/*****************************************************************/
outtextxy(2*x,21*y,". Increment k and go to Step 2.");
Pause(7*x,24*y);
outtextxy(45*x,9*y,"2");
```

```
    /*************************************************************************/
    Pause(7*x,24*y);
    setcolor(backcolor);
    bar(3*x/2,59*y/4,55*x,24*y);
    setcolor(forecolor);
    /*************************************************************************/
    outtextxy(2*x,15*y,". As you see L contains all the vertices");
    outtextxy(2*x,16*y," of the graph G. This means we are done.");
    /*************************************************************************/
    Pause(30*x,24*y);
    closegraph();
    videoinit();
}
```

```
/* PROGRAM  : exspan5.c
   AUTHOR   : Atilla BAKAN
   DATE     : Apr. 18, 1990
   REVISED  : Apr. 18, 1990

   DESCRIPTION : This routine draws the example graph for  a breadth first search
                 implementation.


   MACHINE/COMPILER : This program is written with IBM pc by using Turbo
                      C compiler Version 2.0.
*/

/* header files */
#include <graphics.h>
#include "cxldef.h"
#include "cxlkey.h"
#include "cxlmou.h"

#if defined(__TURBOC__)                 /* Turbo C */
   #include <dir.h>
#else
   #include <direct.h>                  /* all others */
#endif

#if defined(M_I86) && !defined(__ZTC__)         /* MSC/QuickC */
   #define bioskey(a)      _bios_keybrd(a)
   #define findfirst(a,b,c) _dos_findfirst(a,c,b)
   #define findnext(a)     _dos_findnext(a)
   #define ffblk           find_t
   #define ff_name         name
#elif defined(__ZTC__)                  /* Zortech C/C++ */
   #define ffblk           FIND
   #define ff_name         name
   #define ff_attrib       attribute
#endif
```

```
#define _GRAPH_T_DEFINED

/* function prototypes */

/*  Utility functions         */
static void init_graph    (void);
static void confirm_graph_exit (void);
static void Pause         (int i, int j);
static void register_drivers (void);
extern void settext       (void);

/* tutorial functions     */
static void exer           (void);

/*************************************************************************/
/* graphic initialization variables                                      */
/*************************************************************************/
int curr_mode;
int graphdriver;
int graphmode;
int graph_error;
int backcolor;
int forecolor;
int quitcolor;
int x, y, MaxX, MaxY;


/*************************************************************************/
/* This function is used for including drivers to the executable code    */
/*************************************************************************/
static void register_drivers(void)
{
  if(registerbgidriver(CGA_driver) < 0) exit(1);
  if(registerbgidriver(EGAVGA_driver) < 0) exit(1);
  if(registerbgidriver(ATT_driver) < 0) exit(1);
}
```

```
/*************************************************************************/
/* This fuction initializes the necessary graphical routines             */
/*************************************************************************/
static void init_graph(void)
{
  int xasp, yasp;

  register_drivers();
  graphdriver = DETECT;
  /*************************** ***************************************/
  initgraph(&graphdriver,&graphmode,"");
  graph_error = graphresult();
  /*************************************************************************/
  if(graph_error < 0){
  puts(grapherrormsg(graph_error));
  exit(1);
   }
  /*************************************************************************/
   MaxX = getmaxx();
   MaxY = getmaxy();
   x = MaxX/80;
   y = MaxY/25;
  settext();
  if ((graphmode == CGAHI) || (graphmode == MCGAMED) || (graphmode ==
     ATT400MED) || (graphmode == MCGAHI) || (graphmode == ATT400HI)) {
       setfillstyle(SOLID_FILL,BLACK);
       backcolor = BLACK;
       quitcolor = WHITE;
      }
   else {
       setfillstyle(SOLID_FILL,BLUE);
       backcolor = BLUE;
       quitcolor = RED;
      }
  forecolor = WHITE;
}
```

```
/*************************************************************************/
static void confirm_graph_exit(void)
{
  struct _onkey_t *kblist;
  char ch;

  setcolor(backcolor);
  bar(3*x/2,23*y,179*x/2,97*y/4);
  setcolor(quitcolor);
  kblist=chgonkey(NULL);  /* hide any existing hot keys */
  if(_mouse&MS_CURS) mshidecur();
  outtextxy(3*x/2,24*y,"Quit! Are you sure (y/n)?");
  ch = getch ();
  while (!((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))) {
     outtextxy(32*x,24*y," Please type y or n");
     ch = getch ();
     if((ch == 'y') || (ch == 'n') || (ch == 'Y') || (ch == 'N'))
     setcolor(backcolor);
     bar(31*x,23*y,69*x,97*y/4);
     setcolor(quitcolor);
   }
   switch (ch)        {
    case 'y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'Y': closegraph();
              videoinit();
              exit(0);
              break;
    case 'n': setcolor(backcolor);
              bar(4*x/3,23*y,30*x,97*y/4);
              bar(31*x,23*y,69*x,97*y/4);
              setcolor(forecolor);
              break;
    case 'N': setcolor(backcolor);
```

```
            bar(4*x/3,23*y,30*x,97*y/4);
            bar(31*x,23*y,69*x,97*y/4);
            setcolor(forecolor);
            break;
     default : break;
     }
   hidecur();
   if(_mouse&MS_CURS) msshowcur();
   chgonkey(kblist);     /* restore any hidden hot keys */
}
/*************************************************************************/
/* This function sets the text default values                            */
/*************************************************************************/
static void settext(void)
{
  settextstyle(0,0,0);
  setlinestyle(0,4,3);
  settextjustify(HORIZ_DIR,CENTER_TEXT);
}
/*************************************************************************/
/* Equivalent of press_a_key function for graphics screen                */
/*************************************************************************/
 void Pause(i,j)
 int i, j;
  {
  settext();
  outtextxy(i,j,">>>PRESS A KEY TO CONTINUE...<<<");
  if(waitkey()==ESC) confirm_graph_exit();
  }
/*************************************************************************/
/* main routine that calls exer routine                                  */
/*************************************************************************/
void main()
{
  exer();
}
```

```
/*************************************************************************/
/* This routine illustrates an implementation of breadth first search    */
/*************************************************************************/
void exer()
{
  init_graph();
  setcolor(forecolor);
  bar(0,0,MaxX,MaxY);
  rectangle(x,y,MaxX-x,MaxY-y/2);
  outtextxy(38*x,y/2,"EXAMPLE SPAN_5");
  /*************************************************************************/
  pieslice(10*x,4*y,0,359,2);     /* A */
  pieslice(20*x,4*y,0,359,2);     /* B */
  pieslice(30*x,4*y,0,359,2);     /* C */
  pieslice(40*x,4*y,0,359,2);     /* D */
  pieslice(10*x,7*y,0,359,2);     /* E */
  pieslice(20*x,7*y,0,359,2);     /* F */
  pieslice(30*x,7*y,0,359,2);     /* G */
  pieslice(40*x,7*y,0,359,2);     /* H */
  pieslice(20*x,10*y,0,359,2);    /* I */
  pieslice(30*x,10*y,0,359,2);    /* J */
  outtextxy(10*x,7*y/2,"A");
  outtextxy(20*x,7*y/2,"B");
  outtextxy(30*x,7*y/2,"C");
  outtextxy(40*x,7*y/2,"D");
  outtextxy(8*x,7*y,"E");
  outtextxy(18*x,15*y/2,"F");
  outtextxy(32*x,15*y/2,"G");
  outtextxy(42*x,7*y,"H");
  outtextxy(20*x,21*y/2,"I");
  outtextxy(30*x,21*y/2,"J");
  moveto(10*x,4*y); lineto(20*x,4*y); lineto(20*x,7*y);
  lineto(10*x,7*y); lineto(10*x,4*y);
  moveto(30*x,7*y); lineto(30*x,10*y);lineto(20*x,10*y);
  lineto(20*x,7*y); lineto(30*x,7*y);
  moveto(20*x,10*y);lineto(30*x,7*y);
```

```
moveto(30*x,7*y); lineto(30*x,4*y); lineto(40*x,4*y);
lineto(40*x,7*y);lineto(30*x,7*y),
moveto(30*x,4*y);lineto(40*x,7*y);
/*************************************************************************/
outtextxy(45*x,2*y,"k");
outtextxy(52*x,2*y,"L");
outtextxy(58*x,2*y,"Label");
outtextxy(74*x,2*y,"T");
moveto(44*x,5*y/2); lineto(47*x,5*y/2);
moveto(50*x,5*y/2); lineto(55*x,5*y/2);
moveto(57*x,5*y/2); lineto(67*x,5*y/2);
moveto(70*x,5*y/2); lineto(78*x,5*y/2);
outtextxy(2*x,14*y,"THE WAY WE APPLIED THE ALGORITHM");
moveto(3*x/2,29*y/2); lineto(48*x,29*y/2);
/*************************************************************************/
outtextxy(2*x,15*y,". Initially pick (arbitrarily) A. ");
Pause(7*x,24*y);
outtextxy(45*x,3*y,"0"); /* initialize k */
/*************************************************************************/
outtextxy(2*x,17*y,". Label A with 0 (As you see, we will");
outtextxy(2*x,18*y," show the label in paranthesis near");
outtextxy(2*x,19*y," the vertex).");
Pause(7*x,24*y);
outtextxy(58*x,3*y,"A <- 0");
outtextxy(6*x,4*y,"(0)");
/*************************************************************************/
outtextxy(2*x,21*y,". Put A in L.");
Pause(7*x,24*y);
outtextxy(52*x,3*y,"A");
/*************************************************************************/
Pause(7*x,24*y);
setcolor(backcolor);
bar(3*x/2,59*y/4,50*x,49*y/2);
setcolor(forecolor);
/*************************************************************************/
outtextxy(2*x,15*y,". Now put all the vertices adjacent to");
```