

AD733184

United States Naval Postgraduate School



THE SIS

CAI-BASIC

A Program to Teach the
Programming Language "BASIC"

by

Thomas Anthony Barry

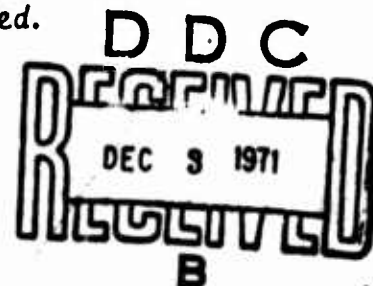
Reproduced by
**NATIONAL TECHNICAL
INFORMATION SERVICE**
Springfield, Va. 22151

Thesis Advisor

A. B. Roberts

September 1971

Approved for public release; distribution unlimited.



**Best
Available
Copy**

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computer aided instruction						
Computer assisted instruction						
Programmed instruction						

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California 93940		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE CAI-BASIC A program to Teach the Programming Language "BASIC"			
4. DESCRIPTIVE NOTES (Type of report and inclusive dates) Master's Thesis; September 1971			
5. AUTHOR(S) (First name, middle initial, last name) Thomas Anthony Barry			
6. REPORT DATE September 1971		7a. TOTAL NO. OF PAGES	7b. NO. OF REFS 10
8a. CONTRACT OR GRANT NO.		9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.			
c.		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.			
10. DISTRIBUTION STATEMENT Approved for public release; distribution unlimited.			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940	
13. ABSTRACT This paper presents a computer aided instruction program that fulfills the objectives of teaching a simple programming language, interpreting student responses, and executing and editing student programs. The CAI-BASIC program is written in FORTRAN IV, level G, and executes on IBM-2741 terminals while running under the CP-67/CMS time sharing system on the U.S. Naval Postgraduate School's IBM-360/67 computer system. The instructional phase of CAI-BASIC presents the fundamentals of "BASIC," a simple user oriented language, in seven lessons. During the instructional sessions the student is presented material and, based on his response to questions, he is routed to the next sequence of instructions. The execution phase of CAI-BASIC allows execution of "BASIC" programs, and has an optional debug feature that provides a trace of program variables to aid the student in finding programming errors. In the event of programming errors the user may enter an edit mode to correct mistakes in his program.			

CAI-BASIC

A Program to Teach the
Programming Language "BASIC"

by

Thomas Anthony Barry
Lieutenant, United States Navy
B.S., United States Naval Academy, 1965

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1971

Author

Thomas A. Barry

Approved by:

Alan B. Roberts

Thesis Advisor

R. E. Gaskell

Chairman, Department of Mathematics

William H. Clavin

Academic Dean

ABSTRACT

This paper presents a computer aided instruction program that fulfills the objectives of teaching a simple programming language, interpreting student responses, and executing and editing student programs. The CAI-BASIC program is written in FORTRAN IV, level G, and executes on IBM- 2741 terminals while running under the CP-67/CMS time sharing system on the U. S. Naval Postgraduate School's IBM-360/67 computer system. The instructional phase of CAI-BASIC presents the fundamentals of "BASIC," a simple user oriented language, in seven lessons. During the instructional sessions the student is presented material and, based on his response to questions, he is routed to the next sequence of instructions. The execution phase of CAI-BASIC allows execution of "BASIC" programs, and has an optional debug feature that provides a trace of program variables to aid the student in finding programming errors. In the event of programming errors the user may enter an edit mode to correct mistakes in his program.

TABLE OF CONTENTS

I.	INTRODUCTION	4
	A. DEFINITION OF CAI.....	4
	B. CAI DEVELOPMENTS	4
	C. OBJECTIVES	6
II.	DESCRIPTION OF CAI-BASIC.....	7
	A. INSTRUCTION PHASE	7
	B. EXECUTION PHASE.....	8
III.	LIMITATIONS AND EXTENSIONS	10
IV.	CONCLUSION	12
	APPENDIX A USING CAI-BASIC	13
	APPENDIX B CAI-BASIC PROGRAM.....	16
	APPENDIX C TERMINAL SESSION WITH CAI-BASIC	70
	BIBLIOGRAPHY.....	119
	INITIAL DISTRIBUTION LIST	120
	FORM DD 1473	121

I. INTRODUCTION

During the last few years many sophisticated projects in Computer Assisted Instruction (CAI) have evolved. The potential of this area has captured the imagination of researchers and the public at large. Yet the U. S. COMMISSION ON INSTRUCTIONAL TECHNOLOGY (ref. 9) found that the status of instructional technology in American Education was low in both quantity and quality. There are many reasons that CAI has not been accepted with any marked degree of enthusiasm by educators in general. The major reasons are the exorbitant costs of individualized instruction and illusions over just what CAI is.

A. DEFINITION OF CAI

Computer Assisted Instruction is machine augmented instruction that differs from Programmed Instruction or simple page turning machines in that the computer system has logic and memory capabilities to assist in the instructional process. The main advantage in CAI is that it can individualize education. A student can proceed at his own pace, spending more time on difficult material, and quickly covering material which comes easy to him.

B. CAI DEVELOPMENTS

A. G. OETTINGER, in his book RUN, COMPUTER, RUN (ref. 6) states that educators and the computer industry are equally to blame for the failure of CAI to realize its potential. Instructional technology has been force fed,

oversold, and prematurely applied; and as a result the educators are wary of false promises. OETTINGER feels that colleges and universities will be an effective proving ground for future educational technology. However, at the present time, of the thousands of colleges and universities in the nation only twenty-five are considered major CAI centers.¹

A common problem confronting the nation's colleges and universities is that of providing instruction in elementary computer programming.² The popularity of the computer science field, and the range of application of computers in everyday life means that the demand for people who know how to communicate with computers is growing rapidly. The solution to the problem is the computer itself. "Whatever the state of CAI with respect to other subjects, the computer is the ideal instrument for teaching its own use."³

W. R. SMITH and J. L. YOUNG, graduate students at the Naval Postgraduate School, presented a proposal for the use of computer Assisted Instruction at the Naval Postgraduate School (ref. 10). As this introduction has done, they created an awareness for the need of CAI, and they made specific recommendations to provide a balanced program for the entire community to emulate. In particular they recommended commencing student projects to probe the potentials of CAI

1. STOLUROW, L. M., "Computer Assisted Instruction," REPORT OF A CONFERENCE, U.S. Department of HEW, p. 49, 1969.

2. FENICHEL, R. R., WEIZENBAUM, J., AND YOCHELSON, J. C. "Program to Teach Programming," COMMUNICATION OF THE ACM, v. 13, p. 141, March 1970.

3. IBID., p. 141-142.

at the Naval Postgraduate School.

It was with these ideas about using the computer to teach computer programming and providing a CAI base at the Naval Postgraduate School that CAI-BASIC was undertaken.

C. OBJECTIVES

The objective of CAI-BASIC was to develop a CAI program to teach a computer programming language on the Naval Postgraduate School's time-sharing terminals. The programming language chosen was "BASIC" (Beginners All-purpose Symbolic Instruction Code), a computer language developed at Dartmouth College in the mid-1960's. "BASIC" was selected because of its simplicity yet fairly large range of application for the user who wants to take advantage of computer processing. It was felt that "BASIC" would provide the average non-computer oriented graduate student with satisfactory results in a minimal amount of study time.

With this objective in mind three sub-goals were determined to be essential for the CAI-BASIC project: to be able to interpret student responses, to be able to execute "BASIC" programs in order to give the student sufficient programming practice, and to be able to allow editing of "BASIC" programs.

II. DESCRIPTION OF CAI-BASIC

The CAI-BASIC program is written in FORTRAN IV, level G, and executes on IBM-2741 terminals while running under CP-67/CMS Time Sharing System on the Naval Postgraduate School's IBM-360/67 Computer System. The "BASIC" language implemented in CAI-BASIC is standard non-extended "BASIC" (refs. 3, 5, 10).

The CAI-BASIC program consists of two phases; an instruction phase, and an execution and editing phase. A detailed discussion of the use of CAI-BASIC is presented in APPENDIX A.

A. INSTRUCTION PHASE

The instruction phase presents fundamental concepts common to most programming languages: identifiers, variables, iteration, branching, sub-routines, built in functions, and recursion. The instruction set consists of seven lesson modules which, for reasons of coherency, are dependent upon each other. The underlying philosophy in preparing the lesson modules was that an important idea should be presented only after a need for it has clearly been established.⁴

Each lesson consists of instruction sequences followed by questions, evaluation of student responses, routing to the next sequence of instruction,

4. FENICHEL, OP. CIT., p. 142.

and a summary of the lesson followed by questions and/or problems to program. The student is allowed to progress through the lessons at his own speed, and allowed to review lessons or execute programs at the completion of each lesson.

Student responses are interpreted by one of two methods depending upon the type of question asked. When the question is of the true-false, multiple choice or give the answer type then the response is compared with a pre-stored result. When the student is asked to reply with a "BASIC" statement or to write a "BASIC" program, the CAI-BASIC compiler interprets the "BASIC" statements for syntactic correctness. When the student writes a program he can compare his answer with a pre-stored result, and if the answer is wrong the student is shown a solution program.

B. EXECUTION PHASE

The heart of the execution phase is the "BASIC" compiler that is used at the Naval Postgraduate School's Computer Center to execute "BASIC" programs in a batch mode. This compiler, written in FORTRAN IV, level G, was modified from a batch processing compiler to a line-by-line interpreter and incorporated into the CAI-BASIC program, and called the CAI-BASIC COMPILER.

The CAI-BASIC COMPILER has an added feature to aid students in debugging programs on the terminal. The added feature is called a "DEBUG" function and it produces a list of numeric and alpha-numeric data that were put into the program; and, in addition it produces a trace of all simple variables as they are assigned values during program execution.

At the completion of program execution or when an execution error occurs, the student has an option to enter the CAI-BASIC Edit mode to correct his program one line at a time. When all editing is completed the program is executed again.

III. LIMITATIONS AND EXTENSIONS

The major limitation of CAI-BASIC is the absence of any supervision over the student's progress as he proceeds through the instruction phase. A possible extension to this project would be to add a supervisory routine to maintain the student's progress and to keep a record of his errors. Thus, having the student's progress level and error record, CAI-BASIC could be tailored to instruct the student at his own learning level. In other words, CAI-BASIC could be made into a more completely interactive teaching program (refs. 1, 4).

An additional feature that would aid the student interaction is a communication link between the student and a professor so that student questions can be answered. R. R. FENICHEL (refs. 1, 2) described how students enter a special mode to type questions during the instruction session on the terminal. Then at the beginning of the student's next instruction session, all of his previous questions are answered on the terminal. FENICHEL refers to this as the "mailbox" system. Students type questions for the "mailbox" and the professor replies with answers or pertinent information for the "mailbox."

Another limitation to the existing CAI-BASIC program is the inability of CAI-BASIC to monitor the student when he enters the execution phase to write programs. The CAI-BASIC system does not oversee the student, beyond checking for syntax errors in his program. There is no method of inspecting his

programs from a tutorial point of view in order to help with semantic programming errors.

A foreseeable development to bring CAI-BASIC into the Artificial Intelligence field would be to make CAI-BASIC an "intelligent" tutor. This would remove the present inflexibility of interpreting student responses with pre-stored answers and open the possibility of CAI-BASIC understanding the logic of student responses and student programs.

IV. CONCLUSION

As stated in the objectives, the goal of CAI-BASIC was to develop a CAI program to teach a computer programming language on the Naval Post-graduate School's terminal system. The technical aim of completing the project and bringing it to an operating level has been met. However, an evaluation of the practicality of the project is still in the speculative stage.

Several computer and non-computer oriented students have tried CAI-BASIC and they found it to be both understandable and beneficial, but a full scale evaluation of the effectiveness of CAI-BASIC as a teaching tool is not possible from this limited sampling. It is sufficient to say that initial indications are that CAI-BASIC can provide a satisfactory means of learning a simple but capable programming language in a minimal amount of time. The student's participation in the learning experience encourages him to teach himself and to practice new programming skills. It is hoped that future use of CAI-BASIC by students will demonstrate its practicality.

Although the scope of CAI-BASIC was not particularly broad, the results obtained provide a basis for future CAI projects. The foundation has been laid, and the limitations and extensions of the previous section suggest a direction for future CAI efforts. Many previous CAI projects have suffered from too many or too large a scope of objectives, and as a result they never realize any practical results. This project has met its objectives and, hopefully, future CAI projects will benefit from its results.

APPENDIX A

In its present configuration the CAI-BASIC program occupies approximately ten cylinders of private disk space and requires an overlaying routine to execute under the control of CP-67/CMS which limits users to a 175K virtual machine. The overlaying routine, written by a Naval Postgraduate School Computer Center system programmer, initially loads only lesson 1 when CAI-BASIC is loaded into core. When any lesson other than lesson 1 is used, it is overlaid onto lesson 1.

The overlaying routine allows students to log on to CP-67/CMS as general users and to link into CAI-BASIC. This allows CAI-BASIC to be available to all of the terminal stations. The user gains access to CAI-BASIC by logging on the terminal as follows:

```
login yyyygxx      (xx is the terminal nr.)
                   (yyyy is your user nr.)
ENTER PASSWORD
npg
ENTER 4-DIGIT PROJECT NUMBER.... ETC.
0623cs04
READY AT (TIME) ON (DATE)
CP
link 0909p 191 192
ENTER PASSWORD
teach
SET TO READ ONLY
1 cms
CMS.. VERSION 01/21/71
login 192 t,p
**T (192) READ ONLY**
teach
```

Small print is typed in by the user, and the capitalized print is the computer terminal response.

The CAI-BASIC program consists of two phases, an instruction phase and an execution and editing phase. After logging on to the system, the user enters the main routine, CAIBAS, which directs the general flow of CAI-BASIC.

If the user has not used CAI-BASIC before he enters the instruction phase, he is presented with an introduction to CAI-BASIC, and then is sequentially guided through the lessons. After each lesson the user is given the opportunity to terminate his session, to review a lesson, to enter the execution phase, or to go on to the next lesson.

If the user has used CAI-BASIC before, he is given the opportunity to review any lesson or to go to the next lesson in his learning sequence.

When the execution and editing routine, TEST1, is entered the user is allowed to execute standard "BASIC" programs (refs. 3,5,10). The Test1 routine incorporates the "BASIC" compiler, COMPLR, which interprets each "BASIC" input statement. Each "BASIC" statement is input one line at a time; and, if a syntax error occurs on the input, COMPLR prints an error message. The user can then input the correct "BASIC" statement. When the "end" statement is input, the program is checked for global errors; and, if none occur, the program is executed.

When global errors or execution errors occur, the user is given the choice of entering the edit mode of TEST1 to correct his program, or using the "DEBUG" feature to find his programming errors.

The edit mode allows the user to delete, add to, or correct "BASIC"

statements in his program one line at a time. The user is given a listing of his program with reference numbers for editing purposes. To edit a program the user is first asked for the statement reference number, and then asked for the specific edit command.

A statement is deleted by typing in the reference number of the statement to be deleted, hitting the carriage return, and then typing in the command "DEL. "

A statement is added 'after' the reference number typed in by typing in the command "ADD1" followed by the "BASIC" statement to be added. To add a statement before the first statement in the program, the reference number zero (0) is used.

A statement is corrected by typing in the reference number of the statement to be corrected, hitting carriage return, and then typing in the 'entire' correct "BASIC" statement.

The "DEBUG" feature is designed to provide useful information to the user who has encountered an execution error in his program. It is used by adding the key word "DEBUG" as a statement in the user's program.

The "DEBUG" feature gives the user a list of all numeric and alpha-numeric data that were used in the program; and, in addition, it produces a trace of all simple variables with their values as they are assigned values during program execution.

APPENDIX B

CAI-BASIC : A PROGRAM TO TEACH THE PROGRAMMING LANGUAGE "BASIC"

T.A. BARRY

U.S.NPS. 8/17/71

THE FILES USED IN CAI-BASIC UNDER CP-67/CMS ARE AS FOLLOWS :

MAIN PROGRAM----- CAIBAS

SUBROUTINES-----INITIAL
CRUNCH
TEST1
LESON1--> LESC7

OVERLAYING ROUTINES FOR CP-67/CMS :

SEARCH ALOAD EXIT

THE FOLLOWING SUBROUTINES ARE PART OF THE CAIBASIC COMPILER,
AS MODIFIED FROM THE NPS "BASIC" COMPILER USED FOR BATCH PROGRAMS,
BUT ARE NOT INCLUDED AS PART OF THE CAI-BASIC PROGRAM :

ACON	COMPLR	ERR	INSNO
APRINT	CUNV	DUMMY	LTL
BUFFIL	CPRINT	EVAL	RANF
CEND	DIM	EXEC	TEST

SYMBOL TABLE : (GLOBAL SYMBOLS AND VARIABLES)

CARDP	VECTOR HOLDING STUDENT INPUT
ILNGTH	VECTOR HOLDING STUDENT INPUT WITH BLANKS REMOVED
ASTRSK	LENGTH OF INPUT IN VECTOR CARDP
ALPHA	WHEN CARDP(1)=ASTRSK THE USER HAS MADE A TYPING ERROR
DIGIT	VECTOR HOLDING THE LETTERS OF THE ALPHABET
ALPHA(25)	VECTOR HOLDING THE DIGITS 0-->9
ALPHA(14)	THE LETTER 'Y'
	THE LETTER 'N'

CCCCCCCCCCCCCCCCCCCC

CAI000010
CAI000030
CAI000040
CAI000050
CAI000060

CAIBAS IS THE MAIN PROGRAM. IT IS A COMPUTER AIDED INSTRUCTION
PROGRAM WHICH TEACHES FUNDAMENTAL PROGRAMMING CONCEPTS USING
THE LANGUAGE BASIC. THE HEART OF THE PROGRAM PROVIDES INSTRUCTION
AND INTERPRETES STUDENT RESPONSES. SOME ADDED FEATURES ARE
EXECUTION AND EDITING OF BASIC PROGRAMS IN "TEST1"
SUBROUTINES CALLED FROM CAIBAS ARE:
INITIAL
CRUNCH
TEST1
LESON1 --> LESON7
SEARCH
ALOAD

SYMBOL TABLE : (LOCAL SYMBOLS AND VARIABLES)

LSNUM CURRENT LESSON NUMBER
OLD FLAG . SET TO 1 IF USER HAS USED CAI-BASIC BEFORE
LES1 --> LES7 VARIABLES FOR OVERLAYING ROUTINES

CAI000070
CAI000080
CAI000090
CAI000100
CAI000110
CAI000120
CAI000130
CAI000140
CAI000150

COMMON
STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRTB(10), LISTST(100), ISTLST(100),
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PARRT, DECMAL, PLUS, CMINUS, SLASH, COMMA,
PARLET, ASTRSK, BLANK
INTERP, IEXERR
LES1/.LESON1 ./.
LES2/.LESON2 ./.
LES3/.LESON3 ./.
LES4/.LESON4 ./.
LES5/.LESON5 ./.
LES6/.LESON6 ./.
LES7/.LESON7 ./.
CALL SEARCH
CALL INITIAL
LSNUM=G
OLD=0

CAI000160

CAI000180
CAI000190
CAI000200
CAI000210

99 * 5X, 2. IF YOU MAKE A TYPING ERROR WHILE MAKING ANY RESPONSE ://
* 1. WHEN ASKED FOR A RESPONSE, TYPE IN THE CORRECT REPLY AND ://
* ON THE TERMINAL WITH CAI-BASIC : THERE ARE ONLY A FEW ://
* SIMPLE RULES TO REMEMBER IN ORDER TO HAVE A SUCCESSFUL SESSION ://
* 0. HI, WELCOME TO CAI-BASIC. *
WRITE(6,99)
FORMAT(0,0)
* 5X, 2. IF YOU MAKE A TYPING ERROR WHILE MAKING ANY RESPONSE ://
* 1. WHEN ASKED FOR A RESPONSE, TYPE IN THE CORRECT REPLY AND ://
* ON THE TERMINAL WITH CAI-BASIC : THERE ARE ONLY A FEW ://
* SIMPLE RULES TO REMEMBER IN ORDER TO HAVE A SUCCESSFUL SESSION ://
* 0. HI, WELCOME TO CAI-BASIC. *

CAI000230
CAI000240

```

*OR INPUT TYPE IN FOUR DOLLAR SIGNS ($$$$), AFTER THE ERROR OR: CA100260
**ANYWHERE ON THAT INPUT LINE AND HIT CARRIAGE RETURN THE CA100270
**ENTIRE LINE WILL THEN BE IGNORED AND YOU CAN TYPE IN THE CORRECT: CA100280
** INPUT OR RESPONSE // 5X;
** IF AT ANY POINT IN THE SESSION YOU WANT TO STOP THE SESSION: CA100300
** TYPE IN THE WORD "QUIT" AS SOON AS YOU ARE ASKED FOR THE NEXT: CA100310
** RESPONSE, HIT CARRIAGE RETURN, THEN HIT ATTN KEY AND TYPE LOGOUT
** // 5X;
** 4. DURING YOUR TERMINAL SESSION CAI-BASIC WILL HALT OCCASIONALLY: CA100330
** TO LET YOU READ A SEQUENCE OF INFORMATION WHEN YOU ARE CA100340
** READY TO CONTINUE TYPE IN "GO", AND HIT CARRIAGE RETURN // 5X
** 5. DURING YOUR TERMINAL SESSION YOU MAY NOTICE THAT CA100360
** THE TYPING IS NOT ALWAYS PERFECT. SOME DAYS THE COMPUTER IS: CA100370
** NOT UP TO PAR AND YOU WILL HAVE TO ADJUST TO THE MINOR IRRITANT: CA100390
** IF YOU ONLY WANT TO EXECUTE PROGRAMS AT THIS TIME THEN CA100400
** REPLY: YES: OTHERWISE REPLY: NO // CA100410
** READ(5,101,END=300)CARD CA100420
** CALL CRUNCH(ILNGTH) CA100430
** IF(CARDP(1).EQ.ASTRSK) GO TO 1 CA100440
** IF(CARDP(1).NE.ALPHA(25)) GO TO 2 CA100450
** CA100460
** CA100470
EXECUTION ONLY PHASE
CALL TEST1
WRITE(6,109)
READ(5,101,END=401)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ALPHA(25)) RETURN
IF (CARDP(1).EQ.ALPHA(14)) GO TO 402
WRITE(6,104)
GO TO 400
IF(CARDP(1).EQ.ALPHA(14)) GO TO 3
WRITE(6,104)
GO TO 1
WRITE(6,100)
FORMAT(0:0) IF THIS IS YOUR FIRST SESSION WITH CAIBASIC, THEN:
*REPLY: YES: OTHERWISE REPLY: NO //
4 READ(5,101,END=302)CARD
FORMAT(8:8)
CALL CRUNCH(ILNGTH)
IF(CARDP(1).NE.ALPHA(14))GO TO 20
OLD USER. DETERMINE PROGRESS LEVEL AND ROUTE TO DESIRED LESSON
402 OLD=1
WRITE(6,102)
102 FORMAT(0:0)
NOW YOU MAY CHOOSE TO REVIEW ANY LESSONS:

```

```

*OR TO BEGIN WHERE YOU ENDED DURING YOUR LAST SESSION.*)
WRITE(6,105)
FORMAT(10,10X,'THE LESSONS ARE AS FOLLOWS : ',10X,' LESSON 1
105 *PROGRAM FORMAT AND BASIC DEFINITIONS
* AND DATA',10X,' LESSON 3 ASSIGNMENT STATEMENTS AND BUILT IN FUNC
* TIONS',10X,' LESSON 4 BRANCHING',10X,' LESSON 5 LOOPING AND SUBSCR
* IPTED VARIABLES',10X,' LESSON 6 SUBROUTINES AND RECURSION',10X,'
* LESSON 7 SUMMARY OF BASIC STATEMENTS
* ** WHEN YOU ARE READY TO CONTINUE , TYPE IN GO AND HIT **
* THE CARRIAGE RETURN ** ** **
5 READ(5,101,END=5)CARD
103 WRITE(6,103)
*OVER**
6 READ(5,101,END=301)CARD
CALL CRUNCH(1,LENGTH)
DO 7 I=2,8
IF (CARDP(1).EQ.DIGIT(I)) GO TO 9
7 CONTINUE
301 WRITE(6,104)
104 *QUESTION AND REPLY AGAIN ** ** **
GO TO 6
9 LSNUM=1-1
GO TO (11,12,13,14,15,16,17),LSNUM
11 CALL LSON1
GO TO 25
12 CALL ALOAD(LES2,N1)
GO TO 25
13 CALL ALOAD(LES3,N1)
GO TO 25
14 CALL ALOAD(LES4,N1)
GO TO 25
15 CALL ALOAD(LES5,N1)
GO TO 25
16 CALL ALOAD(LES6,N1)
GO TO 25
17 CALL ALOAD(LES7,N1)
GO TO 25
C
C INSTRUCTION PHASE ** NEW USER
C
20 IF (CARDP(1).EQ.ALPHA(25))GO TO 21
302 WRITE(6,104)
GO TO 4
21 WRITE(6,106)
106 FORMAT(10,10X,'CAIBASIC IS A PROGRAM TO TEACH YOU THE
**
CA100660
CA100670
CA100680
CA100700
CA100730
CA100740
CA100750
CA100760
CA100780
CA100790
CA100800
CA100820
CA100830
CA100840
CA100850
CA100860
CA100870
CA100900
CA100910
CA100930
CA100950
CA100970
CA100990
CA101010
CA101030
CA101040
CA101060
CA101070
CA101080
CA101090
CA101100
CA101110

```

```

*FUNDAMENTALS OF A PROGRAMMING LANGUAGE. THE LANGUAGE TO BE LEARNED. CAIO1120
*// IS BASIC ; A SIMPLE LANGUAGE FOR THE USER WHO HAS LITTLE.// KNO CAIO1130
*WLEDGE OF COMPUTERS AND WHOSE PRIMARY INTEREST IS IN OBTAINING.//
*RESULTS.//ICX, THE SIMPLICITY OF THE BASIC LANGUAGE AND ITS RANGE. CAIO1150
*// OF CAPABILITY, IT SHOULD ALLOW YOU TO LEARN THE LANGUAGE AND.// CAIO1160
*WRITES PROGRAMS IN A MINIMAL AMOUNT OF TIME.//5X, THE REFERENCE T
*EXITS RECOMMENDED FOR CAIBASIC ARE ://5X, 1. BASIC IN I-147 )//5X
*L, TN # C211-12 APRIL 1971.// ( FREE UPON REQUEST R.L.
*// 2. INTRODUCTION TO COMPUTING THROUGH THE BASIC LANGUAGE PROGRAMMING.
*// NOLAN.// (BOOKSTORE / MAIN LIBRARY) //5X, 3. BASIC PROGRAMMING.
*// V.C. HARE.// (COMPUTER LIBRARY ) //***# AFTER YOU HAV CAIO1170
*// E FINISHED READING AN INPUT, TYPE IN GO AND HIT THE RETURN KEY.// CAIO1180
*// AND THE PROGRAM WILL CONTINUE **//) CAIO1190
*// READ(5,101,END=307)CARD CAIO1200
307 WRITE(6,107) CAIO1210
107 FORMAT(0,10X, DURING YOUR TERMINAL SESSION YOU WILL BE LEARNING. CAIO1220
*// THE STRUCTURE OF THE LANGUAGE BASIC. THE INSTRUCTION SET.// CAIO1230
*CONTAINS 7 LESSONS AND YOU MAY PROCEED THROUGH THE LESSONS AT.// CAIO1240
*YOUR OWN SPEED.//) CAIO1250
*// READ(5,101,END=308)CARD CAIO1260
308 WRITE(5,101,END=308)CARD CAIO1270
108 WRITE(6,108) CAIO1280
*// IN EACH LESSON YOU WILL BE GIVEN INSTRUCTION.// CAIO1290
*SEQUENCES AND THEN YOU WILL BE ASKED QUESTIONS TO SEE IF YOU.// CAIO1300
*UNDERSTOOD THE INSTRUCTIONS. THE QUESTIONS WILL BE OF VARIOUS.// CAIO1310
*// TYPES: MULTIPLE CHOICE, TRUE, FALSE, ACTUAL PROGRAM STATEMENTS, ETC.// CAIO1320
*// YOU WILL BE PROMPTED FOR YOUR ANSWER, AND WHEN READY TYPE IN.//
*// YOUR RESPONSE AND HIT THE RETURN KEY.//10X, IF YOU KEEP THE TELETYPE CAIO1340
*// TYPE OUTPUT FROM YOUR TERMINAL SESSION.// YOU WILL HAVE A READY REF CAIO1350
*// REFERENCE FOR FUTURE USE.//) CAIO1360
*// READ(5,101,END=309)CARD CAIO1370
309 LSNUM=1 CAIO1380
24 IF (LSNUM.GT.7) GO TO 40
25 GO TO (11,12,13,14,15,16,17),LSNUM
109 WRITE(6,109)
*// DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION CAIO1410
26 ?// (REPLY : YES OR NO )//) CAIO1420
*// READ(5,101,END=310)CARD CAIO1430
CALL CRUNCH(1,LENGTH) CAIO1440
IF (CARDP(1).EQ.ALPHA(25)) GO TO 40 CAIO1450
IF (CARDP(1).EQ.ALPHA(14)) GO TO 30 CAIO1460
310 WRITE(6,104) CAIO1470
GO TO 26 CAIO1480
OLD USERS PICK NEXT LESSON ; NEW USERS GET NEXT LESSON IN SEQUENCE CAIO1490
30 WRITE(6,209) CAIO1500
209 FORMAT(0,10X, IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME.// CAIO1510
*// THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION.// CAIO1520
*// WILL CONTINUE.//) CAIO1540

```

CAI01550
CAI01560
CAI01570
CAI01580

CAIO1590
CAIO1630
CAIO1610
CAIO1620
CAIO1630
CAIO1640
CAIO1650

CAIO1670
CAIO1680
CAIO1690
CAIO1700
CAIO1710
CAIO1720
CAIO1730
CAIO1740
CAIO1750
CAIO1760
CAIO1770

TESS00010
TESS00020
TESS00030
TESS00040
TESS00050
TESS00060
TESS00070
TESS00080
TESS00090
TESS00100
TESS00110
TESS00120
TESS00130
TESS00140
TESS00150
TESS00160
TESS00170
TESS00180
TESS00190

```
32 READ(5,101,END=33)CARD
   CALL CRUNCH(ILNGTH)
   IF(CARDP(1).EQ.ALPHA(14)) GO TO 34
   IF(CARDP(1).NE.ALPHA(25)) GO TO 33
```

ENTER EXECUTION PHASE

```

CALL TEST1
GO TO 25
33 WRITE(6,104)
GO TO 32
34 IF(OLD.EQ.1) GO TO 5
WRITE(6,110)
110 FORMAT('0','1GX',' DO YOU WANT TO REV
*/ REPLY : YES OR NO.//')
27 READ(5,101,END=311)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ALPHA(25)) GO TO 5
IF ((CARDP(1)).EQ.ALPHA(14)) GO TO 31
311 WRITE(6,104)
GO TO 27
31 LSNUM=LSNUM+1
GO TO 24
USER INSTRUCTION SESSION COMPLETED.
40 CONTINUE
STOP
END

```

SUBROUTINE TEST1
THIS ROUTINE ALLOWS EXECUTION AND EDITING OF BASIC PROGRAMS
SUBROUTINES CALLED FROM TEST1 ARE:

SYMBOL TABLE : (LOCAL SYMBOLS AND VARIABLES)

```

NEW          SET TO 1 WHEN USER HAS EXECUTED FIRST PROGRAM
IEND        SET TO 1 WHEN END STATEMENT READ
IEXERR      SET TO 1 WHEN AN EXECUTION/GLOBAL ERROR OCCURS
IERRKS      SET TO 1 WHEN A SYNTAX ERROR OCCURS IN INPUT
INTERP      SET TO 1 WHEN ONLY INTERPRETING
INSTMT      SET OF PROGRAM STATEMENTS INPUT
SF          SOURCE FILE VECTOR HOLDING PROGRAM STATEMENTS
NREF        PROGRAM STATEMENT REFERENCE NUMBER

```



```

CALL INITIAL
NEW=1
DEBUG=C.0
NSTMT=0
IEND=0
INBIG=1
IARRAY=808
NFOR=0
INTERP=0
IEXERR=0
5 WRITE(6,100)
100 FORMAT(10,5X,' INPUT BASIC PROGRAM NOW (ONE LINE AT A TIME )'//)
10 READ(5,101,END=5,CARD
101 FORMAT(80A1)
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 10
C FIND ' END ' STATEMENT
C
DO 699 I=1,80
IF(CARDP(I).NE.ALPHA(5).OR.CARDP(I+1).NE.ALPHA(14))GO TO 699
IF(CARDP(I+2).EQ.ALPHA(4).AND.CARDP(I+3).EQ.BLANK)GO TO 701
699 CONTINUE
700 CALL CCMPLR(NFOR,IARRAY,ILNGTH,INBIG)
C IF NO ERRORS , FILE INPUT STATEMENT
C
IF(NERRS.EQ.0) GO TO 9
IF(IEXERR.EQ.1)GO TO 9
NERRS=0
GO TO 10
C PUT IN END STATEMENT
C
701 NSTMT=NSTMT+1
IEND=1
DO 702 I=1,80
702 SFILF(NSTMT,I)=CARD(I)
9 GO TO 700
IF(NSTMT.GT.99) GO TO 511
IF(IEND.EQ.1) GO TO 12
NSTMT=NSTMT+1
DO 11 J=1,80
11 SFILF(NSTMT,J)=CARD(J)
GO TO 10
511 WRITE(6,512)
512 FORMAT(10,' YOU HAVE REACHED THE MAXIMUM PROGRAM SIZE ALLOWED'//
*UNDER CAI-BASIC . YOU WILL HAVE TO MODIFY YOUR PROGRAM TO

```



```

25 IF(CARDP(1).EQ.ALPHA(14))GO TO 26
326 WRITE(6,106)
106 * )
GO TO 20
26 WRITE(6,107)
107 * : YES OR NO : )
29 READ(5,101,END=330) CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ALPHA(14))GO TO 30
330 WRITE(6,106)
30 GO TO 29
RETURN
C C C
EDIT ROUTINE.
31 WRITE(6,108)
108 * )
108 * IF YOU WANT TO CORRECT YOUR PROGRAM NOW THEN REPLY : YES AND : )
* YOU WILL ENTER THE EDIT MODE TO CORRECT YCUR ERRORS
* OTHERWISE REPLY : NO : )
208 READ(5,101,END=343)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).NE.ALPHA(25)) GO TO 43
33 IF(OLD.EQ.1)GO TO 321
110 WRITE(6,110)
110 * )
* BY USING THE REFERENCE NUMBERS LISTED TO THE LEFT OF YOUR BASIC : )
* OF THE PROGRAM STATEMENT AT A TIME. IN ALL EDITING, CR CORRECT ONE LINE : )
* INPUT THE PROGRAM STATEMENT REFERENCE NUMBER AND HIT THE CARRIAGE : )
* RETURN. THE SECOND STEP DEPENDS ON WHAT EDITING YOU DO : )
* 1. DELETE : )
* THE BASIC STATEMENT REFERENCED IS DELETED BY TYPING THE : )
* LETTERS : )
* 2. CORRECT : )
* TO CORRECT THE BASIC STATEMENT REFERENCED TYPE IN THE COMPLETE : )
* CORRECT BASIC STATEMENT : )
* 3. ADD : )
* A BASIC STATEMENT IS ADDED AFTER THE BASIC STATEMENT : )
* REFERENCED BY TYPING THE LETTERS ADD1 FOLLOWED BY THE BASIC : )
* STATEMENT. ALL BLANKS FOLLOWING THE LETTERS ADD1 WILL BE : )
* INCLUDED IN THE BASIC STATEMENT. TO PLACE A STATEMENT BEFORE : )
* THE FIRST STATEMENT IN THE PROGRAM, USE THE REFERENCE NUMBER : )
* : )
OLD=1

```

TES01640
 TES01650
 TES01660
 TES01670
 TES01680
 TES01690
 TES01710
 TES01720
 TES01730
 TES01740
 TES01750
 TES01760
 TES01770
 TES01780
 TES01790
 TES01800
 TES01810
 TES01820
 TES01830
 TES01840
 TES01850
 TES01860
 TES01870
 TES01880
 TES01890
 TES01900
 TES01910
 TES01920
 TES01930
 TES01940
 TES01950
 TES01960
 TES01970
 TES01980
 TES01990
 TES02000
 TES02010
 TES02020
 TES02030
 TES02040
 TES02050
 TES02060
 TES02070
 TES02080
 TES02090
 TES02100
 TES02110


```

C
C
C      CORRECT CURRENT LINE
      DEBUG=C.C
      INTERP=1
      IEXERR=0
      DO 370 I=1,80
      IF(CARDP(I).EQ.ALPHA(14).AND.CARDP(I+3).EQ.ALPHA(20)) GO TO 39
      CCNTINUE
370
C
C      IF NO ERRORS , FILE INPUT STATEMENT
      CALL CCPLR(NFOR,IARRAY,ILNGTH,INBIG)
      IF(NERRS.EQ.0) GO TO 39
      NERRS=0
      GO TO 34
39  DO 40 J=1,80
40  SFILE(NREF,J)=CARD(J)
      GO TO 47
43  IF(CARDP(1).EQ.ALPHA(14)) GO TO 44
343 WRITE(6,106)
44  WRITE(6,113)
113  FORMAT(0,'10X',' IF YOU DESIRE TO HAVE THE PROGRAM RUN WITH
      *THE DEBUG FEATURE REPLY: YES ; OTHERWISE REPLY: NO AND THIS
      *PROGRAM WILL BE LOST .')
      CALL CRUNCH(ILNGTH)
45  READ(5,101,END=346)CARD
      IF(CARDP(1).NE.ALPHA(25)) GO TO 46
      EXECUTE WITH DEBUG FEATURE ADDED
C
C
C      CALL INITIAL
      DEBUG=1.0
      GO TO 136
46  IF(CARDP(1).EQ.ALPHA(14)) GO TO 26
346 WRITE(6,106)
      GO TO 45
47  WRITE(6,114)
114  FORMAT(0,'10X',' MORE CORRECTIONS TO BE MADE ?? REPLY: YES
      *OR REPLY: NO AND THE EDITED PROGRAM WILL BE EXECUTED .')
48  READ(5,101,END=348)CARD
      CALL CRUNCH(ILNGTH)
      IF(CARDP(1).EQ.ALPHA(25)) GO TO 321
      IF(CARDP(1).EQ.ALPHA(14)) GO TO 314
348 WRITE(6,106)
      GO TO 48
C

```

```

TES02600
TES02610
TES02620
TES02630
TES02640
TES02650
TES02660
TES02670
TES02680
TES02690
TES02700
TES02710
TES02720
TES02730
TES02740
TES02750
TES02760
TES02770
TES02780
TES02790
TES02800
TES02810
TES02820
TES02830
TES02840
TES02850
TES02860
TES02870
TES02880
TES02890
TES02900
TES02910
TES02920
TES02930
TES02940
TES02950
TES02960
TES02970
TES02980
TES02990
TES03000
TES03010
TES03020
TES03030
TES03040
TES03050
TES03060
TES03070

```

```

C      C      DELETE STATEMENT
50      NSTMT=NSTMT-1
      DO 52 I=NREF,NSTMT
      DO 52 J=1,80
      SFILE(I,J)=SFILE(I+1,J)
      GO TO 47
C      C      ADD A BASIC STATEMENT AFTER CURRENT LINE
55      IF(NSTMT.GT.99) GO TO 65
      NSTMT=NSTMT+1
      ITEMP=NSTMT
      DEBUG=0.0
      INTERP=1
      IEXERR=0
C      C      FIND BEGINNING OF STATEMENT
      DO 56 K=1,80
      IF(CARD(K).EQ.BLANK) GO TO 56
      IF(CARD(K).NE.ALPHA(1).OR.CARD(K+2).NE.ALPHA(4)) GO TO 56
      IF(CARD(K+3).EQ.DIGIT(2)) GO TO 60
      CONTINUE
56      WRITE(6,106)
      GO TO 37
      IBEG=K+3
      DO 60 L=1,80
      IBEG=IBEG+1
      IF(IBEG.GT.80) GO TO 601
      CARD(L)=CARD(IBEG)
      CONTINUE
600      CALL CRUNCH(ILNGTH)
601      IF(CARDP(1).EQ.ASTRK) GO TO 36
      DO 603 I=1,80
      IF(CARDP(I).EQ.ALPHA(14).AND.CARDP(I+3).EQ.ALPHA(20)) GO TO 602
      CCNTINUE
603      CALL COMPLR(INFOR,IARRAY,ILNGTH,INBIG)
C      C      IF NO ERRORS THEN FILE STATEMENT
C      C      IF(NERRS.EQ.0) GO TO 602
      NERRS=C
      GO TO 36
C      C      ADD STATEMENT TO PROGRAM FILE. MOVE STATEMENTS UP AND DOWN
      602 IADD=NREF+1

```

```

TES03080
TES03090
TES03100
TES03110
TES03120
TES03130
TES03140
TES03150
TES03160
TES03170
TES03180
TES03190
TES03200
TES03210
TES03220
TES03230
TES03240
TES03250
TES03260
TES03270
TES03280
TES03290
TES03300
TES03310
TES03320
TES03330
TES03340
TES03350
TES03360
TES03370
TES03380
TES03390
TES03400
TES03410
TES03420
TES03430
TES03440
TES03450
TES03460
TES03470
TES03480
TES03490
TES03500
TES03510
TES03520
TES03530
TES03540
TES03550

```

```

61 DO 62 I=1,80
62 SFILE(I,TEMP,I)=SFILE(ITEMP-1,I)
   ITEMP=ITEMP-1
   IF(ITEMP.GT.IADD) GO TO 61
63 DO 63 J=1,80
   SFILE(IADD,J)=CARD(J)
64 GO TO 47
65 WRITE(6,512)
   GC TO 47
END

```

TES03560
 TES03570
 TES03580
 TES03590
 TES03600
 TES03610
 TES03620
 TES03630
 TES03640
 TES03650

SUBROUTINE CRUNCH(ILNGTH)

GIVEN THE VECTOR CARD ; THIS SUBROUTINE REMOVES ALL BLANKS IN CARD
 AND RETURNS THE BLANK-LESS VERSION IN CARDP.
 THIS ROUTINE ALSO CHECKS FOR TYPING ERRORS AND CHECKS IF THE USER
 IS COMPLETED WITH HIS SESSION OR IN AN EDIT MODE

CRUC0010
 CRUC0020
 CRUC0030
 CRUC0040
 CRUC0050
 CRUC0060
 CRUC0070
 CRUC0080
 CRUC0090
 CRUC0100
 CRUC0110
 CRUC0120
 CRUC0130
 CRUC0140
 CRUC0150
 CRUC0160
 CRUC0170
 CRUC0180
 CRUC0190
 CRUC0200
 CRUC0210
 CRUC0220
 CRUC0230
 CRUC0240
 CRUC0250
 CRUC0260
 CRUC0270
 CRUC0280
 CRUC0290
 CRUC0300
 CRUC0310
 CRUC0320
 CRUC0330
 CRUC0340
 CRUC0350
 CRUC0360

```

COMMON
STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRIIB(10), LIST(100), IIST(100), QUOTE,
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, COMMA,
EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH,
PARLFT, ASTRSK, BLANK
COMMON INTERP, IEXERR
DO 19 I=1,80
19 CARDP(I)=BLANK

```

19 CARDP(I)=BLANK

REMOVE BLANKS

```

ILNGTH=0
DO 20 I=1,80
IF (CARD(I).EQ.BLANK) GO TO 20
ILNGTH=ILNGTH+1
CARDP(ILNGTH)=CARD(I)
20 CONTINUE

```

CHECK FOR BLANK INPUT

```

IF(ILNGTH.EQ.0) CARDP(1)=ASTRSK
DO 25 I=1,80

```

CHECK FOR TYPING ERROR(\$\$\$\$)

```

IF(CARDP(1).NE.DOLSGN.OR.CARDP(I+1).NE.DOLSGN) GO TO 26
IF(CARDP(I+2).EQ.DOLSGN.AND.CARDP(I+3).EQ.DOLSGN) CARDP(1)=ASTRSK

```


C	IAPTR=C	INI00290
C	INPTR=0	INI00300
	INITIALIZE PRIORITY TABLE	INI00310
	IPRITB(1)=2	INI00320
	IPRITB(2)=2	INI00330
	IPRITB(3)=3	INI00340
	IPRITB(4)=3	INI00350
	IPRITB(5)=4	INI00360
	IPRITB(6)=5	INI00370
C	SET UP VOCABULARY	INI00380
C	LOAD ALPHABET	INI00390
C		INI00400
	DO 10 I=1,26	INI00410
10	ALPHA(I)=ATEMP(I)	INI00420
C	LOAD DIGITS	INI00430
C		INI00440
C	DO 20 I=1,10	INI00450
20	DIGIT(I)=DIGTMP(I)	INI00460
C	LOAD DIGITS INTO ALPHA	INI00470
C		INI00480
C	LOC=26	INI00490
	DO 12 I=1,10	INI00500
	LCC=LOC+1	INI00510
12	ALPHA(LOC)=DIGIT(I)	INI00520
C	LOAD SPECIAL CHAR INTO ALPHA	INI00530
C		INI00540
C	DO 16 I=1,12	INI00550
	LCC=LOC+1	INI00560
16	ALPHA(LOC)=CHARM(I)	INI00570
C	INITIALIZE ARRAY STORAGE POINTERS	INI00580
C		INI00590
C	DO 50 I=287,384,4	INI00600
50	PRT(I)=0.0	INI00610
C	LOAD SPECIAL CHARACTERS	INI00620
C		INI00630
C	ASTRSK=CHARM(1)	INI00640
	BLANK =CHARM(2)	INI00650
	COMMA =CHARM(3)	INI00680
	DECIMAL=CHARM(4)	INI00690
	EQUALS=CHARM(5)	INI00700
	PARRT =CHARM(6)	INI00710
		INI00720
		INI00730
		INI00740
		INI00750
		INI00760


```

*BY AN END STATEMENT. THE WORDS : REM, READ, LET, PRINT, DATA,
*//, AND END ARE KEY WORDS THAT MAKE UP A BASIC STATEMENT. //)
READ(5,101,END=304)CARD
304 WRITE(6,104)
104 FORMAT(0,10X, MOST OF THE KEY WORDS USED IN THE BASIC
* STATEMENTS ARE SELF-EXPLANATORY:
* REM ALLOWS REMARKS/COMMENTS
* READ M,G ASSIGNS NUMBERS IN THE DATA STATEMENT TO THE
* LET ASSIGNS THE RESULT OF M DIVIDED BY G INTO VARIABLE T
* PRINT STRING. CAUSES THE STRIKING IN SINGLE QUOTES TO BE
* PRINT M,G,T PRINTS THE CURRENT VALUE OF THE VARIABLES M,G,T.//5X,
* END TELLS THE COMPUTER THAT THE INPUT PROGRAM IS TO BE
* EXECUTED.//)
READ(5,101,END=305)CARD
305 WRITE(6,105)
105 FORMAT(0,10X, IF AT THIS POINT YOU WOULD LIKE TO RUN THE
* SAMPLE PROGRAM TO GAIN SOME CONFIDENCE IN THE COMPUTER AND ITS
* ABILITY TO PROVIDE SPEEDY RESULTS, THEN REPLY : YES ; OTHER-//
* WISE REPLY : NO AND THE INSTRUCTION WILL CONTINUE. //)
10 READ(5,101,END=306)CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 10
IF(CARDP(1).EQ.ALPHA(14)) GO TO 25
IF(CARDP(1).EQ.ALPHA(25)) GO TO 15
306 WRITE(6,106)
106 FORMAT(0,10X, *** YOUR REPLY WAS TYPED INCORRECTLY ; CHECK THE
* QUESTION AND REPLY AGAIN ***//)
GO TO 10

EXECUTE SAMPLE PROGRAM

15 WRITE(6,107)
107 FORMAT(0,10X, TO EXECUTE THE SAMPLE PROGRAM TYPE IN THE BASIC
* STATEMENTS AS THEY APPEAR, AND DON'T WORRY ABOUT SPACING. THE
* COMPUTER ANALIZES EACH BASIC STATEMENT AS IT IS INPUT ; AND
* IF THERE ARE NO ERRORS, IT UNLOCKS THE KEYBOARD AND WAITS FOR
* YOUR NEXT INPUT. WHEN THE END STATEMENT IS INPUT THE PROGRAM
* IS EXECUTED. //)
READ(5,101,END=308)CARD
308 WRITE(6,108)
108 FORMAT(0,10X, HOWEVER, IF YOU MAKE AN ERROR THE COMPUTER WILL
* TELL YOU THE ERROR, AND EXPECT A CORRECTION. DON'T WORRY ABOUT
* THE ERROR, CHECK YOUR INPUT AGAINST THE SAMPLE PROGRAM, AND
* INPUT THE CORRECT STATEMENT. IF YOU MAKE A TYPING MISTAKE
* INPUT FOUR ASTERIX(****), HIT RETURN, AND THEN INPUT THE
* CORRECT STATEMENT. //10X, YOU MAY USE ANY PAIR OF INTEGER OR DECIMAL

```

LES00380
 LES00390
 LES00400
 LES00410
 LES00420

LES00500
 LES00510
 LES00520
 LES00530
 LES00540
 LES00550
 LES00560
 LES00570
 LES00580
 LES00590
 LES00600
 LES00610
 LES00620
 LES00630
 LES00640
 LES00650
 LES00660
 LES00670
 LES00680
 LES00690
 LES00700
 LES00710
 LES00720
 LES00730
 LES00740
 LES00750
 LES00760
 LES00770
 LES00780
 LES00790
 LES00800
 LES00810
 LES00820

```

MAL NUMBERS'// FOR INPUT DATA YOU MAY OMIT THE REM STATEMENTS AND LES00830
* THE PRINT 'MILES TRAVELED'// ETC'// STATEMENT IF YOU DONT WANT LES00840
* T TO TYPE A LOT OF STATEMENTS'// LES00850
CALL TEST1 LES00860
25 WRITE(6,111) 5X,'B. PROGRAM FORMAT'// 5X,'A BASIC PROGRAM CONSISTS LES00870
111 FORMAT(10,111) 5X,'A BASIC PROGRAM CONSISTS LES00880
* OF A SEQUENCE OF BASIC STATEMENTS'// STATEMENT PER INPUT LINE LES00890
* FOLLOWED BY AN END STATEMENT'// BECAUSE NO BASIC STATEMENT LES00900
* MAY BE LONGER THAN ONE INPUT LINE (80 SPACES)'// THERE IS NO PR LES00910
* DIVISION FOR CONTINUING STATEMENTS FROM ONE LINE TO THE NEXT. HLES00920
* HOWEVER YOU MAY SPACE THE INPUT LINE'// AS DESIRED FOR READABIL LES00930
* TY SINCE THE COMPUTER IGNORES BLANKS IN BASIC'// LES00940
READ(5,101)END=309)CARD LES00950
309 WRITE(6,112) LES00960
112 FORMAT(10,10X,'1. STATEMENT NUMBERS LES00970
* EACH BASIC STATEMENT MAY HAVE AN OPTIONAL LES00980
* STATEMENT NUMBER PRECEDING IT FOR IDENTIFICATION PURPOSES' LES00990
* THIS STATEMENT NUMBER MUST BE AN INTEGER BETWEEN 1 -> 9999 LES01000
* 5X,' FOR EXAMPLE : LES01010
* 10X,' 2. STATEMENT (REM, READ, LET, ETC,)'// ARE SPECIAL TERMINAL SYM LES01020
* BASIC STATEMENTS RECOGNIZED BY THE COMPUTER'// AND FOR THIS REASON TH LES01030
* BCLS THAT ARE SPELLED CORRECTLY AND ONLY'// USED IN BASIC STATEMENTS LES01040
* THEY MUST BE SPELLED CORRECTLY AND ONLY'// USED IN BASIC STATEMENTS LES01050
* /10X,' THE END STATEMENT INDICATES THAT THE INPUT PROGRAM IS'// LES01060
* COMPLETED AND THAT PROGRAM EXECUTION IS TO BEGIN'// THE'// LES01070
* STATEMENT IS ALWAYS THE LAST STATEMENT IN A PROGRAM'// LES01080
READ(5,101)END=313)CARD LES01090
313 WRITE(6,113) LES01100
113 FORMAT(10,113) YOU WILL NOW BE ASKED A FEW SIMPLE QUESTIONS ABOUT'// LES01110
* WHAT YOU HAVE JUST LEARNED'// LES01120
WRITE(6,114) 10X,' 1. ) IS THIS A LEGAL BASIC PROGRAM(REPLY : YES OR LES01130
FORMAT(10,10X,' REM ONE LINE DO NOTHING PROGRAM'// END'//) LES01140
114 *NO)??//5X,' REM ONE LINE DO NOTHING PROGRAM'// END'//) LES01150
30 CALL CRUNCH(1,LENGTH) LES01160
IF(CARDP(1).EQ.ASTRSK) GO TO 30 LES01170
IF(CARDP(1).EQ.ALPHA(14).OR.CARDP(1).EQ.ALPHA(25)) GO TO 31 LES01180
315 WRITE(6,106) LES01190
GO TO 30 LES01200
31 WRITE(6,115) LES01210
115 FORMAT(10,115) YES, THE SIMPLEST BASIC PROGRAM CONSISTS OF JUST AN LES01220
* END STATEMENT'// LES01230
WRITE(6,116) LES01240
FORMAT(10,10X,' 2. ) WHICH OF THE FOLLOWING BASIC STATEMENTS IS'// LES01250
116 *IN THE PROPER FORMAT ://15X,' A. ) 15 LET T=M/G//15X,' B. ) 15 LEFT= LES01260
* M/G//15X,' C. ) 15 L E T = M / G //10X,' REPLY A , B , C OR LES01270
* ALL'// LES01280
32 READ(5,101)END=317)CARD LES01290

```

[illegible]

```

320 READ(5,101,END=320)CARD
120 WRITE(6,120) 5X, 'E. NUMBERS
*10X, 'NUMBERS MAY BE EXPRESSED AS INTEGERS OR AS REALS, AND A
*NUMBER WHETHER INTEGER OR REAL IS LIMITED TO 9 DIGITS, NOT INCLUD
*ING DECIMAL POINT. A NUMBER IS ASSUMED POSITIVE UNLESS IT
*IS PRECEDED BY A - SIGN.
*10X, '1. INTEGERS
*RS ARE NUMBERS WITH NO FRACTIONAL PART, IE. 3, 15
*10X, '2. REALS
*REAL (FIXED-POINT) NUMBERS HAVE A DECIMAL POINT AND A FRACTIONAL
*PART, IE. 3.0, 15.31, 729.1, 0.0
READ(5,101,END=321)CARD
121 WRITE(6,121) 5X, 'F. EXPRESSIONS
*AN EXPRESSION MAY BE A SINGLE CONSTANT OR VARIABLE, OR AN
*ARITHMETIC EXPRESSION. ARITHMETIC EXPRESSIONS ARE FORMED BY
*USING OPERATORS AND PARENTHESES. //10X, '1. OPERATORS
*10X, 'NUMBERS AND SIMPLE VARIABLES MAY BE COMBINED INTO ARITHMETIC
*EXPRESSIONS BY USING ONE OR MORE OF THE ARITHMETIC OPERATORS:
*//4X, 'EXPONENTIATION, /5X, * MULTIPLICATION, /5X, / DIVISION
*ON /5X, + ADDITION, /5X, - SUBTRACTION
*IN WRITING AN EXPRESSION, EACH ARITHMETIC OPERATION MUST BE SHOWN
*// NO TWO OPERATIONS MAY BE ADJACENT, NOR MAY TWO NUMBERS OR
*VARIABLES BE ADJACENT IN AN EXPRESSION. // FOR EXAMPLE: 2+2, A/B
*B+C, X*Z, AND Y-3.0 ARE LEGAL EXPRESSIONS.
READ(5,101,END=322)CARD
122 WRITE(6,122) 5X, 'G. PARENTHESES
* TO CONTROL THE ORDER IN WHICH AN ARITHMETIC EXPRESSION IS EVALU
*ATED, // THE NORMAL HIERARCHY OF OPERATORS IS: /5X, 1. *, /5X, 2.
* AND /5X, 3. + AND -
* PARENTHESES ARE USED, THE EXPRESSION WITHIN A PARENTHESES PAIR
*// IS EVALUATED FIRST. * (4 + 3), (5*(4+3)) ARE EQUIVALENT
* OCCUR IN PAIRS, IE. THE VALUE 35
* EXPRESSIONS ARE EVALUATED ACCORDING TO THE OPERATORS HIERARCHY
* OPERATIONS ARE EVALUATED ACCORDING TO THE SAME HIERARCHY
*// IF TWO OPERATORS OCCUR IN AN EXPRESSION,
* EVALUATION IS FROM LEFT TO RIGHT. // FOR EXAMPLE: 2*4 + 6/3
* IN THIS EXPRESSION 2*4 WILL BE EVALUATED FIRST, THEN 6/3
* AND THEN THE TWO RESULTS WILL BE ADDED.
READ(5,101,END=323)CARD
123 WRITE(6,123) 5X, 'H. HAS THE VALUE (REPLY WITH VALUE)
*10X, '1.) 4 + 6 / 2 HAS THE VALUE (REPLY WITH VALUE)
34 READ(5,101,END=324)CARD

```

```

35 CALL CRUNCH(ILNGTH)
36 IF(CARDP(1).EQ.ASTRSK) GO TO 34
37 IF(CARDP(1).EQ.DIGIT(8)) GO TO 37
38 DO 35 I=1,10
39 IF(CARDP(1).EQ.DIGIT(I)) GO TO 36
40 CCNTINUE
41 WRITE(6,106)
42 GO TO 34
43 FORMAT('0', ' YOUR ANSWER IS INCORRECT . THE EXPRESSION IS EVALUATE
44 *D AS FOLLOWS :')
45 WRITE(6,124)
46 WRITE(6,125)
47 FORMAT('0', ' 10X, ' 4+6/2 -> 4+3 -> 7 ')
48 WRITE(6,126)
49 FORMAT('0', ' 10X, ' 2.) (4+6)/2 HAS THE VALUE (REPLY WITH VALUE)'//)
50 READ(5,101,END=327)CARD
51 CALL CRUNCH(ILNGTH)
52 IF(CARDP(1).EQ.ASTRSK) GO TO 38
53 IF(CARDP(1).EQ.DIGIT(6)) GO TO 41
54 DO 39 I=1,10
55 IF(CARDP(1).EQ.DIGIT(I)) GO TO 40
56 CCNTINUE
57 WRITE(6,106)
58 GO TO 38
59 WRITE(6,124)
60 WRITE(6,127)
61 FORMAT('0', ' 10X, ' (4+6)/2 -> 10/2 -> 5')
62 WRITE(6,128)
63 FORMAT('0', ' 10X, ' 3.) ((4+6)/2)**2 HAS THE VALUE (REPLY : VALUE)'//)
64 READ(5,101,END=46)CARD
65 CALL CRUNCH(ILNGTH)
66 IF(CARDP(1).EQ.ASTRSK) GO TO 42
67 IF(CARDP(1).EQ.DIGIT(3).AND.CARDP(2).EQ.DIGIT(6)) GO TO 48
68 DO 43 I=1,10
69 IF(CARDP(1).EQ.DIGIT(I)) GO TO 44
70 CCNTINUE
71 GO TO 46
72 CONTINUE
73 DO 45 J=1,10
74 IF(CARDP(2).EQ.DIGIT(J)) GO TO 47
75 CCNTINUE
76 WRITE(6,106)
77 GO TO 42
78 WRITE(6,124)
79 WRITE(6,129)
80 FORMAT('0', ' 10X, ' ((4+6)/2)**2 -> (10/2)**2 -> (5)**2 -> 25'//)
81 WRITE(6,130)

```

S02270
 S02280
 S02290
 S02300
 S02310
 S02320
 S02330
 S02340
 S02350
 S02360
 S02370
 S02380
 S02390
 S02400
 S02410
 S02420
 S02430
 S02440
 S02450
 S02460
 S02470
 S02480
 S02490
 S02500
 S02510
 S02520
 S02530
 S02540
 S02550
 S02560
 S02570
 S02580
 S02590
 S02600
 S02610
 S02620
 S02630
 S02640
 S02650
 S02660
 S02670
 S02680
 S02690
 S02700
 S02710
 S02720
 S02730
 S02740

```

130 FORMAT('O', 5X, ' F, SUMMARY
*// YOU WILL NOW BE GIVEN A SAMPLE BASIC PROGRAM AND ASKED TO FIND
*// THE MISTAKES IN IT, CONCERNING WHAT YOU HAVE LEARNED IN THIS
*// LESSON : //5X, ' 1, REM REVIEW PROGRAM //5X, ' 2, READ A1, B2, Z1, $
*Y//5X, ' 3, LET A1=82(A1+Z1)//6X, ' 4, LET B2=((A1*3)+Z1**2)//5X, ' 5
*//5X, ' 5, LET Z1=3.141592763//5X, ' 6, PRINT 1A, B2, Z1//5X, ' 7, DATA 5.0, 3.3, B
*//5X, ' 6, FINISH//. AFTER LOOKING AT THE SAMPLE PROGRAM, YOU WILL
*//5X, ' 7, ASKED QUESTIONS//. ABOUT EACH STATEMENT, //)
331 WRITE(6, 131)
131 FORMAT('O', 1CX, ' 1.) LINE 2 CONTAINS A READ STATEMENT FOLLOWED'//
50 READ(5, 101, END=332) CARD
CALL CRUNCH(1, ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 50
IF(CARDP(1).EQ.ALPHA(14)) GO TO 52
IF(CARDP(1).EQ.ALPHA(25)) GO TO 51
332 WRITE(6, 106)
51 GO TO 50
51 WRITE(6, 132)
132 FORMAT('O', 1CX, ' 2.) INPUT CORRECTION TO THE INCORRECT VARIABLE ONLY .')//
510 READ(5, 101, END=52) CARD
CALL CRUNCH(1, ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 510
IF(CARDP(1).EQ.ALPHA(25).AND.CARDP(2).EQ.ALPHA(46)) GO TO 53
52 WRITE(6, 133)
133 FORMAT('O', 1CX, ' 3.) THE ILLEGAL VARIABLE IS $Y. SIMPLE VARIABLES ARE
*//. A LETTER OR A LETTER FOLLOWED BY A SINGLE DIGIT; ALPHA VARIABLE
*//. ARE A LETTER FOLLOWED BY A DOLLAR SIGN, $, IE. Y$.')
53 WRITE(6, 144)
144 FORMAT('O', 1CX, ' 2.) LINE 3 CONTAINS A LET STATEMENT FOLLOWED'//
54 READ(5, 101, END=345) CARD
CALL CRUNCH(1, ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 54
IF(CARDP(1).EQ.ALPHA(14)) GO TO 58
IF(CARDP(1).EQ.ALPHA(25)) GO TO 55
345 WRITE(6, 106)
55 GO TO 54
55 WRITE(6, 134)
134 FORMAT('O', 1CX, ' 3.) INPUT CORRECTION TO ILLEGAL EXPRESSION ; EVERYTHING'//
56 READ(5, 101, END=58) CARD
CALL CRUNCH(1, ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 56
IF(CARDP(1).EQ.ALPHA(2).AND.CARDP(2).EQ.DIGIT(3)) GO TO 57
IF(CARDP(1).EQ.ASTRSK.OR.CARDP(4).NE.PARLFT) GO TO 58
57 IF(CARDP(3).NE.ASTRSK

```

```

IF(CARDP(5).NE.ALPHA(1).OR.CARDP(6).NE.DIGIT(2))GO TO 58
IF(CARDP(7).NE.PLUS.OR.CARDP(8).NE.ALPHA(26))GO TO 58
IF(CARDP(9).EQ.DIGIT(2).AND.CARDP(10).EQ.PARRT)GO TO 59
58 WRITE(6,135)
135 FORMAT(0, ' THE EXPRESSION SHOULD BE '/10X, ' A1=B2*(A1+Z1) ' /'
*EACH ARITHMETIC OPERATION MUST BE WRITTEN OUT ; A(B) IS NOT ASSUMED
*ED, /, TO BE A*(B) .',
59 WRITE(6,136)
136 FORMAT(0, ' 1CX, ' 3, ' LINE 4 IS SIMILAR TO LINE 3 . ANY ERRORS' /'
* REPLY YES OR NO) ?' /' )
60 READ(5,101,END=361)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 60
IF(CARDP(1).EQ.ALPHA(25))GO TO 64
IF(CARDP(1).EQ.ALPHA(14)) GO TO 65
361 WRITE(6,106)
GO TO 60
64 WRITE(6,137)
137 FORMAT(0, ' THE EXPRESSION IS CORRECT .')
138 WRITE(6,138)
138 FORMAT(0, ' 1CX, ' 4, ' LINE 5 CONTAINS A LET STATEMENT FOLLOWED' /'
* BY A NUMBER . ANY ERRORS (REPLY : YES OR NO) ?' /' )
70 READ(5,101,END=371)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 70
IF(CARDP(1).EQ.ALPHA(14).OR.CARDP(1).EQ.ALPHA(25)) GO TO 71
371 WRITE(6,106)
GO TO 70
71 WRITE(6,139)
139 FORMAT(0, ' THE NUMBER CONTAINS 10 DIGITS AND THE MAXIMUM ALLOWED
* / IS 9 DIGITS .')
140 WRITE(6,140)
140 FORMAT(0, ' 1CX, ' 5, ' LINE 6 CONTAINS A PRINT STATEMENT FOLLOWED'
* / BY A LIST OF VARIABLES . ANY ERRORS (REPLY : YES OR NO)?' /' )
72 READ(5,101,END=373)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 72
IF(CARDP(1).EQ.ALPHA(14)) GO TO 74
IF(CARDP(1).EQ.ALPHA(25)) GO TO 73
373 WRITE(6,106)
GO TO 72
73 WRITE(6,132)
132 FORMAT(0, ' 1CX, ' 6, ' LINE 7 CONTAINS A LET STATEMENT FOLLOWED'
* / BY A LIST OF VARIABLES . ANY ERRORS (REPLY : YES OR NO)?' /' )
74 WRITE(6,141)
141 FORMAT(0, ' THE ILLEGAL VARIABLE IS 1A . SIMPLE VARIABLES ARE A' /

```

LES03230
LES03240
LES03250
LES03260
LES03270
LES03280
LES03290
LES03300
LES03310
LES03320
LES03330
LES03340
LES03350

LES03380
LES03390
LES03510

LES03540
LES03550
LES03560
LES03570
LES03580
LES03590
LES03600
LES03610
LES03620
LES03630
LES03640
LES03650
LES03660
LES03670
LES03680
LES03690
LES03700
LES03710
LES03720
LES03730
LES03740
LES03750
LES03760
LES03770
LES03780
LES03790
LES03800
LES03810
LES03820

```

* LETTER OR A LETTER FOLLOWED BY A SINGLE DIGIT ,IE. A1 .)
75 WRITE(6,142)
142 FORMAT(0,10X,' THERE ARE NO ERRORS IN LINE 7. IS THE PROGRAM'/'
* READY TO EXECUTE (ASSUMING ABOVE ERRORS CCRRECTED)')/' (REPLY : YE
*S OR NO)')/'
750 READ(5,101,END=80)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 750
IF(CARDP(1).EQ.ALPHA(14).OR.CARDP(1).EQ.ALPHA(25)) GO TO 80
WRITE(6,106)
80 WRITE(6,143)
143 FORMAT(0,10X,' THE PROGRAM FORMAT REQUIRES THAT AN END STATEMENT
* BE THE LAST STATEMENT OF THE PROGRAM. THUS THIS SAMPLE PROGRA
*M'/' WOULD NOT EXECUTE .')/' THIS CONCLUDES THE REVIEW QUESTIONS'/'
* FROM LESSON 1 .')/'
CALL EXIT
END

```

CCC

```

LESSON 2 INTRODUCES THE REM , PRINT , READ , DATA , RESTORE ,
AND RESTORES STATEMENTS
COMMON
STACK(100), PROG(200), CARD(80), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRIB(10), LIST(100), ISTR(100),
PRIG(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PART, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLFT, ASTRSK, BLANK
PARTRP, IEXERR
COMMON LES2/LES2N1
REAL*8 LES2/LES2N1
CALL ALOAD(LES2,N1)
WRITE(6,100)

```

```

100 FORMAT(0,5X,' ** LESSON 2. **
* THIS INSTRUCTION SEQUENCE WILL INTRODUCE YOU TO THE BASIC LANGUAGE
* STATEMENTS REM , PRINT , READ , AND DATA. USING THESE STATEMENTS
* YOU WILL BE ABLE TO CONSTRUCT AND EXECUTE ELEMENTARY PROGRAMS
* AMS'/' 1X,' THE FORM FOR EACH BASIC STATEMENT WILL INCLUDE :')/'
* X, 'KEY WORD'/' << ELEMENTS OR LIST OF ELEMENTS SEPARATED BY COMMA
* S, >>
* THE CARET SYMBOLS '<< >>' DELINEATE THE LEGAL ITEMS THAT MAY'/'
* FOLLOW THE KEYWORD AND MAKE UP THE BASIC STATEMENT .')/'
* READ(5,101,END=302)CARD
101 FORMAT(80A1)
302 FORMAT(0,10X,' A. REM
* REMARKS INTO YOUR PROGRAM IS IDENTIFIED BY THE KEY WORD 'REM'/'
* REM IS A NON-EXECUTING STATEMENT WHICH MAY BE USED OPTIONALLY'/'

```

```

303 */ AT ANY PLACE IN YOUR PROGRAM TO INTRODUCE A PROGRAM NAME , TO'//LES000310
304 * EXPLAIN VARIABLES, TO DOCUMENT YOUR PROGRAM , ETC.'//)LES000320
305 READ(5,101,END=303)CARDLES000330
306 WRITE(6,103)LES000340
307 * REM << ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS >>LES000350
308 * THE REM << ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS >>LES000360
309 * ONLY FOR YGUR INFORMATION .15X, FOR EXAMPLE : '15X, AND IS'//LES000370
310 * AM TO COMPUTE INCOME TAX.'//)LES000380
311 READ(5,101,END=304)CARDLES000390
312 WRITE(6,104)LES000400
313 * THE PRINT STATEMENT IS THE METHOD OF WRITING OUT THE RESULTS OF'//5X,LES000410
314 * THE BASIC PROGRAM , TO DISPLAY VALUES OF VARIABLES , TO LABEL'//LES000420
315 * THE RESULTS , AND TO SKIP A LINE. THE FORM OF THE PRINTLES000430
316 * STATEMENT IS : '15X, PRINT << EXPRESSION OR 'STRING' , , , , EXPRELES000440
317 * SSION OR 'STRING' , , , , MULTIPLE ELEMENTS IN THE PRINT LIST ARELES000470
318 * SEPARATED BY COMMAS.'//)LES000480
319 READ(5,101,END=305)CARDLES000490
320 WRITE(6,105)LES000500
321 * CURRENT VALUE OF THE EXPRESSION >> WILL WRITE OUT THE'//LES000520
322 * IN LESSON THAT IS TO BE EVALUATED .15X, FOR EXAMPLE : '15X,LES000530
323 * PRINT 1,A,B1.5*2.'// ASSUMING THAT A=10.0 , B=13.3 WOULD PRINT :LES000540
324 * '15X, 1,101,END=306)CARDLES000550
325 READ(5,101,END=306)CARDLES000560
326 WRITE(6,106)LES000570
327 * ALPHA-NUMERIC CHARACTERS OF THE STRING WITHIN 'SINGLE' QUOTELES000580
328 * THIS FORM IS USED FOR LABELING THE COMPUTER OUTPUT.'//LES000600
329 * FCRL EXAMPLE : PRINT : '15X, THE ANSWER IS : 'LES000610
330 * PRODUCES THE RESULT : '15X, THE ANSWER IS : '15X, 3. PRINTLES000620
331 * ITSELF IS USED TO SKIP A LINE ON THE COMPUTER OUTPUT .'//)BYLES000640
332 READ(5,101,END=307)CARDLES000650
333 WRITE(6,107)LES000660
334 * EACH 15 COLUMNS WIDE . PRINT ZONES CAN BE SKIPPED BY PUTTING A'//LES000670
335 * BLANK IN THE PRINT LIST . FOR EXAMPLE : '15X, PRINT A,B ,X'//LES000680
336 * OUTPUTS THE THIRD ZONE AND PUTS X IN THE FIRST ZONE , B ,X'//LES000700
337 * SKIPS THE THIRD ZONE AND PUTS X IN THE FOURTH ZONE'//LES000710
338 * ZONES , BUT NUMERIC RESULTS ARE LEFT ADJUSTED OVER SEVERAL'//LES000720
339 * ITEMS . IF MORE THAN EIGHT ITEMS OCCUR IN THE PRINT LIST , THE'//LES000730
340 * WILL OVERFLOW AND BE PRINTED ON THE NEXT LINE .'//)LES000750
341 READ(5,101,END=308)CARDLES000760
342 WRITE(6,108)LES000770
343 * USING THE PRINT AND END STATEMENT YOU NOW HAVE THE'//LES000780
344 FORMAT('0',)

```



```

315 READ(5,101,END=315)CARD
115 WRITE(6,115)
*// D. DATA
*// THE DATA STATEMENT IS A LIST OF INPUT NUMBERS OR STRINGS THAT
*// WILL BE ASSIGNED TO VARIABLES IN A READ STATEMENT. THE FORM
*// OF THE DATA STATEMENT IS
*// DATA << NUMBER OR STRING >>
*// THE STRING OF ALPHA-NUMERIC CHARACTERS MUST BE ENCLOSED IN
*// SINGLE QUOTES
*// DATA STATEMENTS MAY BE PLACED ANYWHERE IN A PROGRAM, BUT THERE
*// IS AN UPPER LIMIT OF 500 NUMERIC AND 500 ALPHA-NUMERIC DATA
*// ELEMENTS FOR EACH PROGRAM
*//
316 READ(5,101,END=316)CARD
116 WRITE(6,116)
*// WHEN THE FIRST DATA STATEMENT IS INTERPRETED
*// BY THE CAIBASIC COMPILER A FIRST IN, FIRST OUT STACK IS FORMED
*// FOR THE NUMERIC AND ALPHA-NUMERIC DATA AS EACH NUMBER OR STRING
*// IN A DATA LIST IS INTERPRETED. IT IS PLACED ON THE BOTTOM OF
*// ITS RESPECTIVE DATA STACK. AS OTHER DATA STATEMENTS ARE LOCATED
*// IN THE PROGRAM ITS ELEMENTS ARE PLACED ON THE BOTTOM OF THE
*// PROPER STACK.
*// EXAMPLE 1. DATA 10.0,13.33, J.E. SMITH
*// PRODUCES 18.75X, 10.0, 10X, J.E. SMITH
*// ERIC 75X, 10.0, 10X, J.E. SMITH
*// 18.75X, 10.0, 10X, J.E. SMITH
*//
317 READ(5,101,END=317)CARD
117 WRITE(6,117)
*// DURING EXECUTION OF READ STATEMENTS AS THE
*// VARIABLES IN THE READ LIST ARE ASSIGNED VALUES FROM THE TOP OF
*// THE APPROPRIATE DATA STACK, THE DATA STACK IS DECREMENTED AND
*// THE NEXT ELEMENT THAT THE DATA FROM EXAMPLE 1. IS AVAILABLE, THE
*// VARIABLES IN THE READ LIST ARE ASSIGNED VALUES AS FOLLOWS
*// A <-- 13.33, 10X, 18.75X, 10.0, 10X, J.E. SMITH
*// THE RESULTING DATA STACKS ARE AS FOLLOWS
*// ALPHA-NUMERIC 75X, 10.0, 10X, J.E. SMITH
*// ERIC 75X, 10.0, 10X, J.E. SMITH
*// 18.75X, 10.0, 10X, J.E. SMITH
*//
318 READ(5,101,END=318)CARD
118 WRITE(6,118)
*// RESTORE STATEMENTS ARE USED TO RETURN THE NUMERIC
*// AND ALPHA-NUMERIC DATA STACKS TO THEIR ORIGINAL CONDITION SO
*// THAT THE DATA MAY BE USED AGAIN. THE FORM OF THE RESTORE STATEMENT
*// IS
*// RESTORE (RESTORE NUMERIC DATA)
*// (RESTORE ALPHA-NUMERIC DATA)
*// YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT THE READ STATEMENTS AS
*// PART OF A BASIC PROGRAM
*// CORRECT, 25, 0.0, 10X, 18.75X, 10.0, 10X, J.E. SMITH
*// READ A, B3, 15, C1, D, 5X, READ J$

```

```

* E4,K$,X'//)
  READ(5,101,END=319)CARD
319 WRITE(6,119)
119 FORMAT('O',10X,' I.) WHAT IS THE VALUE OF C1 ?? REPLY WITH VALUE'//
*//)
  15 READ(5,101,END=320)CARD
    CALL CRUNCH(1,ILNGTH)
    IF(CARDP(1).EQ.ASTRK) GO TO 15
    IF(CARDP(1).EQ.DIGIT(8)) GO TO 20
320 WRITE(6,120)
120 FORMAT('O',10X,' THE CORRECT ANSWER IS C1=7 . THE DATA IS PUT '//'
*//) INTO TWO FIFO STACKS , ONE FOR NUMERIC DATA , AND ONE FOR ALPHA-//
*//) NUMERIC DATA , WHEN THE READ STATEMENTS ARE EXECUTED , THE '//'
*//) VARIABLES ARE ASSIGNED AS FOLLOWS ://5X, A <-- 2, 10X, I$ <-- ANLESC1870
*//5X, B3 <-- 5, 10X, J$ <-- CORRECT//5X, C1 <-- 7, 10X, K$ <
*-- WRONG//5X, D <-- 16//5X, E4 <-- 25//5X, X <-- 0.0//)
20 WRITE(6,121)
121 FORMAT('O',10X,' NOW CONSIDER THAT THE FOLLOWING STATEMENTS '//'
*//) WERE ADDED TO THE ABOVE PROGRAM ://5X, RESTORE//5X, A '//'
*//) 10X, 2.) WHAT IS THE VALUE OF Z4 ?? REPLY WITH VALUE . '//)
25 READ(5,101,END=322)CARD
  CALL CRUNCH(1,ILNGTH)
  IF(CARDP(1).EQ.ASTRK) GO TO 25
  IF(CARDP(1).EQ.DIGIT(3).AND.CARDP(2).EQ.BLANK) GO TO 30
322 WRITE(6,122)
122 FORMAT('O',10X,' THE CORRECT ANSWER IS Z4=2 . THE RESTORE COMMAND'//
*//) RETURNS THE NUMERIC DATA STACK TO ITS ORIGINAL CONDITION, AND'//
*//) THE READ STATEMENT ASSIGNS VALUES FROM THE TOP OF THE DATA '//'
*//) STACK TO THE VARIABLES IN THE READ LIST AS FOLLOWS ://5X, Z4 <--
*//5X, A <-- 5 '//)
30 WRITE(6,123)
123 FORMAT('O',10X,' F. SUMMARY
*//) NOW THAT YOU HAVE SEEN HOW READ AND DATA STATEMENTS WORK TOGETHER'//
*//) TO INPUT VALUES INTO YOUR PROGRAM , AND HOW THE PRINT STATEMENT
*//) IS USED TO OUTPUT AND LABEL RESULTS YOU HAVE THE FACILITY '//'
*//) TO WRITE SIMPLE PROGRAMS USING INPUT DATA .//5X, FOR EXAMPLE '//'
*//) COULD BE READ IN A FOLLOWS: //5X, READ A,B//5X, PRINT,ANS=, (LESC2130
*//) ( 8**2)**5) , DATA 8,2//5X, END//. IN THE NEXT LESSON YOU WLESC2140
*//) WILL BE SHOWN HOW TO USE ASSIGNMENT STATEMENTS
*//) YOU GREATER FLEXIBILITY IN WRITING EXPRESSIONS
*//) AND WILL ALLOW YOU TO DO ASSIGNMENTS SUCH AS ://5X, A=B**2-4*A*C
*//) AND Z=(A**2+3)/A , THEN BY SAYING PRINT A,Z '//' THE RESULTS OF B
*//) OTHER EXPRESSIONS WOULD BE DISPLAYED.'//)
324 WRITE(6,124)
124 FORMAT('O',10X,' YOU WILL NOW BE GIVEN SOME REPRESENTATIVE PROBLEMS'//
*//) TO GIVE YOU A CHANCE TO EXERCISE YOUR NEW PROGRAMMING TOOLS'//

```

```

*/10X, '1.) WRITE A PROGRAM TO SOLVE THE EQUATION. X**2+10Y-24. '// LES022240
**WHERE THE INPUT DATA IS X=10, Y=3. '//10X, '2.) WRITE A PROGRAM LES022250
**TO SOLVE THE QUADRATIC EQUATION: '// (-B+(B**2-4*A*C)**.5)/2A
**INPUT DATA IS A=2, B=5, C=2. '// IF YOU WISH TO WRITE AND EXECUTE LES022270
**E THESE PROGRAMS NOW, REPLY : YES.// AND YOU WILL ENTER THE CAIBAL LES022280
**SIC COMPILER. OTHERWISE REPLY : NO.// AND YOU WILL GO ON TO THE N LES022290
**EXIT LESSON.//)
35 READ(5,101,END=325)CARD LES022300
CALL CRUNCH(1,END=325)CARD LES022310
IF(CARDP(1).EQ.ASTRSK) GO TO 35 LES022320
IF(CARDP(1).EQ.ALPHA(25))GO TO 45 LES022330
IF(CARDP(1).EQ.ALPHA(14))GO TO 40 LES022340
325 WRITE(6,125) LES022350
FORMAT(10,1) ** YOUR REPLY IS INCORRECTLY TYPED , REPLY AGAIN *** LES022360
**//) LES022370
40 WRITE(6,126) LES022380
FORMAT(10,1) IF YOU DECIDE TO RUN THESE PROBLEMS LATER THE ANSWERS LES022390
**// WILL NOW BE GIVEN SO YOU MAY CHECK YOUR RESULTS ://5X, '1.) LES022400
*106.//5X, '2.) -.500.//) LES022410
CALL EXIT LES022420
CALL TEST1 LES022440
WRITE(6,127) 10X, ' REPLY WITH YOUR ANSWER TO PROBLEM 1.)'//) LES022450
FORMAT(10,1,END=328)CARD LES022460
35 READ(5,101,END=328)CARD LES022470
IF(CARDP(1).EQ.ASTRSK)GO TO 50 LES022480
IF(CARDP(1).EQ.DIGIT(2).AND.CARDP(3).EQ.DIGIT(7)) GO TO 55 LES022490
328 WRITE(6,128) LES022500
FORMAT(10,1) THE CORRECT ANSWER IS 106 AND THE PROGRAM SHOULD '// LES022510
**HAVE BEEN SIMILAR TO ://5X, ' READ X,Y//5X, ' PRINT:ANSWER=, (X** LES022520
*2 + 10*Y - 24) //5X, ' DATA 10,3//5X, ' END.//) LES022530
55 WRITE(6,129) LES022540
FORMAT(10,1,10X, ' REPLY WITH YOUR ANSWER TO PROGRAM 2.)'//) LES022550
35 READ(5,101,END=62)CARD LES022560
CALL CRUNCH(1,END=62)CARD LES022570
IF(CARDP(1).EQ.ASTRSK) GO TO 60 LES022580
IF(CARDP(1).EQ.CMINUS.AND.CARDP(2).NE.DECMAL)GO TO 62 LES022590
IF(CARDP(3).EQ.DIGIT(6))GO TO 70 LES022600
62 WRITE(6,130) LES022610
FORMAT(10,1,10X, ' THE CORRECT ANSWER IS -.50 AND THE PROGRAM SHOULD LES022620
**// LOOK SIMILAR TO ://5X, ' READ A,B,C//5X, ' PRINT:ANSWER=, (-B LES022630
* + (B**2 - 4*A*C)**.5) / 2*A//5X, ' DATA 2,5,2//5X, ' END.//) LES022640
70 CALL EXIT LES022660
END

```

LESSON 3 PRESENTS THE 'LET' STATEMENT AND BUILT-IN FUNCTIONS

COMMON
STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
LES04010
LES04020

```

LES04030
LES04040
LES04050
LES04060
LES04070
LES04080

- - - -
100  IAPTR, INPTR, IADATA(500), XNDATA(500), ISTLST(100), STRING(5),
      DIGIT(10), IPRTB(10), LISIST(100), ISTLST(100), QUOTE,
      PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, COMMA,
      EQUALS, PARRT, DECMAL, PLUS, CMINUS, SLASH, COMMA,
      PARLFT, ASTRSK, BLANK
      COMMON INTERP, IEXERR, /
      REAL*8 LES37, LESON3
      CALL ALOAD(LES3, N1)
      WRITE(6, 100)
      FORMAT(0, 5X, ' *** LESSON 3. ***', //)
      * THIS INSTRUCTION SET WILL INTRODUCE YOU TO ASSIGNMENT STATEMENTS. //
      * AND BUILT-IN FUNCTIONS. THE LET STATEMENT AND THE IO BUILT IN. //
      * FUNCTION WILL ENABLE YOU TO EVALUATE AND ASSIGN VARIABLES TO. //
      * COMPLEX ARITHMETIC EXPRESSIONS. //)
      READ(5, 101, END=302) CARD
      FORMAT(180A1)
      WRITE(6, 102)
      FORMAT(0, 10X, ' A. LET', //)
      * THE LET STATEMENT IS AN ASSIGNMENT OR SUBSTITUTION COMMAND. IT. //
      * CAUSES THE EVALUATION OF AN EXPRESSION TO BE SUBSTITUTED FOR THE. //
      * CURRENT VALUE OF A VARIABLE. THE FORM OF THE LET STATEMENT IS: //
      * 10X, ' LET << VARIABLE >> = << EXPRESSION >> OR', //10X, '
      * LET << VARIABLE >> = << VARIABLE >> = ..... = << EXPRESSION >> //)
      READ(5, 101, END=303) CARD
      WRITE(6, 103)
      FORMAT(0, 10X, ' B. THERE MAY BE ANY NUMBER OF VARIABLE = VARIABLE. //
      * IN THE FORM OF THE LET STATEMENT. AN EXPRESSION IS A NUMBER. //
      * VARIABLE OR AN ARITHMETIC EXPRESSION. //10X, '
      * WHEN THE LET STATEMENT IS EXECUTED AND THE EXPRESSION ON THE RIGHT. //
      * SIDE OF THE LET STATEMENT IS EVALUATED AND THE RESULTING VALUE IS. //
      * ASSIGNED TO THE VARIABLE OR VARIABLES ON THE LEFT SIDE OF THE EQUA
      * LES. // SIGN. THE PREVIOUS VALUE ASSIGNED TO THE VARIABLE OR VARIABLE
      * = 14.4 //10X, ' BE LET A=8.25 // THE VALUE OF THE VARIABLES A AND B IS
      * NOW A=B=25. //)
      READ(5, 101, END=304) CARD
      WRITE(6, 104)
      FORMAT(0, 10X, ' C. THE ONLY RESTRICTION ON THE USE OF VARIABLES. //
      * IS THAT SIMPLE AND SUBSCRIPTED VARIABLES CAN ONLY BE ASSIGNED. //
      * NUMERIC VALUES, AND ALPHA VARIABLES CAN ONLY BE ASSIGNED ALPHA. //
      * NUMERIC STRINGS. FOR EXAMPLE: //10X, ' LET A=D4=X(5)=(3**2 -
      * 6)/4 //10X, ' LET A$=X$. HELP. //10X, ' LET Y=(B - 4*A*C)**.5 //10X, '
      * LET O$=ANSWER. //)
      * LET A=D$=(3+4) //)
      READ(5, 101, END=305) CARD
      WRITE(6, 105)
      FORMAT(0, 10X, ' D. YOU WILL NOW BE ASKED QUESTIONS CONCERNING', //
      * WHAT YOU HAVE JUST LEARNED: //)

```

```

106 WRITE(6,106)
    FORMAT(0,'10X',1.) REPLY WITH VALUE OF X IN BELOW PROGRAM './11
    *OX, LET A=4./10X, LET B=6./10X, LET C=3./10X, LET X=A**2 - 4*B
    *+ 3*C./10X, PRINT, ANSWER =,X./10X, END./)
5 READ(5,101,END=306) CARD
    CALL CRUNCH(ILNGTH)
    IF(CARDP(1).EQ.ASTRSK) GO TO 5
    IF(CARDP(1).EQ.DIGIT(2)) GO TO 10
    IF(CARDP(1).EQ.PLUS.AND.CARDP(2).EQ.DIGIT(2)) GO TO 10
306 WRITE(6,107)
1107 FORMAT(0,' YOUR RESPONSE WAS INCORRECT. THE EXPRESSION TO './
    *EVALUATED IS : X=4**2 - 4*B + 3*C WHICH EVALUATES AS :15X, X <
    *-- 16 - 24 + 9 : X <-- + 1./)
10 WRITE(6,108)
1108 FORMAT(0,'10X',2.) REPLY WITH VALUE OF Z IN BELOW PROGRAM './11
    *OX, DATA 1,2,3,4,6./10X, DATA, RIGHT ON.,7,8./10X, READ A,B,X,
    */10X, READ G,Y,C,D./10X, LET Z=X+Y./10X, END./)
15 READ(5,101,END=309) CARD
    CALL CRUNCH(ILNGTH)
    IF(CARDP(1).EQ.ASTRSK) GO TO 15
    IF(CARDP(1).EQ.DIGIT(8)) GO TO 20
309 WRITE(6,109)
1109 FORMAT(0,' YOUR RESPONSE WAS INCORRECT. THE VARIABLES ARE './
    *-- 1,10X, Y<-- RIGHT ON./5X, NUMERIC,10X, ALPHA,/5X, A<
    */5X, C<-- 6./5X, D<-- 7./5X, Z<-- X+Y : Z <-- 7./)
20 WRITE(6,110)
1110 FORMAT(0,'10X',3.) LET R= A+B/C-D./ WHICH FORMULA DOES THIS
    * STATEMENT REPRESENT ?./10X, REPLY WITH CORRECT LETTER./10X, A.
    *) R=(A+B)/(C-D)/10X, B.) R= A+(B/C)-D./)
25 READ(5,101,END=311) CARD
    CALL CRUNCH(ILNGTH)
    IF(CARDP(1).EQ.ASTRSK) GO TO 25
    IF(CARDP(1).EQ.ALPHA(2)) GO TO 30
311 WRITE(6,111)
1111 FORMAT(0,' YOUR RESPONSE WAS INCORRECT. WHEN THERE ARE NO './
    *PARENTHESES IN AN EXPRESSION, THE HIERARCHY OF OPERATORS APPLIES.
    */. THUS IN THIS EXPRESSION THE DIVIDE OPERATION IS DONE FIRST./
    * THEN ADDITION AND SUBTRACTION. IF YOU ARE STILL HAVING PROBLEMS./
    * WITH EXPRESSIONS YOU HAD BETTER REVIEW YOUR SESSION ON LESSON 1.
    *./)
30 WRITE(6,112)
1112 FORMAT(0,'10X',B. BUILT-IN FUNCTIONS
    * BUILT-IN FUNCTIONS ARE COMMONLY USED FOR YOUR USE. THERE ARE
    * AND STORED IN THE CAIBASIC COMPILER FOR LOGARITHMS, ABSOLUTE VALUES,
    * FUNCTIONS TO FIND SQUARE ROOTS, THE FORM FOR THE BUILT-IN FUNCTIONS, IS
    * AND TRIGONOMETRIC VALUES. WHERE THE EXPRESSION >>./ WHERE THE EXPRESSION
    * :./10X, FUNCTION NAME << (EXPRESSION) >>./
    *./)

```



```

50 READ(5,101,END=319)CARD
   IF(CARDP(1).EQ.ASTRSK) GO TO 50
319 WRITE(6,119)
119 *UNDEFINED OPERATION. NO THE SQUARE ROOT OF A NEGATIVE NUMBER IS AN
   *BASIC STATEMENT FOR TESTING AND BRANCHING TO ANOTHER SEGMENT
   *OF THE PROGRAM IF THE TEST IS TRUE. FOR EXAMPLE THE ABOVE
   *PROGRAM SEQUENCE MIGHT BE ALTERED AS FOLLOWS:
   *IF B LT 100 THEN 100/5X, 50 LET X=SQR(X)/10X, LET B=-9/5X
   */10X, 50 REM NEGATIVE ARGUMENT
   */10X, 50 LET X=SQR(X)/10X, LET B=ABS(B)/5X
   *GO TO 50/10X, 50 REM THIS PROGRAM SEQUENCE TESTS
   *FOR A NEGATIVE ARGUMENT, IF TRUE, IT BRANCHES TO STATEMENT NUMB
   *ER 100, MAKES THE ARGUMENT POSITIVE, AND BRANCHES BACK TO STATE
   *MENT 50, TO COMPLETE THE PROGRAM
   READ(5,101,END=320)CARD
320 WRITE(6,123)
123 *WITH THE LET STATEMENT AND BUILT-IN FUNCTIONS, PLUS THE PREVIOUS
   *FAST BASIC STATEMENTS (REM, READ, DATA, PRINT), YOU ARE
   *THE NEXT LESSON YOU WILL LEARN HOW TO SET UP LOOPS IN A PROGRAM
   *SO THAT THE MAIN BODY OF A PROGRAM MAY BE EXECUTED AS OFTEN
   *AS DESIRED. AS YOU ARE DOING YOUR REVIEW PROBLEMS, THINK ABOUT
   *HOW YOU COULD SET UP A LOOP TO READ IN ANY AMOUNT OF DATA
   *PROCESS IT AND THEN HALT FOR SOME TEST CONDITION
   READ(5,101,END=324)CARD
324 WRITE(6,124)
124 *YOUR PROGRAMMING SKILLS TO DATE, THE FOLLOWING REVIEW PROBLEMS WILL EXERCISE
   *MPUTE THE PRESENT WORTH OF AN INVESTMENT FOR SOME NUMBER OF YEA
   *RS. HENCE THE FORMULA IS:  $P = S(1/(1+I)^N)$ , WHERE P
   *IS THE PRESENT WORTH OF A PAYMENT S IN N YEARS. HENCE, AT AN
   *INTEREST RATE OF I, FOR DATA USE I=.08, S=5000, N=20,
   *2) WRITE A PROGRAM TO FIND SIDES A, AND C OF A TRIANGLE USING:
   */ THE LAW OF SINES FORMULA:  $A/\sin(A) = B/\sin(B) = C/\sin(C)$ 
   */ WHERE THE NUMERATOR IS THE SIDE AND THE DENOMINATOR IS THE
   */ SINE OF THE ANGLE, THE CONVERSION FACTOR FROM DEGREES TO RADIANS
   *IS:  $1 \text{ DEG} = 3.1416/180 \text{ RADIANS}$ . THE DATA FOR THE PROGR
   *AM IS: SIDE B=34.91 IN, SIDE C=31.32 DEG. IF YOU WISH TO RUN THE
   *PROGRAM CRUNCH(LENGTH)
   IF(CARDP(1).EQ.ASTRSK) GO TO 60
   IF(CARDP(1).EQ.ALPHA(14)) GO TO 65
   IF(CARDP(1).EQ.ALPHA(25)) GO TO 70
325 WRITE(6,125)
125 *YOUR REPLY IS INCORRECT, REPLY AGAIN
65 WRITE(6,126)

```

```

126 FORMAT('0',, THE ANSWERS TO THE PROBLEMS WILL NOW BE GIVEN SO './,
* THAT YOU MAY CHECK YOUR RESULTS LATER :./5X,.1.) $1072.74./5X, 2
*.) SIDE A=45.06 IN. AND SIDE C=23.69 IN. './)
CALL EXIT
CALL TEST1
70 WRITE(6,127)
127 FORMAT('0',,10X, REPLY WITH ANSWER TO QUESTION 1.) ??'./)
75 READ(5,101,END=80) CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 75
IF(CARDP(1).NE.DIGIT(2).OR.CARDP(4).NE.DIGIT(3)) GO TO 80
IF(CARDP(5).EQ.DECMAL.AND.CARDP(6).EQ.DIGIT(8)) GO TO 90
80 WRITE(6,128)
128 FORMAT('0',,10X, THE CORRECT ANSWER IS $1075.74 AND THE PROGRAM./,
* SHOULD HAVE BEEN SIMILAR TO :./5X, READ S,I,N./5X, LET P=S*(1/
* ((1+I)**N))./5X, PRINT, PRESENT WORTH=, P./5X, DATA 5000, .08, 20,
*/5X, END./)
90 WRITE(6,129)
129 FORMAT('0',,10X, REPLY WITH ANSWER TO SIDE A FOR QUESTION 2.) './)
91 READ(5,101,END=95) CARD
IF(CARDP(1).EQ.ASTRSK) GO TO 91
IF(CARDP(1).NE.DIGIT(5).OR.CARDP(2).NE.DIGIT(6)) GO TO 95
IF(CARDP(3).EQ.DECMAL.AND.CARDP(4).EQ.DIGIT(1)) GO TO 97
95 WRITE(6,130)
130 FORMAT('0',,10X, THE CORRECT ANSWER IS SIDE A=45.06 IN. AND
* SIDE C=23.6 IN. THE PROGRAM SHOULD HAVE BEEN SIMILAR TO :./5X,
* READ A,B,C,81./5X, LET A1=(B1*SIN(A*(3.1416/180)))/SIN(B*3.141
* 6/180))./5X, LET C1=(B1*SIN(C*(3.1416/180)))/SIN(B*(3.1416/180)).
*/5X, PRINT, SIDE A=, A1, SIDE C=, C1./5X, DATA 98.71, 49.97,
* 31.32, 34.91./5X, END./)
97 CALL EXIT
END

```

THIS LESSON INTRODUCES BRANCHING , BOTH CONDITIONAL AND UNCONDITIONAL .

```

- - - - - COMMON
STACK(100), PRUG(200), CARD(80), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRIB(10), LISTST(100), ISTLST(100),
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLFT, ASTRSK, BLANK
COMMON INTERP, IEXERR,
REAL*8 LES4, LESON4
CALL ALOAD(LES4, N1)
WRITE(6,100)
100 FCRMAT('0',,5X, *** LESSON 4 ***
*./)
LES000030
LES000040
LES000050
LES000060
LES000070
LES000080
LES000090
LES000100
LES000110
LES000120

```



```

* VALUE OF THE EXPRESSION , FOR EXAMPLE ://5X, LET I=3*/5X, ON I
* GO TO 100133,475,9999://)
* READ(5,101,END=306)CARD
306 WRITE(6,106)
106 FORMAT(0, , EXECUTION OF THE **COMPUTED GO TO** WOULD CAUSE **
* PROGRAM CONTROL TO TRANSFER UNCONDITIONALLY TO STATEMENT NUMBER 475
* **/5X, YOU MUST BE CAREFUL WHEN USING THE **COMPUTED GO TO**//
* ** NOT ONLY BECAUSE OF INFINITE LOOPS ; BUT BECAUSE THE EXPRESSION**
* ** MUST BE AN INTEGER BETWEEN 1-->9999 , AND THERE MUST BE A **
* ** STATEMENT NUMBER FOR **ALL** POSSIBLE VALUES OF THE EXPRESSION **
* **)
* READ(5,101,END=307)CARD
307 WRITE(6,107)
107 FORMAT(0, ,5X, , B. CONDITIONAL BRANCHING ONLY IF CERTAIN **/5X,
* ** THE CONDITIONS ARE TRUE . IF THE TEST OF RELATIONS IS TRUE THEN **/
* ** TRANSFER OF CONTROL OCCURS ; OTHERWISE PROGRAM CONTROL CONTINUES **/
* ** WITH THE NEXT STATEMENT >><< RELATION >><< EXPRESSION >> THEN
* **// 1X, IF << EXPRESSION >><< RELATION >><< EXPRESSION >> THEN
* ** STATEMENT NUMBER >> **// NOTE THAT ALPHA VARIABLES ARE NOT ALLOWED
* ** AS AN EXPRESSION IN A **// CONDITIONAL BRANCH **//)
* READ(5,101,END=308)CARD
308 WRITE(6,108)
108 FORMAT(0, ,5X, , THE RELATIONS ARE :
* ** SYMBOLS : GT, ,5X, , EXAMPLE, ,5X, , MEANING
* **//2X, , GT, ,5X, , A GT B, ,5X, , A GREATER THAN B, ,2X, , GE, ,5X, , A GE
* ** B, ,5X, , A GREATER THAN OR EQUAL TO B, ,2X, , LT, ,5X, , A LT B, ,5X, ,
* ** A LESS THAN B, ,2X, , LE, ,5X, , A LE B, ,5X, , A LESS THAN OR EQUAL TO
* ** B, ,2X, , NE, ,5X, , A NE B, ,5X, , A NOT EQUAL TO B, ,3X, , =, ,5X, , A = B
* **//6X, , A EQUAL B, ,//)
* READ(5,101,END=309)CARD
309 WRITE(6,109)
109 FORMAT(0, ,5X, , WHEN THE CONDITIONAL STATEMENT IS EXECUTED THE **/
* ** EXPRESSION RELATION EXPRESSION ) IS TESTED , AND IF THE RELATION
* **// IS TRUE , THEN CONTROL IS TRANSFERRED TO THE STATEMENT NUMBER.
* **// OTHERWISE , PROGRAM CONTROL CONTINUES TO THE NEXT SEQUENTIAL **/
* ** STATEMENT . FOR EXAMPLE ://5X, , 1 READ A, /5X, , **/5X, , **/5X, ,9
* ** IF A LT C, THEN 90, /5X, , LET X=SQR(A), /5X, , **/5X, , **/5X, ,9
* ** PRINT **// ILLEGAL ARGUMENT **// A, /5X, , GO TO , /5X, , **// THE ONLY
* ** TIME THAT THE CONDITIONAL BRANCH IS EXECUTED IS WHEN **// A IS LESS
* ** THAN 0 **//)
* READ(5,101,END=310)CARD
310 WRITE(6,110)
110 FORMAT(0, ,5X, , YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT BRANCHING
* **//10X, ,1, , ARE THE FOLLOWING STATEMENTS CORRECT? **// REPLY: YES OR
* ** NC) **//15X, , GO TO END **//)
5 READ(5,101,END=211)CARD
5 CALL CRUNCH(11,LENGTH)

```

```

IF(CARDP(1).EQ.ASTRSK) GO TO 5
IF(CARDP(1).EQ.ALPHA(14))GO TO 10
311 WRITE(6,111)
111 FORMAT(0, ' YOUR ANSWER IS INCORRECT . THE ONLY THING THAT IS'//
*ALLOWED AFTER A 'GO TO' IS A STATEMENT NUMBER BETWEEN 1-->9999'//
*)
10 WRITE(6,112)
112 FORMAT(0, '15X, ' IF X1 LT A$ THEN 10'//)
15 READ(5,101,END=313)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 15
IF(CARDP(1).EQ.ALPHA(14)) GO TO 20
313 WRITE(6,113)
113 FORMAT(0, ' YOUR ANSWER IS INCORRECT . ALPHA VARIABLE MAY NOT'//
*BE USE FOR TESTING PURPOSES IN A CONDITIONAL STATEMENT .'//)
20 WRITE(6,114)
114 FORMAT(0, '15X, ' IF(X*3-4) GE 10 THEN GO TO 100'//)
25 READ(5,101,END=315)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 25
IF(CARDP(1).EQ.ALPHA(14)) GO TO 30
315 WRITE(6,115)
115 FORMAT(0, ' YOUR ANSWER IS INCORRECT . THE ONLY THING ALLOWED'//
*AFTER THEN IS A STATEMENT NUMBER .'//)
30 WRITE(6,116)
116 FORMAT(0, '15X, 'LET Y=6, '15X, ' ON Y GO TO 1,3,5,7,999'//)
35 READ(5,101,END=317)CARD
CALL CRUNCH(ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 35
IF(CARDP(1).EQ.ALPHA(14)) GO TO 40
317 WRITE(6,117)
117 FORMAT(0, ' YOUR ANSWER IS INCORRECT . THERE ARE NOT ENOUGH '//
*STATEMENT NUMBERS IN THE 'COMPUTED GO TO'.'//)
40 WRITE(6,118)
118 FORMAT(0, '1CX, ' 2.) CONSIDER THIS PROGRAM SEGMENT , ANY ERRORS (R
EPLY: YES OR NO) '15X, '1 READ A,B'15X, ' 3 LET Z=A*B'15X, ' IF Z
*GO TO 999'15X, ' ...'15X, ' GO TO 3'15X, ' DATA 3,5,7,
*10'15X, ' 999 END'//)
319 READ(5,101,END=319)CARD
119 WRITE(6,119)
119 FORMAT(0, ' THERE IS AN 'INFINITE' LOOP IN THIS PROGRAM 'IT'//
*COULD BE CORRECTED BY CHANGING THE GO TO 3 TO GO TO 1'15X, '
*WHEN YOU WRITE PROGRAMS CR LOOPS YOU MUST ALWAYS
*/ CONSIDER HOW THE PROGRAM WILL STOP. YOU HAVE OBSERVED THAT '
*AN 'INFINITE LOOP' CAN BE STOPPED BY HAVING A READ STATEMENT '
*IN THE LOOP AND JUST RUN OUT OF DATA . HOWEVER, ENDING A PROGRAM
*/ ON AN ERROR IS AN INELIGANT METHOD . THE MOST COMMON PROGRAM
* USE THE 'GO TO' AND 'IF/THEN' COMMANDS TO CONTROL PROGRAM'//

```

```

* LOOPS AND ARE AS FOLLOWS : (//)
320 READ(5,101,END=320)CARD
120 WRITE(6,120)
* IS INCREMENTED, IN THE LOOP, AND WHEN THE COUNTER REACHES A
* CERTAIN VALUE, BRANCH OUT OF THE LOOP. EXAMPLE PROGRAM : //10X, //
* REM COUNT AND TEST METHOD, //10X, REM N IS COUNTER, INITIALIZED TO
* 0, AND COUNTS FROM 1-->10, //10X, LET N=2, //10X, READ X, //10X, GO
* 0 LET Z=Z+X, //10X, LET N=N+1, //10X, IF N GT 10 THEN 100, //10X,
* TO 10, //10X, SUM=, //2, //10X, DATA 10, //10X, END, //
* PLY WITH VALUE CF Z, (//)
45 READ(5,101,END=50)CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 45
IF(CARDP(1).NE.DIGIT(2).OR.CARDP(2).NE.DIGIT(2)) GO TO 50
IF(CARDP(3).EQ.DIGIT(1)) GO TO 60
50 WRITE(6,121)
121 FORMAT(0, YOUR ANSWER IS INCORRECT, THE ONLY WAY TO BE
* SURE OF VALUES IN A LOOP IS TO SET UP A TABLE OF THE VARIABLES, //
* AND KEEP TRACK OF THERE VALUES IN THE LOOP : //5X, X, 5X, N, 5X, Z,
* /4X, 10, 4X, 0, 5X, 0, //10X, 1, 5X, 10, //10X, 2, 5X, 20, //10X, 3, 5X,
* /30, //10X, 0, //10X, 9, 5X, 90, //10X, 10, 4X, 100, //10X, 1,
* 1, 4X, 110, //, THUS Z=110, //)
60 WRITE(6,122)
122 FORMAT(0, 5X, 2, READ AND TEST METHOD, IN WHICH A VALUE IS, //
* READ IN AND TESTED FOR THE END OF LOOP CONDITION, //10X, //10X,
* REM READ AND TEST DEMONSTRATION, //10X, REM 9999 IS END VALUE, //10X,
* LET Z=0, //10X, IF X=9999 THEN 9999, //10X, LET Z=
* Z+X, //10X, GO TO 1, //10X, DATA 1, 2, 3, 4, 5, 6, 9999, //10X, PRINT
* SUM=, //2, //10X, END, //, REPLY WITH VALUE OF Z, //)
65 READ(5,101,END=65)CARD
CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 65
IF(CARDP(1).EQ.DIGIT(3).AND.CARDP(2).EQ.DIGIT(2)) GO TO 70
323 WRITE(6,123)
123 FORMAT(0, YOUR ANSWER IS INCORRECT, VARIABLE VALUES IN THE, //
* LOOP ARE : //5X, X, 5X, Z, //5X, 1, 5X, 1, //5X, 2, 5X, 3, 5X, 6,
* /5X, 4, 5X, 10, //5X, 5, 5X, 15, //5X, 6, 5X, 21, //5X, 9999, //, THUS Z
* =21, //)
70 WRITE(6,124)
124 FORMAT(0, 10X, C, SUMMARY
* YOU HAVE SEEN HOW THE UNCONDITIONAL BRANCHES (GO TO, AND
* COMPUTED GO TO, TRANSFER PROGRAM CONTROL AND HOW THE
* CONDITIONAL BRANCH (IF/THEN,)) TESTS FOR TRANSFER OF PROGRAM, //
* CONTROL, AND YOU HAVE OBSERVED THE CONTROL CF LOOPS SO THAT
* A PROGRAM SEGMENT MAY BE REPEATED UNTIL A SPECIFIED CONDITION, //
* IS MET. IN THE NEXT LESSON YOU WILL LEARN A BASIC STATEMENT TO, //
* CONTROL A LOOP BY THE INCREMENT AND TEST METHOD. //, THE FOLLOWING

```



```

-- -- --
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRITB(10), LIST(100), IIST(100),
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PARRT, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLFT, ASTRSK, BLANK
INTERP, IEXERR, /
REAL*8 LES5/LES5,N1)
CALL ALDAD(LES5,N1)
WRITE(6,100)
FORMAT(0,5X, ' ** LESSON 5 **
* THIS LESSON WILL INTRODUCE YOU TO ITERATION, SUBSCRIPTED
* VARIABLES, AND LISTS (VECTORS) AND TABLES (MATRICES).
* A. ITERATION (LOOPING)
* IN THE LAST LESSON YOU WERE SHOWN HOW TO USE CONDITIONAL AND
* UNCONDITIONAL BRANCHES TO CONTROL LOOPING. THE COUNT AND TEST
* ITERATIVE LOOP OCCURS SO FREQUENTLY THAT AN ABBREVIATED BASIC
* STATEMENT HAS BEEN DEvised TO CONTROL LOOPING. THE ITERATIVE
* LOOP HAS THE FOLLOWING FORM:
* FOR << SIMPLE VARIABLE >> = << EXPRESSION >> TO << EXPRESSION >> /
* ZCX, << STEP << EXPRESSION >> /12X, '.../12X, '.../10X,
* NEXT, << SIMPLE VARIABLE >> /11)
READ(5,101,END=302)CARD
FORMAT(80A1)
302 WRITE(6,102)
102 FORMAT(0, ' FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT:
* THE SIMPLE VARIABLE I, LET X=X+1./5X, THE LOOP INDEX I
* THE FOR/NEXT PAIR IS EXECUTED, THE LOOP INDEX IS GIVEN THE
* VALUE (INITIALIZED) OF THE FIRST EXPRESSION (I=1 IN EXAMPLE).
* THIS INDEX IS THEN TESTED TO DETERMINE WHETHER IT IS GREATER
* THAN THE SECOND EXPRESSION AFTER TO THE STATEMENT FOLLOWING IS.
* GREAT. OTHERWISE THE REMAINING STATEMENTS WITHIN THE LOOP
* (FOR/NEXT) ARE EXECUTED SEQUENTIALLY UNTIL THE NEXT STATEMENT
* IS REACHED.
READ(5,101,END=303)CARD
303 WRITE(6,103)
103 FORMAT(0, ' WHEN THE NEXT STATEMENT IS REACHED, THE LOOP
* INDEX IS INCREASED (INCREMENTED) BY THE AMOUNT OF THE EXPRESSION
* FOLLOWING STEP, AND CONTROL IS TRANSFERRED BACK TO THE FOR
* STATEMENT WHERE THE LOOP CONTINUES UNTIL THE INDEX VALUE
* GREATER THAN THE FINAL VALUE FOR EXAMPLE:
* GREM DEMO LOOP SUM FROM 1-->10. /5X, ' LET C=0./5X, SUM=
* R I=1 TO 10 STEP 1./5X, ' LET C=C+1./5X, NEXT I, PRINT SUM=
* C./5X, ' END.
READ(5,101,END=304)CARD
304 WRITE(6,104)
104 FORMAT(0, ' YOU WILL NOTICE THAT THE SIMPLE VARIABLE FOLLOWING'

```

```

**NEXT** IS THE SAME AS THE SIMPLE VARIABLE FOLLOWING..FOR..//
**AND THAT THE NEXT..STATEMENT MARKS THE END OF THE LOOP..//X..
**BECAUSE THE INCREMENT AND ITS EXPRESSION MAY BE COMMONLY ONE(1), THE..
**STEP..MODIFIER WILL BE ASSUMED TO BE OMITTED, AND THE..
**INCREMENT VALUE STATEMENT BE ASSUMED TO BE ONE(+1). FOR EXAMPLE THE..
**ABOVE..FOR..STEP..MAY BE..
**FOR I=1 TO 10//5X, THE INCREMENT VALUE AFTER STEP..MAY BE..
**POSITIVE OR NEGATIVE ALLOWING THE FLEXIBILITY OF LOOPING FORWARD..
**OR BACKWARD..STEP..VALUE THE TEST BECOMES..
**LESS THAN..FOR EXAMPLE THE FOLLOWING..FOR..STATEMENTS ARE..
**EQUIVALENT..//5X, FOR I=1 TO 10//5X, FOR I=10 TO 1 STEP -1..
**READ(5,101,END=305)CARD
305 WRITE(6,105)
105 FORMAT(10,10X, //)
**ANOTHER USEFUL TECHNIQUE OF LOOPING IS "NESTING"
**THE NESTING LOOP REFERS TO PLACING ONE LOOP INSIDE ANOTHER LOOP..
**IS INCREMENTED..SPINS..AROUND AS MANY TIMES AS THE OUTER LOOP..
**FOR I=1 TO 10//5X, FOR J=1 TO 20//6X, //6X, //5X,
**NEXT J//6X, NEXT I//. IN THIS EXAMPLE THE OUTSIDE LOOP(I) IS..
**REPEATED 10 TIMES, AND THE INNER LOOP(J) WOULD BE REPEATED 20..
**TIMES FOR EACH INCREMENT UP TO A MAXIMUM OF 20, OR 200 REPETITIONS..
**LOOPS MAY BE NESTED TO A MAXIMUM OF 20; HOWEVER, THEY..
**CANNOT OVERLAP..THE INNERMOST LOOP MUST BE CLOSED WITH ITS..
**NEXT..STATEMENT BEFORE ENCOUNTERING THE NEXT OUTER LOOP..S..
**NEXT X=10 TO 1 STEP -1//5X, FOR Y=3 TO 5//5X, FOR Z=-3 TO -5 STEP
**-1//6X, //5X, NEXT Z//5X, NEXT Y//5X, NEXT X//
**READ(5,101,END=306)CARD
306 WRITE(6,106)
106 FORMAT(10,10X, //)
**WITHIN A FOR/NEXT LOOP CONDITIONAL AND UNCONDITIONAL
**OR WITHIN LIMITS OF THE SAME LOOP, HOWEVER, IT IS NOT POSSIBLE..
**TO BRANCH INTO THE MIDDLE OF A FOR/NEXT LOOP BECAUSE LOGIC..
**PROBLEMS OCCUR ABOUT IS USING THE INDEX VARIABLE OF THE FOR/NEXT..
**TO BE CAREFUL ABOUT IS USING THE INDEX VARIABLE OF THE LOOP INDEX..
**LOOP IN COMPUTATIONS, IF YOU ALTER THE VALUE OF THE LOOP INDEX..
**FOR I=1 TO 10//5X, LET I=I+10//5X, FOR EXAMPLE: //5X, EN
**D//, REPLY WITH VALUE OF I THAT IS PRINTED //
**READ(5,101,END=307)CARD
5 CALL CRUNCH(1,LENGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 5
307 WRITE(6,107)
107 FORMAT(10,10X, //)
**YOUR ANSWER IS INCORRECT..THE LOOP INDEX(I) IS..
**ALTERED IN THE LOOP (I=11), AND THEN INCREMENTED AND TESTED (I=12)
**THE FINAL VALUE OF I=12..
10 WRITE(6,108)

```

LES00500
 LES00510
 LES00520
 LES00530
 LES00540
 LES00550
 LES00560
 LES00570
 LES00580
 LES00590
 LES00600
 LES00610
 LES00620
 LES00630
 LES00640
 LES00650
 LES00660
 LES00690
 LES00700
 LES00710
 LES00720
 LES00730
 LES00740
 LES00750
 LES00760
 LES00770
 LES00780
 LES00800
 LES00810
 LES00820
 LES00830
 LES00840
 LES00850
 LES00860
 LES00870
 LES00880
 LES00890
 LES00900
 LES00920
 LES00930
 LES00940
 LES00950
 LES00960


```

*ZERO IN EACH ELEMENT OF THE LIST YOU ARE ASSURED THAT THE LIST'//
*IS CLEANED UP BEFORE YOU USE IT.'//)
321 READ(5,101,END=321)CARD
121 WRITE(6,121)
*YOU THE ABILITY TO USE TABLES, OR MATRICES, OR TWO DIMENSIONAL'//
*ARRAYS OF THE SUBSCRIPTS OF A TABLE REPRESENT THE ROWS AND'//
*COLUMNS OF THE TABLE. THE FIRST SUBSCRIPT IS THE ROW AND THE'//
*SECOND SUBSCRIPT IS THE COLUMN. FOR EXAMPLE IN TABLE'D'://5X,
*OF TABLE'D'://)
322 READ(5,101,END=322)CARD
122 WRITE(6,122)
*REFERENCED BY DEFINING THE PAIR OF SUBSCRIPTS AS DESIRED IN'//
*ROW-COLUMN ORDER. IN A 3X3 TABLE(MATRIX) THE TABLE IS REFERENCED'//
*AS FOLLOWS: 1,1,5X, 1,2,5X, 1,3,5X, 2,1,5X, 2,2,5X, 2,3,5X, 3,1,5X, 3,2,5X, 3,3,5X. WHERE THE FIRST SUBSCRIPT IS THE ROW AND THE SECOND SUBSCRIPT IS THE COLUMN. THE FOLLOWING TABLE IS USED IN THIS REFERENCE SYSTEM. CONSIDER HOW TO FILL THE FOLLOWING TABLE'//
*LE: 1,5X, 2,5X, 3,5X, 4,5X, 5,5X, 6,5X, 7,5X, 8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X
*8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X
323 READ(5,101,END=323)CARD
123 WRITE(6,123)
*THIS TABLE'X' COULD BE FILLED AS FOLLOWS: 1,5X, 2,5X, 3,5X, 4,5X, 5,5X, 6,5X, 7,5X, 8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X
*DIMX(3,3), 1,5X, 2,5X, 3,5X, 4,5X, 5,5X, 6,5X, 7,5X, 8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X
*5X, NEXT J, 1,5X, 2,5X, 3,5X, 4,5X, 5,5X, 6,5X, 7,5X, 8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X
*55 READ(5,101,END=324)CARD
*CALL CRUNCH(1,5X, 2,5X, 3,5X, 4,5X, 5,5X, 6,5X, 7,5X, 8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X)
*IF(CARDP(1)) EQ.ASTPSK) GO TO 55
324 WRITE(6,124)
124 *GO BACK AND LOOK AT THE REFERENCE SYSTEM. X(2,3) IS THE ELEMENT'//
*IN ROW 2, COLUMN 3.'//)
60 WRITE(6,125)
125 *PROGRAM (ASSUMING TABLE'X' HAS BEEN FILLED BY THE ABOVE PROGRAM) THIS'//
*5X, LET S=0, 1,5X, 2,5X, 3,5X, 4,5X, 5,5X, 6,5X, 7,5X, 8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X
*E J THEN 5,5X, LET S=S+X(I,J), 5,5X, 6,5X, 7,5X, 8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X
*PRINT, SUM=, 5,5X, 6,5X, 7,5X, 8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X
65 READ(5,101,END=326)CARD
126 *CALL CRUNCH(1,5X, 2,5X, 3,5X, 4,5X, 5,5X, 6,5X, 7,5X, 8,5X, 9,5X, 10,5X, 11,5X, 12,5X, 13,5X, 14,5X, 15,5X)
*IF(CARDP(1)) EQ.ASTRSK) GO TO 65
*IF(CARDP(1)) EQ.DIGIT(2), AND.CARDP(2), EQ.DIGIT(6)) GO TO 70
326 WRITE(6,126)
126 *YOUR ANSWER IS INCORRECT. THIS PROGRAM SUMS THE'//

```

```

*ELEMENTS OF THE TABLE ON THE MAJOR DIAGONAL (UPPER LEFT TO LOWER
  70 *RIGHT).//. THE SUM=15.0.//)
127 *WRITE(6,127)
127 *FORMAT(0,5X, ' YOU HAVE SEEN HOW SUBSCRIPTED VARIABLES ARE TO:
*USED TO SET UP LISTS(VECTORS) AND TABLES(MATRICES). HOWEVER TO:
*USE LISTS AND TABLES IN A PROGRAM, YOU MUST DIMENSION THE
*MAXIMUM SIZE OF YOUR LIST OR TABLE, SO THAT ENOUGH SPACE WILL
*BE ALLOCATED IN THE COMPUTER MEMORY. THIS DIMENSIONING IS DONE:
*WITH THE DIM STATEMENT.//)
  328 *READ(5,101,END=328)CARD
128 *WRITE(6,128)
128 *FORMAT(0,5X, ' C. DIM STATEMENT
*THE DIM STATES TELL THE COMPUTER THE MAXIMUM SIZE OF VECTORS.//
*AND TABLES THAT WILL BE USED IN YOUR PROGRAM. THE DIM STATEMENT
*// MUST APPEAR IN THE PROGRAM BEFORE ANY REFERENCE IS MADE.//
*TO THE LIST OR TABLE. IN GENERAL PRACTICE THE DIM STATEMENT
*IS USUALLY THE FIRST STATEMENT IN THE PROGRAM. THE FORM OF THE
*DIM STATEMENT IS: //1CX, ' DIM <<LIST VARIABLE>> ( <<SIZE>> )'//1
*1CX, ' DIM <<TABLE VARIABLE>> ( <<SIZE,SIZE>> )'//1
  329 *READ(5,101,END=329)CARD
129 *WRITE(6,129)
129 *FORMAT(0,5X, ' THE LIST AND TABLE VARIABLES ARE SUBSCRIPTED VARIABLE
*// AS DEFINED EARLIER. THE SIZE IS AN UNSIGNED INTEGER IN //
*PARENTHESES WHICH DENOTES THE MAXIMUM SIZE OF THE LIST OR TABLE.//
*THE DIM STATES MAY CONTAIN A NUMBER OF LISTS OR TABLES.//
*WITH THEIR SIZES SEPARATED BY COMMAS, FOR EXAMPLE: //5X,
*DIM A(6),X(10,3),Z(14,21)')//)
  330 *READ(5,101,END=330)CARD
130 *WRITE(6,130)
130 *FORMAT(0,5X, ' IF YOU TRY TO REFERENCE AN ELEMENT IN A LIST
*FOR TABLE BEYOND THE MAXIMUM SIZE IN THE DIM STATEMENT YOU WILL:
*GET AN ERROR. THE MAXIMUM SIZE LIST(VECTOR) OR TABLE(MATRIX)
*ALLOWED BY THE CAL-BASIC COMPILER IN ANY ONE PROGRAM IS A TOTAL:
*OF 1600 COMPUTER MEMED SOME QUESTIONS:
*YOU WILL NOW BE ASKED
*1.) ARE THE FOLLOWING DIM STATEMENTS CORRECT, REPLY: YES OR NO.//1
*1CX, ' DIM A(10)')//)
  75 *READ(5,101,END=331)CARD
  75 *CALL CRUNCH(1,LENGTH)
  75 *IF(CAROP(1).EQ.ASTRSK) GO TO 75
  75 *IF(CAROP(1).EQ.ALPHA(14)) GO TO 80
  331 *WRITE(6,131)
  331 *FORMAT(0,5X, ' THE STATEMENT IS INCORRECT. SUBSCRIPTED VARIABLES:
131 *ARE A SINGLE LETTER FOLLOWED BY ONE OR TWO SUBSCRIPTS IN:
*PARENTHESES.//)
  80 *WRITE(6,132)
  80 *FORMAT(0,5X, ' DIM X(500),B(1,10),C(5,5,5)')//)
  132 *READ(5,101,END=333)CARD
  85 *WRITE(6,133)

```

```

CALL CRUNCH(1,ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 85
IF(CARDP(1).EQ.ALPHA(14)) GO TO 90
333 WRITE(6,133)
133 *ONLY SINGLE OR DOUBLE SUBSCRIPTS ARE ALLOWED.*/)
90 WRITE(6,134)
134 *IN THIS LESSON YOU HAVE LEARNED HOW TO USE THE FOR/NEXT STATEMENT.
**// AND HOW TO MANIPULATE LISTS(VECTORS) AND TABLES(MATRICES).
**BY USING SUBSCRIPTS AND THE DIM STATEMENT YOU NOW
**HAVE ALL THE TOOLS TO BEGIN WRITING SOPHISTICATED PROGRAMS.
**AND AS YOU WRITE MORE COMPLICATED PROGRAMS YOU WILL FIND A
**NEED FOR SUBROUTINES. SUBROUTINES ARE COMMONLY USED IN YOUR PROGRAM.
**SEGMENTS THAT ARE USED OVER AGAIN IN OTHER PARTS OF YOUR PROGRAM.
**SUBROUTINES ALLOW YOU TO BRANCH TO THE COMMONLY USED SEGMENT.
**AND THEN RETURN TO WHERE YOU WERE AND CONTINUE EXECUTING.
**SUBROUTINE CAN BE CALLED FROM ANYWHERE IN YOUR PROGRAM.
**GOSUB STATEMENT ALLOWS SUBROUTINES IN BASIC, AND YOU WILL BE
**INTRODUCED TO IT IN THE NEXT LESSON.*/)
READ(5,101,END=335)CARD
335 WRITE(6,135)
135 *YOU WILL NOW BE GIVEN TWO OPTIONAL PROGRAMMING.
**PROBLEMS TO EXERCISE YOUR NEW TOOLS.
**NO REVERSE TO THE NUMBERS IN A 10-ELEMENT LIST.
**ALUES AND TEST YOUR PROGRAM BY PRINTING THE LIST.
**// DECENT CHECK THE FIRST ELEMENT OF THE LIST.
**HIS.// IS THE FIRST ELEMENT OF THE LIST.
**D.// GO AND COMPARE THE FIRST ELEMENT OF THE LIST.
**ATED.// UNTIL THERE ARE NO MORE EXCHANGES TO BE MADE.
**// THE EXCHANGES CAN BE MADE, AND WHEN THE COUNT EQUALS 0 THE
**ST IS IN ORDER.*/)
READ(5,101,END=336)CARD
336 WRITE(6,136)
136 *IF YOU WANT TO EXECUTE THESE PROBLEMS REPLY : YES, OTHERWISE NO.
95 *PERFORM CRUNCH(1,ILNGTH)
95 READ(5,101,END=337)CARD
**CALL CRUNCH(1,ILNGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 95
IF(CARDP(1).EQ.ALPHA(25)) CALL TEST1
337 WRITE(6,137)
137 *S REPLY : YES, IF YOU WOULD LIKE TO SEE A SOLUTION TO THE PROBLEM.
96 *READ(5,101,END=338)
**CALL CRUNCH(1,ILNGTH)

```

```

LES03340
IF(CARDP(1).EQ.ASTRSK) GO TO 96
IF(CARDP(1).EQ.ALPHA(14)) CALL EXIT
338 WRITE(6,138)
138 FORMAT(10,1)
* A LISTED.//5X, REM PROBLEM 1.//5X, REM PROGRAM TO REVERSE ELEMENTS IN
* SWAPPED.//5X, REM Y IS A TEMPORARY LOCATION FOR THE ELEMENT BEING
* LET X(1)=X(11-I).//5X, LET X(11-I)=Y.//5X, NEXT I.//5X, REM S.//5X, P
* ROBLEM 2.//5X, REM SORT LIST IN DECREASING ORDER.//5X, REM S.//5X, P
* THE EXCHANGE COUNTER, Y- IS A TEMPORARY LOCATION.//5X, DIM D(5)
* /5X, FOR I=1 TO 5.//5X, READ D(I).//5X, NEXT I.//5X, LET S=D(I)
* /5X, FOR I=1 TO N-1.//6X, IF D(I) GE D(I+1), THEN 16.//5X, LET S=D(I)
* /6X, LET D(I)=D(I+1).//6X, LET D(I+1)=S.//5X, LET S=1.//5X,
* /5X, IF S NE 0, THEN 5.//5X, PRINT, NUMBERS IN ORDER.//5X, FOR
* I=1 TO 5.//5X, PRINT D(I).//5X, NEXT I.//5X, END.//
CALL EXIT
END
LES03360
LES03370
LES03380
LES03390
LES03410
LES03420
LES03440
LES03460
LES03470
LES03490

```

THIS LESSON INTRODUCES SUBROUTINES AND RECURSION

```

COMMON
STACK(100), PROG(200), CARD(60), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRIB(10), LISTST(100), IYST(100),
PRT(2500), NERRS(10), NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PARTT, DECMAL, PLUS, CMINUS, SLASH, COMMA,
PARLFT, ASTRSK, BLANK
INTERP, EXERR, /
COMMON LES6, LES6, N1
REAL*8 LES6, LES6, N1
WRITE(6,100)
100 FCRMAT(10,10,5X, ** LESSON 6 **
* THIS LESSON WILL INTRODUCE YOU TO SUBROUTINES AND THE PROGRAMMING.//
* TECHNIQUE CALLED RECURSION.//5X, A SUBROUTINE ON MORE.//
* IN MANY PROGRAMS A BLOCK OF STATEMENTS MAY BE NEEDED ON STATEMENTS.
* THAN ONE OCCASION. SINCE THE REPEITION OF A BLOCK OF STATEMENTS.
* IS BURDEN A SUBROUTINE, A TECHNIQUE TO ECONOMIZE ON CODING IN SUBROUTINE.
* IS ://5X, STATEMENT NUMBER >> //5X, . . . //5X, . . . //5X,
* READ(5,101,1) RETURN.//
101 FCRMAT(10,10,1)
302 WRITE(6,102)
102 FCRMAT(10,10,5X, EXECUTION OF THE GOSUB STATEMENT CAUSES THE
* COMPUTER TO TRANSFER CONTROL TO THE STATEMENTS AFTER THE GOSUB.
* WHEN CONTROL IS TRANSFERRED TO THE STATEMENTS AT THAT TIME,
* IALLY, UNTIL A RETURN IS ENCOUNTERED, THE STATEMENT FOLLOWING THE GOSUB.
* IS RETURNED TO THE NEXT STATEMENT FOLLOWING THE STATEMENT.
* STATEMENT WHICH IS CALLED. THE SUBROUTINE. FOR EXAMPLE CONSIDER.//
LES00020
LES00030
LES00040
LES00050
LES00060
LES00070
LES00080
LES00090
LES00100
LES00110
LES00120
LES00140
LES00150
LES00160
LES00170
LES00180
LES00190
LES00200
LES00210
LES00220
LES00230
LES00240
LES00250
LES00260
LES00270
LES00280

```



```

** READ N./5X, IF N GT 0 THEN 20./5X, GO TO 50./3X, 20 PRINT N./5X
** GOSUB 100./5X, PRINT F./5X, GO TO 10./8X, 100 REM ITERATIVE
** SOLUTION./13X, LET F=1./13X, FOR I=1 TO N./14X, LET F=F*I./13X, LES00790
** NEXT I./10X, REM RETURN F=FACTORIAL(N)/8X, RETURN/3X, 50 REM
** FACTORIAL: IF N=0 THEN F(N)=1./5X, REM OTHERWISE, F(N)=LES00810
** N*(N-1)/5X, RESTORE/5X, PRINT, RESURSIVE SOLUTION./3X, 60 R
** READ N./5X, IF N GT 0 THEN 70./5X, GO TO 9999./3X, 70C PRINT N./5
** X, GO SUB 200./5X, PRINT F./6X, GO TO 60./8X, 200 REM RECURSIVE
** SOLUTION./13X, IF N GT 0 THEN 210./13X, LET F=1./13X, RETURN/8
** X, 210 LET N=N-1./13X, REM RECURSIVE CALL./13X, GOSUB 200./13X,
** LET N=N+1./13X, LET F=F*N./8X, RETURN/5X, DATA 2.5,8.6,-1./9
** 999 END./13X,
308 READ(5,101,END=308)CARD
108 WRITE(6,108)
** F(N) WAS MULTIPLIED BY ITSELF IN A LOOP FROM I=1 TO N, IN THE
** RECURSIVE SOLUTION OF ITS FINAL VALUE. WHEN N=0, F(N) IS DEFINED
** IN TERMS OF ITS FINAL VALUE. OTHERWISE,
** F(N)=N*(FACTORIAL OF N-1). SUBROUTINE CALLS RETURN IN RECURSION,
** IT HELPS TRACK OF WHERE THE SUBROUTINE CALLS RETURN IN RECURSION,
** G, THE ADDRESS OF THE GOSUB STATEMENTS, EVERY TIME A GOSUB
** IS ENCOUNTERED, PUT ITS ADDRESS ON TOP OF THE STACK (PUSHING
** DOWN ANYTHING PREVIOUSLY ON THE STACK). THEN EVERY TIME A RETURN
** IS ENCOUNTERED, RETURN TO THE TOP ADDRESS ON THE STACK (AND
** POP UP THE NEXT ADDRESS ON THE STACK).
309 READ(5,101,END=309)CARD
109 WRITE(6,109)
** TO FULLY UNDERSTAND THE CONCEPT OF RECURSION YOU
** SHOULD STEP THROUGH THE FACTORIAL PROBLEM BY HAND USING THE
** HELP OF THE STACK TO SEE HOW RECURSION WORKS IF YOU UNDERSTAND
** THE CONCEPT OF RECURSION YOU ARE READY TO SOLVE THIS PROBLEM
** /5X, LET X=1./5X, GOSUB 100./6X, PRINT X./5X, GO TO 9999./8X,
** LET X=X+1./13X, IF X GT 3 THEN 150./13X, GOSUB 100./8X, 150
** LET X=X+1./8X, RETURN./9999 END./13X, REPLY WITH VALUE OF X.
5 READ(5,101,END=310)CARD
** CALL CRUNCH(LENGTH)
IF(CARDP(1).EQ.ASTRSK) GO TO 5
IF(CARDP(1).EQ.DIGIT(8).AND.CARDP(2).EQ.BLANK)GO TO 10
310 WRITE(6,110)
** YOUR ANSWER IS INCORRECT. CONSIDER THE FOLLOWING:
** TABLE OF VALUES FOR X, AND ITEMS IN THE STACK(1-FIRST GOSUB, 2-STACK
** COND GOSUB) : /6X, X=1, 10X, STACK./21X, 1./5X, X=2, 10X,
** /21X, 2./2X, 1./5X, X=3, 10X, STACK./21X, 2./2X, 1./5X, X=4,
** 1./5X, X=5, 10X, STACK./21X, 2./2X, 1./5X, X=6,
** 10X, STACK./21X, 1./5X, X=7, YOU SHOULD STEP THROUGH THIS
** EXAMPLE UNTIL YOU UNDERSTAND THE RECURSION TECHNIQUE. HOWEVER,
** KNOWING HOW TO USE RECURSION IS NOT A REQUIREMENT FOR

```

```

* HOW TO USE BASIC , IT IS ONLY A CLASSIC PROGRAMMING TECHNIQUE .*/LES01250
*) WRITE(6,111)LES01260
101 FFORMAT(10,1,5X,1, C. SUMMARY ARE A VALUABLE ADDITION TO YOUR REPERTOILES01270
111 *SUBROUTINES AND STATEMENTS . THEY SAVE BOTH PROGRAMMER TIME AND .*/5X,1LES01280
*FORM THE BASIC STATEMENTS . THAT YOU HAVE ALREADY LEARNED AND USED .*/LES01300
*THE FACILITY THAT YOU GAIN IN PROGRAMMING BY USING THE BASIC .*/LES01310
*LANGUAGE WILL DEPEND UPON HOW OFTEN YOU EXERCISE YOUR SKILLS .*/5XLES01320
*THE LAST LESSON WILL PROVIDE YOU WITH A BRIEF SUMMARY OF THE .*/LES01330
* BASIC LANGUAGE .*/LES01340
* READ(5,101,END=312)CARDLES01350
312 WRITE(6,112)LES01360
112 FFORMAT(10,1,5X,1, THE FOLLOWING PROBLEMS WILL TEST YOUR LATEST SKILLLES01370
*AND TAX FOR N EMPLOYEES. NET PAY=GROSS $ SALARY-RETIREMENT CONTRIBUION.*/LES01380
*USE SUBROUTINES TO CALCULATE :*/10X,1, A) RETIREMENT CONTRIBUION.*/LES01390
*15X,1, IF SALARY <$800, R=$0.00/15X,1, B) TAX.*/15X,1, IF SALARY <$2500, R=$10.*/LES01400
*600,1, IF SALARY >$2500, R=$20.*/15X,1, T=.03(SALARY).*/15X,1, THE DATA IS:*/5XLES01410
*EMPLOYEE,10X,1, OTHERWISE,10X,1, JONES,15X,1,3000,1,77X,1,SMITH,15XLES01420
*5500,1,77X,1,BROWN,15X,1,475,1,77X,1,DALEY,15X,1,10,500,1,77X,1,BERRY,15XLES01430
*5X,1,4300,1,77X,1,2. WRITE A RECURSIVE PROGRAM TO SUM THE NUMBERS.*/LES01440
* FROM X TO Y. READ IN YOUR OWN DATA AND TEST YOUR PROGRAM .*/LES01450
* READ(5,101,END=313)CARDLES01460
313 WRITE(6,113)LES01470
113 FFORMAT(10,1,5X,1, IF YOU WANT TO EXECUTE THESE PROGRAMS NOW , THEN .*/LES01480
*REPLY : YES,END=313)CARDLES01490
20 CALL CRUNCH(1),EQ.ASTRSK)GO TO 20LES01500
IF(CARDP(1),EQ.ALPHA(14)) CALL EXITLES01510
CALL TEST1LES01520
WRITE(6,114)LES01530
114 FFORMAT(10,1,5X,1, IF YOU WOULD LIKE TO SEE A SOLUTION TO THE PROBLEMS.*/LES01540
*REPLY : YES,END=315)CARDLES01550
25 READ(5,101,END=315)CARDLES01560
CALL CRUNCH(1),EQ.ASTRSK)GO TO 25LES01570
IF(CARDP(1),EQ.ALPHA(14)) CALL EXITLES01580
WRITE(6,115)LES01590
315 FFORMAT(10,1,5X,1, PROBLEM 1.*/5X,1, REM ES=EMPLOYEE,S=GROSS SALARY,R=RETIREMENT,15X,1,LES01600
*IREMENT.*/5X,1, REM T=TAX,P=NET PAY.*/5X,1, READ N.*/5X,1, FOR I=1 TO N.*/LES01610
*/9X,1, READ ES,S.*/9X,1, GO SUB 20.*/9X,1, GOSUB 30.*/9X,1, P=S-R-T.*/9X,1,LES01620
*PRINT ES,P.*/5X,1, NEXT I.*/5X,1, GO TO 9999.*/9X,1,20 REM TAX CALLES01630
*ULATION.*/10X,1, IF S LT 600 THEN 40.*/13X,1, LET T=0.03*S.*/13X,1, GOLES01640
LES01650
LES01660
LES01670
LES01680
LES01690
LES01700

```

```

*TO 50./9X: '40 LET T=5./9X: '50 RETURN./9X: '30 REM RETIREMENT CALCULLES01730
*ATION./13X: ' IF S GE 2500 THEN 60./13X: ' IF S GE 800 THEN 70./13X: LES01740
* LET R=0./13X: ' GO TO 80./9X: '60 LET R=20./13X: ' GO TO 80./9X: '70 LES01750
* LET R=10./9X: '80 RETURN./5X: ' DATA 5./5X: ' DATA ' JONES: '3000./5
*X: ' DATA SMITH: '5500./5X: ' DATA ' BROWN: '475./5X: ' DATA ' DALE
*X: '10500./5X: ' DATA ' BERRY: '4300./5X: '9999 END./5X: ' PROBLEM 2./5
*X: ' REM SUM FROM X-->Y./6X: 'PRINT SUM FROM X-->Y./6X: 'LET S=
*C: '3X: '1 READ X: 'Y./5X: ' IF X=9999 THEN 9999./5X: ' GCSUB 10./5X: ' P
*RINT X: 'Y: 'S./5X: ' GO TO 1./9X: '10 REM SUM NUMBERS: RECURSIVELY./13X: LES01810
* IF X NE Y THEN 20./13X: ' LET S=X./13X: ' RETURN./9X: '20 LET X=X+1: LES01820
*./13X: ' GOSUB 10./13X: ' LET Y=Y-1./13X: ' LET S=S+Y./9X: ' RETURN./5 LES01830
*X: ' DATA 3.6./5X: ' DATA 1.10./5X: ' DATA 9999.1./5X: '9999 END./5 LES01840
*CALL EXIT
END

```

C C

THIS LESSON GIVES A COMPLETE SUMMARY OF BASIC LANGUAGE STATEMENTS

```

COMMON
STACK(100), PROG(2000), CARD(80), CARDP(80), ALPHA(48),
IAPTR, INPTR, IADATA(500), XNDATA(500), STRING(5),
DIGIT(10), IPRTB(10), LIST(100), ISTLST(100),
PRT(2500), NERRS, INST, NSTLST, DEBUG, DOLSGN, QUOTE,
EQUALS, PART, DECIMAL, PLUS, CMINUS, SLASH, COMMA,
PARLFT, IEXERR,
COMMON INTERP, LEXON7
REAL*8 LES7, LES7.N1
CALL ALLOC(LES7.N1)
WRITE(6,100)
FCRMAT(100,5X, ' ** LESSON 7 **

```

```

100 *THIS LESSON SUMMARIZES THE BASIC DEFINITIONS AND BASIC STATEMENTS.
*/. THAT YOU HAVE LEARNED IN CAI-BASIC .//5X. A. BASIC DEFINITION
*/. I. ITS: 0-->9
*/. DIGITERS: A-->Z
*/. SPECIAL CHARACTERS: ** * / + - ( ) = ' ' $ -->9999)
*/. STATEMENT: ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS ENCLOSED .//
*/. IN SINGLE: QUOTES. EG. 'HELP: '//'
READ(5,101)END=302)CARD
FORMAT(6,102)
101 FORMAT(6,102)
102

```

```

101
102
*/. NUMBERS: (LIMITED TO 9 DIGITS) EG. 5.7, 10./5X:
*/. INTEGERS: DIGITS WITH NO FRACTIONAL PART. EG. 5, 7, 10./5X:
*/. REAL: DIGIT WITH A FRACTIONAL PART. EG. 5.0, 7.31, -16.0.
*/. A NUMBER: MAY BE PRECEDED BY A SIGN (+, -), BUT IS ASSUMED TO .//
*/. POSITIVE: IF NONE IS GIVEN.
*/. VARIABLES: A SINGLE LETTER, OR A SINGLE LETTER .//
*/. SIMPLE VARIABLES: A SINGLE LETTER, OR A SINGLE LETTER .//

```



```

* 1'// 13. GOSUB << STATEMENT NUMBER >>
* //5X, 101. //5X, 101. //5X, 101. //5X, 101. //5X, 101.
* READ(5,101)END=306)CARD
306 WRITE(6,106)
106 FORMAT(106)
* HOWEVER, YOU ARE INVITED TO USE WHATEVER FACILITIES OF CAI-BASIC.
* THE OS/BATCH MODE (PUNCHING YOUR OWN CARDS AND HANDING THEM
* ACROSS THE COUNTER TO BE RUN), YOU SHOULD GET THE BASIC MANUAL.
* TECHNICAL NOTE NR. 3211-12, IN ROOM I-147 TO FIND THE PROPER.
* JCB CONTROL CARDS REQUIRED. GOOD LUCK WITH THE COMPUTER, AND.
* REMEMBER IT ONLY DOES WHAT YOU TELL IT TO DO - GIGO (GARBAGE IN,
* GARBAGE OUT).
* CALL EXIT
END

```

LES02030

APPENDIX C

13.13.46 START
EXECUTION BEGINS...

HI , WELCOME TO CAI-BASIC . THERE ARE ONLY A FEW
SIMPLE RULES TO REMEMBER IN ORDER TO HAVE A SUCCESSFUL SESSION
ON THE TERMINAL WITH CAI-BASIC :

1. WHEN ASKED FOR A RESPONSE , TYPE IN THE CORRECT REPLY AND
HIT THE CARRIAGE RETURN KEY ON THE RIGHT SIDE OF THE KEYBOARD.

2. IF YOU MAKE A TYPING ERROR WHILE MAKING ANY RESPONSE
OR INPUT , TYPE IN FOUR DOLLAR SIGNS (\$\$\$\$) AFTER THE ERROR OR
ANYWHERE ON THAT INPUT LINE AND HIT CARRIAGE RETURN .THE
ENTIRE LINE WILL THEN BE IGNORED AND YOU CAN TYPE IN THE CORRECT
INPUT OR RESPONSE .

3. IF AT ANY POINT IN THE SESSION YOU WANT TO STOP THE SESSION
TYPE IN THE WORD 'QUIT' AS SOON AS YOU ARE ASKED FOR THE NEXT
RESPONSE ,HIT CARRIAGE RETURN,THEN HIT ATTN KEY AND TYPE LOGOUT .

4. DURING YOUR TERMINAL SESSION CAI-BASIC WILL HALT OCCASIONALLY
TO LET YOU READ A SEQUENCE OF INFORMATION . WHEN YOU ARE
READY TO CONTINUE , TYPE IN 'GO' , AND HIT CARRIAGE RETURN

5. DURING YOUR TERMINAL SESSION YOU MAY NOTICE THAT
THE TYPING IS NOT ALWAYS PERFECT . SOME DAYS THE COMPUTER IS
NOT UP TO PAR AND YOU WILL HAVE TO ADJUST TO THE MINOR IRRITANT

IF YOU ONLY WANT TO EXECUTE PROGRAMS AT THIS TIME THEN
REPLY : YES ; OTHERWISE REPLY : NO .

no

IF THIS IS YOUR FIRST SESSION WITH CAIBASIC, THEN
REPLY: YES ;OTHERWISE REPLY: NO .

yes

CAIBASIC IS A PROGRAM TO TEACH YOU THE
FUNDAMENTALS OF A PROGRAMMING LANGUAGE.THE LANGUAGE TO BE LEARNED
IS BASIC ; A SIMPLE LANGUAGE FOR THE USER WHO HAS LITTLE
KNOWLEDGE OF COMPUTERS AND WHOSE PRIMARY INTEREST IS IN OBTAINING
RESULTS.

THE SIMPLICITY OF THE BASIC LANGUAGE AND ITS RANGE
OF CAPABILITIES SHOULD ALLOW YOU TO LEARN THE LANGUAGE AND

WRITE PROGRAMS IN A MINIMAL AMOUNT OF TIME.

THE REFERENCE TEXTS RECOMMENDED FOR CAIBASIC ARE :

1. BASIC LANGUAGE MANUAL , TN # 0211-12 APRIL 1971
(FREE UPON REQUEST IN 1-247)

2. INTRODUCTION TO COMPUTING THROUGH THE BASIC LANGUAGE , R.L. NOLAN
(BOOKSTORE / MAIN LIBRARY)

3. BASIC PROGRAMMING , V.C. HARE
(COMPUTER CENTER LIBRARY)

*** AFTER YOU HAVE FINISHED READING AN INPUT , TYPE IN GO AND HIT THE RETURN
AND THE PROGRAM WILL CONTINUE ***

80

DURING YOUR TERMINAL SESSION YOU WILL BE LEARNING
THE STRUCTURE OF THE LANGUAGE BASIC. THE INSTRUCTION SET
CONTAINS 7 LESSONS AND YOU MAY PROCEED THROUGH THE LESSONS AT
YOUR OWN SPEED.

THE LESSONS ARE AS FOLLOWS :

LESSON 1 PROGRAM FORMAT AND BASIC DEFINITIONS
LESSON 2 REMARKS, INPUT/OUTPUT AND DATA
LESSON 3 ASSIGNMENT STATEMENTS AND BUILT IN FUNCTIONS
LESSON 4 BRANCHING
LESSON 5 LOOPING AND SUBSCRIPTED VARIABLES
LESSON 6 SUBROUTINES AND RECURSION
LESSON 7 SUMMARY OF BASIC STATEMENTS

** WHEN YOU ARE READY TO CONTINUE , TYPE IN GO AND HIT
THE CARRIAGE RETURN ***

80

IN EACH LESSON YOU WILL BE GIVEN INSTRUCTION
SEQUENCES AND THEN YOU WILL BE ASKED QUESTIONS TO SEE IF YOU
UNDERSTOOD THE INSTRUCTIONS. THE QUESTIONS WILL BE OF VARIOUS
TYPES: MULTIPLE CHOICE, TRUE/FALSE, ACTUAL PROGRAM STATEMENTS, ETC.
YOU WILL BE PROMPTED FOR YOUR ANSWER, AND WHEN READY TYPE IN
OUR RESPONSE AND HIT THE RETURN KEY.

IF YOU KEEP THE TELETYPE OUTPUT FROM YOUR TERMINAL SESSION
YOU WILL HAVE A READY REFERENCE FOR FUTURE USE

80

*** LESSON 1. ***

THIS INSTRUCTION SET WILL INTRODUCE YOU TO THE STRUCTURE OF BASIC LANGUAGE STATEMENTS, THE RULES FOR VARIABLES AND NUMBERS, AND THE SYMBOLS FOR ARITHMETIC OPERATIONS.

***DONT FORGET TO TYPE GO AND HIT RETURN WHEN READY TO CONTINUE . ***

80

A. PROGRAM STRUCTURE

THE BASIC LANGUAGE, AND ALL OTHER LANGUAGES, HAS A SPECIFIED STRUCTURE. EACH BASIC STATEMENT HAS A REQUIRED FORM WITH POSSIBLY ONE OR MORE VARIATIONS. THE FOLLOWING EXAMPLE WILL ILLUSTRATE SOME SIMPLE BASIC STATEMENTS IN THE PROPER PROGRAM STRUCTURE

```
REM PROGRAM TO COMPUTE GAS MILAGE
REM M=MILES TRAVELED, G=GAS USED
READ M,G
LET T=M / G
PRINT 'MILES TRAVELED','GAS USED',' MILES/GAL'
PRINT M,G,T
DATA 500,25
END
```

OUTPUT PRODUCED IS :

MILES TRAVELED	GAS USED	MILES/GAL
500	25	20

80

AS YOU CAN SEE FROM THE ABOVE SAMPLE, THE PROGRAM STRUCTURE CONSISTS OF BASIC LANGUAGE STATEMENTS FOLLOWED BY AN END STATEMENT. THE WORDS: REM, READ, LET, PRINT, DATA AND END ARE KEY WORDS THAT MAKE UP A BASIC STATEMENT.

80

MOST OF THE KEY WORDS USED IN THE BASIC STATEMENTS ARE SELF-EXPLANATORY:

```
REM  ALLOWS REMARKS/COMMENTS
READ M,G  ASSIGNS NUMBERS IN THE DATA STATEMENT TO THE
           VARIABLES M AND G
LET  ASSIGNS THE RESULT OF M DIVIDED BY G INTO VARIABLE T
```

PRINT 'STRING' CAUSES THE STRING IN SINGLE QUOTES TO BE
PRINTED LITERALLY
PRINT M,G,T PRINTS THE CURRENT VALUE OF THE VARIABLES M,G,T
END TELLS THE COMPUTER THAT THE INPUT PROGRAM IS TO BE
EXECUTED

80

IF AT THIS POINT YOU WOULD LIKE TO RUN THE
SAMPLE PROGRAM TO GAIN SOME CONFIDENCE IN THE COMPUTER AND ITS
ABILITY TO PROVIDE SPEEDY RESULTS, THEN REPLY : YES ; OTHER-
WISE REPLY : NO AND THE INSTRUCTION WILL CONTINUE .

no

B. PROGRAM FORMAT

A BASIC PROGRAM CONSISTS OF A SEQUENCE OF BASIC STATEMENTS , ONE
STATEMENT PER INPUT LINE , FOLLOWED BY AN END STATEMENT .
BECAUSE NO BASIC STATEMENT MAY BE LONGER THAN ONE INPUT LINE (80 SPACES) ,
THERE IS NO PROVISION FOR CONTINUING STATEMENTS FROM ONE LINE
TO THE NEXT . HOWEVER ; YOU MAY SPACE THE INPUT LINE
AS DESIRED FOR READABILITY SINCE THE COMPUTER IGNORES BLANKS IN BASIC .

80

1. STATEMENT NUMBERS

EACH BASIC STATEMENT MAY HAVE AN OPTIONAL
STATEMENT NUMBER PRECEDING IT FOR IDENTIFICATION PURPOSES .
THIS STATEMENT NUMBER MUST BE AN INTEGER BETWEEN 1 -> 9999
FOR EXAMPLE : 12 READ M,G

2. KEY WORDS

THE KEY WORDS THAT MAKE UP A BASIC STATEMENT (REM , READ , LET , ETC ,)
ARE SPECIAL TERMINAL SYMBOLS THAT ARE RECOGNIZED BY THE COMPUTER
AND FOR THIS REASON THEY MUST BE SPELLED CORRECTLY AND ONLY
USED IN BASIC STATEMENTS .

THE END STATEMENT INDICATES THAT THE INPUT PROGRAM IS
COMPLETED AND THAT PROGRAM EXECUTION IS TO BEGIN . THE 'END'
STATEMENT IS ALWAYS THE LAST STATEMENT IN A PROGRAM .

80

YOU WILL NOW BE ASKED A FEW SIMPLE QUESTIONS ABOUT
WHAT YOU HAVE JUST LEARNED .

1.) IS THIS A LEGAL BASIC PROGRAM(REPLY : YES OR NO)??

REM ONE LINE DO NOTHING PROGRAM
END

yes

YES , THE SIMPLEST BASIC PROGRAM CONSISTS OF JUST AN 'END' STATEMENT.

2.) WHICH OF THE FOLLOWING BASIC STATEMENTS IS
IN THE PROPER FORMAT :

- A.) 15 LET T=M/G
- B.) 15LETT=M/G
- C.) 15 L E T T = M / G

REPLY A , d , C OR ALL .

all

ALL OF THE ABOVE BASIC STATEMENTS ARE CORRECT BECAUSE
SPACES ARE DISREGARDED . NOTE THAT STATEMENT B.) , ALTHOUGH
CORRECT IS CONFUSING TO READ . BLANKS AND INDENTATIONS MAKE A
PROGRAM EASY TO READ .

C. ALPHA-NUMERIC CHARACTERS

ALPHA-NUMERIC CHARACTERS ARE THE LEGAL CHARACTERS , DIGITS , AND
SPECIAL CHARACTERS THAT CAN BE USED IN BASIC .

1. CHARACTERS

CHARACTERS CONSIST OF THE LETTERS IN THE ALPHABET A-->Z

2. DIGITS

DIGITS ARE THE SINGLE NUMBERS 0-->9

3. SPECIAL CHARACTERS

THE SPECIAL CHARACTERS ARE * * . / + - () = ' , %

4. STRING

A STRING IS ANY LIST OF ALPHA-NUMERIC CHARACTERS
ENCLOSED IN 'SINGLE' QUOTES . FOR EXAMPLE :
'THIS IS A STRING'

so

D. VARIABLES

IN BASIC THERE ARE THREE TYPES OF VARIABLES : SIMPLE , ALPHA AND SUBSCRIPTED . SUBSCRIPTED VARIABLES WILL BE COVERED IN LESSON 5 .

1. SIMPLE VARIABLES

SIMPLE VARIABLES ARE IDENTIFIED BY A SINGLE LETTER OR A SINGLE LETTER FOLLOWED BY A DIGIT BETWEEN 0-9 .

FOR EXAMPLE : A , A3 , Z0 , AND X ARE LEGAL VARIABLES ; BUT A26 , 9Z , ABC , AND X1Y ARE ILLEGAL VARIABLES . THEREFORE ; YOU HAVE 286 SIMPLE VARIABLES FOR USE IN YOUR PROGRAMS.

80

2. ALPHA VARIABLES

ALPHA VARIABLES ARE USED FOR ALPHA-NUMERIC MANIPULATIONS IN WHICH A GROUP OF ALPHA-NUMERIC CHARACTERS , CALLED A STRING , ARE REPRESENTED BY AN ALPHA VARIABLE . THE ALPHA VARIABLE CONSISTS OF A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN , \$. THE MAXIMUM LENGTH OF THE ALPHA-NUMERIC STRING ASSIGNED TO THE ALPHA VARIABLE IS 16 CHARACTERS.

FOR EXAMPLE : A\$, X\$, Z\$ ARE LEGAL ALPHA VARIABLES. A SIMPLE USE OF ALPHA VARIABLES FOLLOWS :

```
READ A$,B$
PRINT A$,B$
DATA 'MONDAY','21 JUNE'
END
```

THIS PROGRAM WILL PRODUCE THE OUTPUT :

MONDAY 21 JUNE

80

E. NUMBERS

NUMBERS MAY BE EXPRESSED AS INTEGERS OR AS REALS ,AND A NUMBER WHETHER INTEGER OR REAL IS LIMITED TO 9 DIGITS , NOT INCLUDING DECIMAL POINT. A NUMBER IS ASSUMED POSITIVE UNLESS IT IS PRECEDED BY A - SIGN .

1. INTEGERS

INTEGERS ARE NUMBERS WITH NO FRACTIONAL PART ,IE. 3 , 15

2. REALS

REAL (FIXED-POINT) NUMBERS HAVE A DECIMAL POINT AND A FRACTIONAL PART , IE. 3.0 , 15.31, 729.1 , 0.0

80

F. EXPRESSIONS

AN EXPRESSION MAY BE A SINGLE CONSTANT OR VARIABLE , OR AN ARITHMETIC EXPRESSION . ARITHMETIC EXPRESSIONS ARE FORMED BY USING OPERATORS AND PARENTHESIS .

1. OPERATORS

NUMBERS AND SIMPLE VARIABLES MAY BE COMBINED INTO ARITHMETIC EXPRESSIONS BY USING ONE OR MORE OF THE ARITHMETIC OPERATORS :

- EXPONENTIATION
- MULTIPLICATION
- / DIVISION
- + ADDITION
- SUBTRACTION

IN WRITING AN EXPRESSION , EACH ARITHMETIC OPERATION MUST BE SHOWN. NO TWO OPERATIONS MAY BE ADJACENT , NOR MAY TWO NUMBERS OR VARIABLES BE ADJACENT IN AN EXPRESSION. FOR EXAMPLE: $2+2$, A/B , $B+C$, $X*Z$, AND $Y-3.0$ ARE LEGAL EXPRESSIONS.

80

2. PARENTHESIS

PARENTHESIS MAY BE USED TO GROUP EXPRESSIONS AND TO CONTROL THE ORDER IN WHICH AN ARITHMETIC EXPRESSION IS EVALUATED . THE NORMAL HIERARCHY OF OPERATORS IS :

1. ••
2. • AND /
3. + AND -

IF PARENTHESIS ARE USED , THE EXPRESSION WITHIN A PARENTHESIS PAIR IS EVALUATED FIRST . PARENTHESIS IN EXPRESSIONS MUST ALWAYS OCCUR IN PAIRS , IE. $5 * (4 + 3)$, $(5*(4+3))$ ARE EQUIVALENT EXPRESSIONS HAVING THE VALUE 35 .

OPERATIONS ARE EVALUATED ACCORDING TO THE OPERATORS HIERARCHY IF TWO OPERATORS OF THE SAME HIERARCHY OCCUR IN AN EXPRESSION EVALUATION IS FROM LEFT TO RIGHT . FOR EXAMPLE: $2*4 + 6/3$

IN THIS EXPRESSION $2*4$ WILL BE EVALUATED FIRST , THEN $6/3$, AND THEN THE TWO RESULTS WILL BE ADDED .

80

YOU WILL NOW BE ASKED A FEW SIMPLE QUESTIONS ABOUT
WHAT YOU HAVE JUST LEARNED .

1.) $4 + 6 / 2$ HAS THE VALUE (REPLY WITH VALUE)

5

YOUR ANSWER IS INCORRECT . THE EXPRESSION IS EVALUATED AS FOLLOWS :

$4 + 6 / 2 \rightarrow 4 + 3 \rightarrow 7$

2.) $(4 + 6) / 2$ HAS THE VALUE (REPLY WITH VALUE)

5

3.) $((4 + 6) / 2) * 2$ HAS THE VALUE (REPLY : VALUE)

25

F. SUMMARY

THIS CONCLUDES THE INSTRUCTION SET FOR LESSON 1.
YOU WILL NOW BE GIVEN A SAMPLE BASIC PROGRAM AND ASKED TO FIND
THE MISTAKES IN IT , CONCERNING WHAT YOU HAVE LEARNED IN THIS
LESSON :

```
1 REM REVIEW PROGRAM
2 READ A1,B2,Z1,$Y
3 LET A1=B2(A1+Z1)
4 LET B2=((A1+3)+Z1**2)
5 LET Z1=3.141592763
6 PRINT 1A,B2,Z1
7 DATA 5.0,3,3.1416,'FINI'
```

AFTER LOOKING AT THE SAMPLE PROGRAM , YOU WILL BE ASKED QUESTIONS
ABOUT EACH STATEMENT ,

80

1.) LINE 2 CONTAINS A READ STATEMENT FOLLOWED
BY A LIST OF VARIABLES . ANY ERRORS (REPLY: YES OR NO)?

yes

INPUT CORRECTION TO THE INCORRECT VARIABLE ONLY .

yes

2.) LINE 3 CONTAINS A LET STATEMENT FOLLOWED
BY A1 = EXPRESSION . ANY ERRORS (REPLY : YES OR NO) ?

yes

INPUT CORRECTION TO ILLEGAL EXPRESSION ; EVERYTHING
AFTER = SIGN .

b2*(a1+z1)

3.) LINE 4 IS SIMILAR TO LINE 3 . ANY ERRORS
REPLY YES OR NO) ?

no

4.) LINE 5 CONTAINS A LET STATEMENT FOLLOWED
BY A NUMBER . ANY ERRORS (REPLY : YES OR NO) ?

no

THE NUMBER CONTAINS 10 DIGITS AND THE MAXIMUM ALLOWED
IS 9 DIGITS.

5.) LINE 6 CONTAINS A PRINT STATEMENT FOLLOWED
BY A LIST OF VARIABLES . ANY ERRORS (REPLY : YES OR NO)?

yes

INPUT CORRECTION TO THE INCORRECT VARIABLE ONLY .

a1

THERE ARE NO ERRORS IN LINE 7 . IS THE PROGRAM
READY TO EXECUTE (ASSUMING ABOVE ERRORS CORRECTED)
(REPLY : YES OR NO)

yes

THE PROGRAM FORMAT REQUIRES THAT AN END STATEMENT

BE THE LAST STATEMENT OF THE PROGRAM . THUS THIS SAMPLE PROGRAM
WOULD NOT EXECUTE .

THIS CONCLUDES THE REVIEW QUESTIONS
FROM LESSON 1 .

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?
(REPLY : YES OR NO)

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION
WILL CONTINUE .

no

EXECUTION BEGINS...

*** LESSON 2. ***

THIS INSTRUCTION SEQUENCE WILL INTRODUCE YOU TO THE BASIC LANGUAGE STATEMENTS REM, PRINT, READ, AND DATA. USING THESE STATEMENTS YOU WILL BE ABLE TO CONSTRUCT AND EXECUTE ELEMENTARY PROGRAMS. THE FORM FOR EACH BASIC STATEMENT WILL INCLUDE:

'KEY WORD' << ELEMENTS OR LIST OF ELEMENTS SEPARATED BY COMMAS >>

THE CARET SYMBOLS '<< >>' DELINEATE THE LEGAL ITEMS THAT MAY FOLLOW THE KEYWORD AND MAKE UP THE BASIC STATEMENT.

80

A. REM

THE BASIC STATEMENT WHICH ALLOWS YOU TO INSERT REMARKS INTO YOUR PROGRAM IS IDENTIFIED BY THE KEY WORD 'REM'. REM IS A NON-EXECUTING STATEMENT WHICH MAY BE USED OPTIONALLY AT ANY PLACE IN YOUR PROGRAM TO INTRODUCE A PROGRAM NAME, TO EXPLAIN VARIABLES, TO DOCUMENT YOUR PROGRAM, ETC.

80

THE FORM FOR THE REM STATEMENT IS:

REM << ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS >>

THE REM STATEMENT IS IGNORED BY THE CAIBASIC COMPILER, AND IS ONLY FOR YOUR INFORMATION. FOR EXAMPLE:

REM PROGRAM TO COMPUTE INCOME TAX

80

B. PRINT

THE PRINT STATEMENT IS THE METHOD OF WRITING OUT THE RESULTS OF THE BASIC PROGRAM, TO DISPLAY VALUES OF VARIABLES, TO LABEL STATEMENT IS:

PRINT << EXPRESSION OR 'STRING',..., EXPRESSION OR 'STRING' >>

MULTIPLE ELEMENTS IN THE PRINT LIST ARE SEPARATED BY COMMAS.

80

1. PRINT << EXPRESSION >> WILL WRITE OUT THE CURRENT VALUE OF THE EXPRESSION, WHERE AN EXPRESSION AS DEFINED IN LESSON 1 WAS A NUMBER, A VARIABLE, OR AN ARITHMETIC

EXPRESSION THAT IS TO BE EVALUATED .

FOR EXAMPLE :

PRINT 1,A,B1,5**2

ASSUMING THAT A=10.0 , B=13.3 WOULD PRINT :

1	10.0	13.3	25
---	------	------	----

80

2. PRINT <<'STRING' >> WILL PRINT OUT ALL THE ALPHA-NUMERIC CHARACTERS OF THE STRING WITHIN 'SINGLE' QUOTES. THIS FORM IS USED FOR LABELING THE COMPUTER OUTPUT.

FOR EXAMPLE : PRINT 'THE ANSWER IS :'

PRODUCES THE RESULT :

THE ANSWER IS :

3. PRINT BY ITSELF IS USED TO SKIP A LINE ON THE COMPUTER OUTPUT .

80

THE COMPUTER OUTPUT SHEET IS DIVIDED INTO 8 ZONES EACH 15 COLUMNS WIDE . PRINT ZONES CAN BE SKIPPED BY PUTTING A BLANK IN THE PRINT LIST . FOR EXAMPLE :

PRINT A, B, ,X
OUTPUTS THE VALUE OF A IN THE FIRST ZONE , B IN THE SECOND , SKIPS THE THIRD ZONE , AND PUTS X IN THE FOURTH ZONE .

ALPHA VARIABLES AND 'STRINGS' MAY EXTEND OVER SEVERAL ZONES , BUT NUMERIC RESULTS ARE LEFT ADJUSTED IN THE SPECIFIED ZONE . IF MORE THAN EIGHT ITEMS OCCUR IN THE PRINT LIST , THE ITEMS WILL OVERFLOW AND BE PRINTED ON THE NEXT LINE .

80

USING THE PRINT AND END STATEMENT YOU NOW HAVE THE FACILITY TO EXECUTE YOUR FIRST PROGRAMS . FOR EXAMPLE :

```
REM PROGRAM TO COMPUTE THE SQUARE OF A NUMBER
PRINT'S SQUARED =' ,5**2
END
```

PRODUCES THE RESULT :

5 SQUARED = 25

YOU WILL NOW BE GIVEN TWO PROBLEMS TO SOLVE . YOU WILL ENTER

THE EXECUTION PHASE OF CAIBASIC WHERE YOU CAN RUN YOUR PROBLEMS AND THEN YOU WILL RETURN TO FINISH THE LESSON.

1) FIND THE SQUARE ROOT OF 5 SQUARED MINUS 4 TIMES 2 TIMES 2 .

2) FIND THE VALUE OF $3.1416(B^3)H/12$
WHERE $B=2.50$, $H=3.03$.

IF YOU WISH TO SKIP THESE PROBLEMS REPLY : YES
AND THE LESSON WILL CONTINUE.

YES

C. READ

THE READ STATEMENT IS THE METHOD WHICH PROVIDES INPUT TO THE PROGRAM . THE FORM OF THE READ STATEMENT IS :

READ << VARIABLE,....,VARIABLE >>

FOR EVERY VARIABLE IN THE READ LIST THERE MUST BE A CORRESPONDING ELEMENT IN A DATA STATEMENT . THE READ AND DATA STATEMENTS ARE USED TOGETHER TO ASSIGN INPUT VALUES TO PROGRAM VARIABLES.

WHEN THE READ STATEMENT IS EXECUTED , EACH VARIABLE IS ASSIGNED SUCCESSIVE NUMBERS FROM A STACK OF NUMERIC DATA OR SUCCESSIVE 'STRINGS' FROM A STACK OF ALPHA-NUMERIC DATA. AS EACH VARIABLE IS READ , IT TAKES THE TOP ELEMENT OF THE APPROPRIATE DATA STACK .

80

D. DATA

THE DATA STATEMENT IS A LIST OF INPUT NUMBERS OR 'STRINGS' THAT WILL BE ASSIGNED TO VARIABLES IN A READ STATEMENT . THE FORM OF THE DATA STATEMENT IS :

DATA << NUMBER OR 'STRING',...., NUMBER OR 'STRING' >>

THE 'STRING' OF ALPHA-NUMERIC CHARACTERS MUST BE ENCLOSED IN SINGLE QUOTES .

DATA STATEMENTS MAY BE PLACED ANYWHERE IN A PROGRAM , BUT THERE IS AN UPPER LIMIT OF 500 NUMERIC AND 500 ALPHA-NUMERIC DATA ELEMENTS FOR EACH PROGRAM .

80

WHEN THE FIRST DATA STATEMENT IS INTERPRETED BY THE CAIBASIC COMPILER A FIRST IN , FIRST OUT STACK IS FORMED FOR NUMERIC AND ALPHA-NUMERIC DATA . AS EACH NUMBER OR STRING IN A DATA LIST IS INTERPRETED , IT IS PLACED ON THE BOTTOM OF ITS RESPECTIVE DATA STACK . AS OTHER DATA STATEMENTS ARE LOCATED IN THE PROGRAM ITS ELEMENTS ARE PLACED ON THE BOTTOM OF THE PROPER STACK .

1.) EXAMPLE :

```
DATA 10.0,13.33,'J.E.SMITH',18
DATA 'Z.X. DOE',19
```

PRODUCES THE FOLLOWING DATA STACKS :

NUMERIC	ALPHA-NUMERIC
10.0	J.E.SMITH
13.33	Z.X. DOE
18	
19	

80

DURING EXECUTION OF READ STATEMENTS , AS THE VARIABLES IN THE READ LIST ARE ASSIGNED VALUES FROM THE TOP OF THE APPROPRIATE DATA STACK , THE DATA STACK IS DECREMENTED AND THE NEXT ELEMENT POPS UP .

2.) EXAMPLE :

```
READ A,B1,Z$
```

ASSUMING THAT THE DATA FROM EXAMPLE 1.) IS AVAILABLE , THE VARIABLES IN THE READ LIST ARE ASSIGNED VALUES AS FOLLOWS :

```
A <-- 10.0
B1 <-- 13.33
Z$ <-- J.E.SMITH
```

THE RESULTING DATA STACKS ARE AS FOLLOWS :

NUMERIC	ALPHA-NUMERIC
18	Z.X. DOE
19	

80

E. RESTORE , RESTORE\$

THE RESTORE , RESTORE\$ STATEMENTS ARE USED TO RETURN THE NUMERIC AND ALPHA-NUMERIC DATA STACKS TO THEIR ORIGINAL CONDITION SO

7.1
THAT THE DATA MAY BE USED AGAIN . THE FORM OF THE RESTORE STATEMENT
IS :

RESTORE (RESTORES NUMERIC DATA)

RESTORES (RESTORES ALPHA-NUMERIC DATA)

YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT THE READ , DATA AND
RESTORE STATEMENTS :

CONSIDER THE FOLLOWING STATEMENTS AS PART OF A BASIC PROGRAM:

```
DATA 2,5,7,16,'ANS='  
DATA 'CORRECT',25,'WRONG',0.0  
READ A,B3,I$,C1,D  
READ J$,E4,K$,X
```

80

1.) WHAT IS THE VALUE OF C1 ?? REPLY WITH VALUE

7

NOW CONSIDER THAT THE FOLLOWING STATEMENTS
WERE ADDED TO THE ABOVE PROGRAM :

```
RESTORE  
READ Z4,A
```

2.) WHAT IS THE VALUE OF Z4 ?? REPLY WITH VALUE .

16

THE CORRECT ANSWER IS Z4=2 . THE RESTORE COMMAND
RETURNS THE NUMERIC DATA STACK TO ITS ORIGINAL CONDITION, AND
THE READ STATEMENT ASSIGNS VALUES FROM THE TOP OF THE DATA
STACK TO THE VARIABLES IN THE READ LIST AS FOLLOWS :

```
Z4 <-- 2  
A <-- 5
```

F. SUMMARY

NOW THAT YOU HAVE SEEN HOW READ AND DATA STATEMENTS WORK TOGETHER
TO INPUT VALUES INTO YOUR PROGRAM , AND HOW THE PRINT STATEMENT
IS USED TO OUTPUT AND LABEL RESULTS YOU HAVE THE FACILITY
TO WRITE SIMPLE PROGRAMS USING INPUT DATA .

FOR EXAMPLE
((8**2)**.5) COULD BE WRITTEN IN SYMBOLIC FORM AND THE DATA

COULD BE READ IN AS FOLLOWS:

```
READ A,B
PRINT'ANS=',(( A**B)**.5)
DATA 8,2
END
```

IN THE NEXT LESSON YOU WILL BE SHOWN HOW TO USE ASSIGNMENT STATEMENTS
THIS WILL GIVE YOU GREATER FLEXIBILITY IN WRITING EXPRESSIONS
AND WILL ALLOW YOU TO DO ASSIGNMENTS SUCH AS :

$A=B**2-4*A*C$, AND $Z=(A**2+3)/A$, THEN BY SAYING PRINT A,Z
THE RESULTS OF BOTH EXPRESSIONS WOULD BE DISPLAYED.

so

YOU WILL NOW BE GIVEN SOME REPRESENTATIVE PROBLEMS
TO GIVE YOU A CHANCE TO EXERCISE YOUR NEW PROGRAMMING TOOLS

1.) WRITE A PROGRAM TO SOLVE THE EQUATION $X**2+10Y-24$
WHERE THE INPUT DATA IS $X=10$, $Y=3$.

2.) WRITE A PROGRAM TO SOLVE THE QUADRATIC EQUATION

$(-B+(B**2-4*AC)**.5)/2A$
INPUT DATA IS $A=2$, $B=5$, $C=2$.

IF YOU WISH TO WRITE AND EXECUTE THESE PROGRAMS NOW , REPLY : YES
AND YOU WILL ENTER THE CAIBASIC COMPILER ; OTHERWISE REPLY : NO
AND YOU WILL GO ON TO THE NEXT LESSON .

no

IF YOU DECIDE TO RUN THESE PROBLEMS LATER THE ANSWERS
WILL NOW BE GIVEN SO YOU MAY CHECK YOUR RESULTS :

1.) 106
2.) -.500

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?
(REPLY : YES OR NO)

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION
WILL CONTINUE .

no

EXECUTION BEGINS...

... LESSON 3. ...

THIS INSTRUCTION SET WILL INTRODUCE YOU TO ASSIGNMENT STATEMENTS AND BUILT-IN FUNCTIONS. THE LET STATEMENT AND THE 10 BUILT IN FUNCTION WILL ENABLE YOU TO EVALUATE AND ASSIGN VARIABLES TO COMPLEX ARITHMETIC EXPRESSIONS.

80

A. LET

THE LET STATEMENT IS AN ASSIGNMENT OR SUBSTITUTION COMMAND. IT CAUSES THE EVALUATION OF AN EXPRESSION TO BE SUBSTITUTED FOR THE CURRENT VALUE OF A VARIABLE. THE FORM OF THE LET STATEMENT IS:

LET << VARIABLE >> = << EXPRESSION >> OR
LET << VARIABLE >> = << VARIABLE >> = ... = << EXPRESSION >>

80

THERE MAY BE ANY NUMBER OF VARIABLE = VARIABLE IN THE FORM OF THE LET STATEMENT. AN EXPRESSION IS A NUMBER, VARIABLE, OR AN ARITHMETIC EXPRESSION.

WHEN THE LET STATEMENT IS EXECUTED THE EXPRESSION ON THE RIGHT SIDE OF THE EQUAL SIGN IS EVALUATED AND THE RESULTING VALUE IS ASSIGNED TO THE VARIABLE OR VARIABLES ON THE LEFT SIDE OF THE EQUAL SIGN. THE PREVIOUS VALUE ASSIGNED TO THE VARIABLE OR VARIABLES WILL BE LOST. FOR EXAMPLE:

LET A=12.3
LET B=14.4
LET A=B+25

THE VALUE OF THE VARIABLES A AND B IS NOW A=B= 25.

80

THE ONLY RESTRICTION ON THE USE OF VARIABLES IS THAT SIMPLE AND SUBSCRIPTED VARIABLES CAN ONLY BE ASSIGNED NUMERIC VALUES, AND ALPHA VARIABLES CAN ONLY BE ASSIGNED ALPHA-NUMERIC STRINGS. FOR EXAMPLE:

LET A=D4-X(5)=(3**2 - 6)/4
LET A\$=X\$='HELP'

```

      LET Y=(B - 4*A*C)**.5
      LET US='ANSWER ='
THE FOLLOWING ASSIGNMENT IS ILLEGAL :

```

```

      LET A=D$(3+4)

```

80

YOU WILL NOW BE ASKED QUESTIONS CONCERNING
WHAT YOU HAVE JUST LEARNED :

1.) REPLY WITH VALUE OF X IN BELOW PROGRAM .

```

LET A=4
LET B=6
LET C=3
LET X=A**2 - 4*B + 3*C
PRINT'ANSWER =' ,X
END

```

1

2.) REPLY WITH VALUE OF Z IN BELOW PROGRAM .

```

DATA 1,2,3,4,6
DATA'RIGHT ON',7,8
READ A,B,X
READ G$,Y,C,D
LET Z=X*Y
END

```

6

YOUR RESPONSE WAS INCORRECT . THE VARIABLES ARE
ASSIGNED VALUES AS FOLLOWS :

NUMERIC	ALPHA
A<-- 1	Y\$<-- RIGHT ON
B<-- 2	
X<-- 3	
Y<-- 4	
C<-- 6	
D<-- 7	

Z <-- X*Y ; Z <-- 7

3.) LET R= A*B/C-D
WHICH FORMULA DOES THIS STATEMENT REPRESENT ?

R
 EPLY WITH CORRECT LETTER
 A.) $R = (A+B)/(C-D)$
 B.) $R = A \cdot (B/C) - D$

b

8. BUILT-IN FUNCTIONS
 BUILT-IN FUNCTIONS ARE COMMONLY USED PROGRAMS ALREADY WRITTEN
 AND STORED IN THE CAIBASIC COMPILER FOR YOUR USE . THERE ARE
 FUNCTIONS TO FIND SQUARE ROOTS , LOGARITHMS , ABSOLUTE VALUES ,
 AND TRIGONOMETRIC VALUES . THE FORM FOR THE BUILT-IN FUNCTIONS IS :

FUNCTION NAME << (EXPRESSION) >>
 WHERE THE EXPRESSION IS ENCLOSED IN PARENTHESIS .

80

THE BUILT-IN FUNCTIONS AND DEFINITIONS ARE :

SQR(X)	--- SQUARE ROOT OF ARGUMENT (MUST BE POSITIVE)
ABS(X)	--- ABSOLUTE VALUE OF ARGUMENT
LOG(X)	--- NATURAL LOGARITHM OF ARGUMENT
EXP(X)	--- EXPONENTIAL FUNCTION , VALUE OF $2.718218 \cdot X$
INT(X)	--- INTEGER PART OF ARGUMENT IS RETURNED
SIN(X)	--- SINE OF THE ARGUMENT
COS(X)	--- COSINE OF ARGUMENT
TAN(X)	--- TANGENT OF ARGUMENT
ATN(X)	--- ARCTANGENT IN RADIANS OF ARGUMENT

IN ALL THE ABOVE BUILT-IN FUNCTIONS THE ARGUMENT IS ANY LEGAL EXPRESSION ;
 AND AS NOTED THE SQR FUNCTION REQUIRES A POSITIVE ARGUMENT ,
 THE TRIGONOMETRIC FUNCTIONS REQUIRE AN ARGUMENT VALUE IN RADIANS .

80

THE BUILT-IN FUNCTIONS ARE USED BY SIMPLY
 CALLING THEM WITH THE APPROPRIATE FUNCTION NAME AND ARGUMENT .
 THESE BUILT-IN FUNCTIONS ARE CONSIDERED TO BE EXPRESSIONS ,
 AND THEY MAY BE USED ANY PLACE WHERE AN EXPRESSION IS LEGAL ,
 FOR EXAMPLE :

LET Z=SQR(ABS(-5)) IS A CORRECT USE OF BUILT IN FUNCTIONS.

80

THE FOLLOWING PROGRAM IS A EXAMPLE OF HOW TO
USE BUILT-IN FUNCTIONS :

```
REM PROGRAM TO COMPUTE SQUARE ROOT AND LOGARITHMS
READ A
LET Y=SQR(A)
LET Z=LOG(A)
PRINT 'SQUARE ROOT=',Y,'LOG=',Z
DATA 4,6,8
END
```

PRODUCES RESULT
SQUARE ROOT= 2.0 LOG= 1.386

80

YOU WILL NOW BE ASKED QUESTIONS CONCERNING
WHAT YOU HAVE JUST LEARNED :

1.) IS THE FOLLOWING STATEMENT LEGAL
REPLY : YES , OR NO.

```
LET Z1=4*SQR(3.0 * EXP(9.3))
```

yes

2. IS THE FOLLOWING SEQUENCE OF PROGRAM STATEMENTS LEGAL
REPLY: YES OR NO.

```
LET B=-9
LET X=SQR(B)
...
...
...
```

no

NO , THE SQUARE ROOT OF A NEGATIVE NUMBER IS AN
UNDEFINED OPERATION . IN THE NEXT LESSON YOU WILL BE SHOW A
BASIC STATEMENT FOR TESTING AND BRANCHING TO ANOTHER SEGMENT
OF THE PROGRAM IF THE TEST IS TRUE . FOR EXAMPLE THE ABOVE
PROGRAM SEQUENCE MIGHT BE ALTERED AS FOLLOWS :

```
LET B=-9
IF B LT 0 THEN 100
50 LET X=SQR(X)
...
```

```

      ...
100 REM  NEGATIVE ARGUMENT
LET B=ABS(B)
GO TO 50
      ...

```

THIS PROGRAM SEQUENCE TESTS FOR A NEGATIVE ARGUMENT , IF TRUE IT BRANCHES TO STATEMENT NUMBER 100 , MAKES THE ARGUMENT POSITIVE AND BRANCHES BACK TO STATEMENT 50 TO COMPLETE THE PROGRAM .

so

C. SUMMARY

WITH THE LET STATEMENT AND BUILT-IN FUNCTIONS , PLUS THE PREVIOUS BASIC STATEMENTS (REM , READ , DATA , PRINT) , YOU ARE FAST GAINING AN EFFECTIVE REPERTOIRE FOR PROGRAMMING USE . IN THE NEXT LESSON YOU WILL LEARN HOW TO SET UP LOOPS IN A PROGRAM SO THAT THE MAIN BODY OF A PROGRAM MAY BE EXECUTED AS OFTEN AS DESIRED . AS YOU ARE DOING YOUR REVIEW PROBLEMS , THINK ABOUT HOW YOU COULD SET UP A LOOP TO READ IN ANY AMOUNT OF DATA , PROCESS IT AND THEN HALT FOR SOME TEST CONDITION .

so

THE FOLLOWING REVIEW PROBLEMS WILL EXERCISE YOUR PROGRAMMING SKILLS TO DATE :

1.) WRITE A PROGRAM TO COMPUTE THE PRESENT WORTH OF AN INVESTMENT FOR SOME NUMBER OF YEARS HENCE . THE FORMULA IS :

$$P = S / ((1 + I)^N)$$

WHERE P IS THE PRESENT WORTH OF A PAYMENT S IN N YEARS HENCE AT AN INTEREST RATE OF I . FOR DATA USE I=.08 , S=5000 , N=20 .

2.) WRITE A PROGRAM TO FIND SIDES A , AND C OF A TRIANGLE USING THE LAW OF SINES FORMULA :

$$A / \sin(A) = B / \sin(B) = C / \sin(C)$$

WHERE THE NUMERATOR IS THE SIDE AND THE DENOMIATOR IS THE SINE OF THE ANGLE , THE CONVERSION FACTOR FROM DEGREES TO RADIANS IS:

$$1 \text{ DEGREE} = (3.1416/180) \text{ RADIANS} . \text{ THE DATA FOR THE PROGRAM IS :}$$

SIDE B=34.91 IN. ANGLE A=98.71 DEG.

ANGLE B=49.97 DEG. ANGLE C=31.32 DEG.

IF YOU WISH TO RUN THESE PROGRAMS NOW , THEN REPLY : YES
OTHERWISE REPLY : NO.

no

THE ANSWERS TO THE PROBLEMS WILL NOW BE GIVEN SO
THAT YOU MAY CHECK YOUR RESULTS LATER :

- 1.) \$1072.74
- 2.) SIDE A=45.06 IN. AND SIDE C=23.69 IN.

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?
(REPLY : YES OR NO)

yes
R; T=4.84/16.96 14.03.44

EXECUTION BEGINS...

*** LESSON 4 ***

THIS INSTRUCTION SEQUENCE WILL COVER BRANCHES . AS YOU HAVE SEEN FROM PREVIOUS PROGRAMS , A PROGRAM'S EXECUTION USUALLY TAKES A DIRECT ROUTE FROM THE FIRST TO THE LAST STATEMENT . THE CONDITION THAT ALLOWS DETOURS TO OCCUR IN PROGRAMS IS CALLED BRANCHING . THERE ARE TWO TYPES OF BRANCHING : UNCONDITIONAL AND CONDITIONAL .

80

A. UNCONDITIONAL BRANCHES

AN UNCONDITIONAL BRANCH IS AN IMPERATIVE TRANSFER OF CONTROL FROM ONE POINT IN A PROGRAM TO ANOTHER . THERE ARE TWO FORMS OF THE UNCONDITIONAL BRANCH : THE 'GO TO' AND THE 'COMPUTED GO TO'

1.) GO TO << STATEMENT NUMBER >>

THIS COMMAND TRANSFERS PROGRAM CONTROL DIRECTLY TO THE STATEMENT NUMBER AND CONTINUES EXECUTION FROM THAT POINT . THE 'GO TO' IS USED FOR FORMING LOOPS IN A PROGRAM .

80

A SAMPLE LOOP FOLLOWS :

```
REM PROGRAM TO COMPUTE PRESENT WORTH
REM P=INVESTMENT,S=PRINCIPAL,I=INTEREST RATE,N=NR.YEARS
PRINT'INVESTMENT','PRINCIPAL','INTEREST','NR.YEARS'
10 READ S,I,N
LET P=S*(1/((1+I)**N))
PRINTP,S,I,N
GO TO 10
DATA 5000,.08,20,5000,.08,10,5000,.06,10
END
```

PRODUCES THE OUTPUT :

INVESTMENT	PRINCIPAL	INTEREST	NR.YEARS
1072.74	5000	.08	20
2315.97	5000	.08	10
2791.99	5000	.06	10

*** ERROR , YOU TRIED TO READ MORE NUMERIC DATA THAN YOU PUT IN ***

80

THE ERROR OCCURS BECAUSE YOU RUN OUT OF DATA DURING THE EXECUTION OF THE LOOP SET-UP BY THE UNCONDITIONAL TRANSFER . IF THE READ STATEMENT WERE NOT IN THE LOOP TO CAUSE THE PROGRAM TO STOP . THEN YOU WOULD BE IN AN 'INFINITE LOOP' , A CONDITION IN WHICH THERE IS NO WAY TO STOP , YOU MUST ALWAYS CHECK FOR THE 'INFINITE LOOP' CONDITION BY MAKING SURE THAT YOUR PROGRAM HAS AN EXIT .

80

2,) ON << EXPRESSION >> GO TO << STATEMENT NUMBER ,...., STATEMENT NUMBER >>

THIS SPECIAL FORM OF THE 'GO TO' COMMAND , IS CALLED THE 'COMPUTED GO TO' , THE EXPRESSION IN THE FORM OF THE STATEMENT MUST EVALUATE TO AN INTEGER NUMBER BETWEEN 1-->9999 . IF IT IS NOT AN AN INTEGER , OR OUTSIDE THIS RANGE AN ERROR WILL OCCUR .

WHEN THE 'COMPUTED GO TO' IS EXECUTED THE EXPRESSION IS EVALUATED , AND PROGRAM CONTROL TRANSFERS TO THE N-TH STATEMENT NUMBER , WHERE N-TH REPRESENTS THE VALUE OF THE EXPRESSION . FOR EXAMPLE :

LET I=3
ON I GO TO 100,33,475,9999

80

EXECUTION OF THE 'COMPUTED GO TO' WOULD CAUSE PROGRAM CONTROL TO TRANSFER UNCONDITIONALLY TO STATEMENT NUMBER 475 .

YOU MUST BE CAREFUL WHEN USING THE 'COMPUTED GO TO' NOT ONLY BECAUSE OF INFINITE LOOPS ; BUT BECAUSE THE EXPRESSION MUST BE AN INTEGER BETWEEN 1-->9999 , AND THERE MUST BE A STATEMENT NUMBER FOR 'ALL' POSSIBLE VALUES OF THE EXPRESSION .

80

3. CONDITIONAL BRANCHING

THE CONDITIONAL BRANCH TRANSFERS CONTROL ONLY IF CERTAIN RELATIONS ARE TRUE . IF THE THE TEST OF RELATIONS IS TRUE THEN TRANSFER OF CONTROL OCCURS ; OTHERWISE PROGRAM CONTROL CONTINUES WITH THE NEXT STATEMENT . THE FORM OF THE CONDITIONAL BRANCH IS:

IF << EXPRESSION >> << RELATION >> << EXPRESSION >> THEN << STATEMENT NUMBER >>

NOTE THAT ALPHA VARIABLES ARE NOT ALLOWED AS AN EXPRESSION IN A CONDITIONAL BRANCH .

80

THE RELATIONS ARE :

SYMBOLS	EXAMPLE	MEANING
GT	A GT B	A GREATER THAN B
GE	A GE B	A GREATER THAN OR EQUAL TO B
LT	A LT B	A LESS THAN B
LE	A LE B	A LESS THAN OR EQUAL TO B
NE	A NE B	A NOT EQUAL TO B
=	A = B	A EQUAL B

80

WHEN THE CONDITIONAL STATEMENT IS EXECUTED THE
(EXPRESSION RELATION EXPRESSION) IS TESTED , AND IF THE RELATION
IS TRUE , THEN CONTROL IS TRANSFERRED TO THE STATEMENT NUMBER.
OTHERWISE PROGRAM CONTROL CONTINUES TO THE NEXT SEQUENTIAL
STATEMENT , FOR EXAMPLE :

```
1 READ A
...
...
IF A LT 0 THEN 90
LET X=SQR(A)
...
...
90 PRINT 'ILLEGAL ARGUMENT',A
GO TO 1
...
```

THE ONLY TIME THAT THE CONDITIONAL BRANCH IS EXECUTED IS WHEN
A IS LESS THAN 0 .

80

YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT BRANCHING :
1.) ARE THE FOLLOWING STATEMENTS CORRECT
REPLY: YES OR NO)

GO TO END

no

IF X1 LT A\$ THEN 10

no

IF(X=3-4) GE 10 THEN GO TO 100

yes

YOUR ANSWER IS INCORRECT . THE ONLY THING ALLOWED
AFTER THEN IS A STATEMENT NUMBER .

LET Y=6,
ON Y GO TO 1,3,5,7,999

no

2.) CONSIDER THIS PROGRAM SEGMENT , ANY ERRORS (REPLY: YES OR NO).

```
1 READ A,B
3 LET Z=A*B
IF Z LT 0 GO TO 999
...
...
GO TO 3
DATA 3,5,7,10
999 END
```

yes

THERE IS AN 'INFINITE' LOOP IN THIS PROGRAM . IT
COULD BE CORRECTED BY CHANGING THE GO TO 3 , TO GO TO 1 .
WHEN YOU WRITE PROGRAMS HAVING BRANCHES OR LOOPS YOU MUST ALWAYS
CONSIDER HOW THE PROGRAM WILL STOP. YOU HAVE OBSERVED THAT
AN 'INFINITE LOOP' CAN BE STOPPED BY HAVING A READ STATEMENT
IN THE LOOP AND JUST RUN OUT OF DATA . HOWEVER, ENDING A PROGRAM
ON AN ERROR IS AN INELIGANT METHOD . THE MOST COMMON METHODS
USE THE 'GO TO' AND ' IF/THEN' COMMANDS TO CONTROL PROGRAM
LOOPS AND ARE AS FOLLOWS :

40

1. COUNT AND TEST METHOD , IN WHICH A COUNTER
IS INCREMENTED IN THE LOOP , AND WHEN THE COUNTER REACHES A
CERTAIN VALUE , BRANCH OUT OF THE LOOP . EXAMPLE PROGRAM

```
REM COUNT AND TEST METHOD
REM N IS COUNTER , INITIALIZED TO 0 , AND COUNTS FROM 1-->10
```

```

LET N=Z=0
READ X
10 LET Z=Z+X
LET N=N+1
IF N GT 10 THEN 100
GO TO 10
100 PRINT'SUM=',Z
DATA 10
END

```

REPLY WITH VALUE OF Z .

11

YOUR ANSWER IS INCORRECT . THE ONLY WAY TO BE SURE OF VALUES IN A LOOP IS TO SET UP A TABLE OF THE VARIABLES AND KEEP TRACK OF THERE VALUES IN THE LOOP :

X	N	Z
10	0	0
	1	10
	2	20
	3	30
	...	
	9	90
	10	100
	11	110

THUS Z=110

2. READ AND TEST METHOD , IN WHICH A VALUE IS READ IN AND TESTED FOR THE END OF LOOP CONDITION .

```

REM READ AND TEST DEMONSTRATION
REM 9999 IS END VALUE
LET Z=0
1 READ X
IF X=9999 THEN 9999
LET Z=Z+X
GO TO 1
DATA 1,2,3,4,5,6,9999
9999 PRINT 'SUM=',Z
END

```

REPLY WITH VALUE OF Z

21

C. SUMMARY

YOU HAVE SEEN HOW THE UNCONDITIONAL BRANCHES ('GO TO' AND 'COMPUTED GO TO') TRANSFER PROGRAM CONTROL, AND HOW THE CONDITIONAL BRANCH ('IF/THEN') TESTS FOR TRANSFER OF PROGRAM CONTROL; AND YOU HAVE OBSERVED THE CONTROL OF LOOPS SO THAT A PROGRAM SEGMENT MAY BE REPEATED UNTIL A SPECIFIED CONDITION IS MET. IN THE NEXT LESSON YOU WILL LEARN A BASIC STATEMENT TO CONTROL A LOOP BY THE INCREMENT AND TEST METHOD.

THE FOLLOWING PROBLEMS WILL TEST YOUR NEW SKILLS:

80

1.) WRITE A PROGRAM TO COUNT THE NUMBERS BETWEEN 50 AND 60, AND ALSO PRINT THEM OUT. THE INPUT DATA IS 10,50,35,75,62,60,54,54.

2.) WRITE A PROGRAM TO COMPUTE THE PRESENT WORTH OF AN INVESTMENT FOR SOME YEARS HENCE, AT VARIOUS INTEREST RATES. THE FORMULA IS:
$$P = S / ((1 + I)^N)$$
 WHERE P=PRESENT WORTH, S=PRINCIPAL, I=INTEREST, AND N=NR.OF YEARS. FOR DATA USE S=5000, AND I=.04-->.8, IN INCREMENTS OF .01, AND N=20

IF YOU DESIRE TO EXECUTE THESE PROGRAMS NOW
REPLY: YES; OTHERWISE REPLY: NO

no

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION?
(REPLY: YES OR NO)

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME
THEN REPLY: YES; OTHERWISE REPLY: NO AND THE INSTRUCTION
WILL CONTINUE.

no

REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO COVER

EXECUTION BEGINS...

*** LESSON 5 ***
THIS LESSON WILL INTRODUCE YOU TO ITERATION , SUBSCRIPTED
VARIABLES , AND LISTS (VECTORS) AND TABLES (MATRICES).

A. ITERATION (LOOPING)

IN THE LAST LESSON YOU WERE SHOWN HOW TO USE CONDITIONAL AND
UNCONDITIONAL BRANCHES TO CONTROL LOOPING . THE COUNT AND TEST
ITERATIVE LOOP OCCURS SO FREQUENTLY THAT AN ABBREVIATED BASIC
STATEMENT HAS BEEN DEvised TO CONTROL LOOPING . THE ITERATIVE
LOOP HAS THE FOLLOWING FORM :

```
FOR << SIMPLE VARIABLE >> = << EXPRESSION >> TO << EXPRESSION >>  
    STEP << EXPRESSION >>  
    ...  
    ...  
    ...  
NEXT << SIMPLE VARIABLE >>
```

SO

FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT :

```
FOR I=1 TO 10 STEP 2  
  LET X=X+1  
NEXT I
```

THE SIMPLE VARIABLE FOLLOWING 'FOR' IS THE LOOP INDEX . WHEN
THE FOR/NEXT PAIR IS EXECUTED , THE LOOP INDEX IS GIVEN THE
VALUE (INITIALIZED) OF THE FIRST EXPRESSION (I=1 IN EXAMPLE).
THIS INDEX IS THEN TESTED TO DETERMINE WHETHER IT IS 'GREATER THAN'
THE SECOND EXPRESSION AFTER 'TO' (10 IN EXAMPLE). IF IT IS
GREATER, CONTROL IS TRANSFERRED TO THE STATEMENT FOLLOWING
'NEXT'. OTHERWISE THE REMAINING STATEMENTS WITHIN THE LOOP
(FOR/NEXT) ARE EXECUTED SEQUENTIALLY UNTIL THE 'NEXT' STATEMENT
IS REACHED.

SO

WHEN THE 'NEXT' STATEMENT IS REACHED , THE LOOP
INDEX IS INCREASED (INCREMENTED) BY THE AMOUNT OF THE EXPRESSION
FOLLOWING 'STEP', AND CONTROL IS TRANSFERRED BACK TO THE 'FOR'
STATEMENT WHERE THE LOOP CONTINUES UNTIL THE INDEX VALUE IS

GREATER THAN THE FINAL VALUE . FOR EXAMPLE :

```
REM DEMO LOOP. SUM THE NUMBERS FROM 1-->10.
LET C=0
FOR I=1 TO 10 STEP 1
LET C=C+1
NEXT I
PRINT'SUM=',C
END
```

80

YOU WILL NOTICE THAT THE SIMPLE VARIABLE FOLLOWING 'NEXT' IS THE SAME AS THE SIMPLE VARIABLE FOLLOWING 'FOR', AND THAT THE 'NEXT' STATEMENT MARKS THE END OF THE LOOP . BECAUSE THE INCREMENT VALUE OF A LOOP IS COMMONLY ONE(1) , THE 'STEP' MODIFIER AND ITS EXPRESSION MAY BE OMITTED , AND THE INCREMENT VALUE WILL BE ASSUMED TO BE ONE(+1). FOR EXAMPLE THE ABOVE 'FOR' STATEMENT COULD BE WRITTEN :

```
FOR I=1 TO 10
```

THE INCREMENT VALUE AFTER 'STEP' MAY BE POSITIVE OR NEGATIVE ALLOWING THE FLEXIBILITY OF LOOPING FORWARD OR BACKWARD . FOR A NEGATIVE 'STEP' VALUE THE TEST BECOMES 'LESS THAN'. FOR EXAMPLE THE FOLLOWING 'FOR' STATEMENTS ARE EQUIVALENT :

```
FOR I=1 TO 10
FOR I=10 TO 1 STEP -1
```

80

ANOTHER USEFUL TECHNIQUE OF LOOPING IS 'NESTING' NESTING REFERS TO PLACING ONE LOOP INSIDE ANOTHER LOOP . THE INNER LOOP 'SPINS' AROUND AS MANY TIMES AS THE OUTER LOOP IS INCREMENTED . FOR EXAMPLE CONSIDER THIS PROGRAM SEGMENT

```
FOR I=1 TO 10
FOR J=1 TO 20 STEP 2
...
...
NEXT J
NEXT I
```

IN THIS EXAMPLE THE OUTSIDE LOOP(I) IS

REPEATED 10 TIMES, AND THE INNER LOOP(J) WOULD BE REPEATED 20 TIMES FOR EACH INCREMENT OF THE OUTSIDE LOOP, OR 200 REPETITIONS. LOOPS MAY BE NESTED UP TO A MAXIMUM OF 20 ; HOWEVER , THEY CANNOT OVERLAP . THE INNERMOST LOOP MUST BE CLOSED WITH ITS 'NEXT' STATEMENT BEFORE ENCOUNTERING THE NEXT OUTER LOOP'S 'NEXT' STATEMENT . FOR EXAMPLE :

```
FOR X=10 TO 1 STEP -1
FOR Y=3 TO 5
FOR Z=-3 TO -5 STEP -1
...
NEXT Z
NEXT Y
NEXT X
```

80

WITHIN A FOR/NEXT LOOP CONDITIONAL AND UNCONDITIONAL BRANCHES MAY BE USED TO TRANSFER CONTROL OUT OF A LOOP OR WITHIN LIMITS OF THE SAME LOOP . HOWEVER , IT IS NOT POSSIBLE TO BRANCH INTO THE MIDDLE OF A FOR/NEXT LOOP BECAUSE LOGIC PROBLEMS OCCUR AND AN ERROR WILL RESULT . AN ADDITIONAL ITEM TO BE CAREFUL ABOUT IS USING THE INDEX VARIABLE OF THE FOR/NEXT LOOP IN COMPUTATIONS . IF YOU ALTER THE VALUE OF THE LOOP INDEX YOU WILL EFFECT THE ACTION OF THE LOOP . FOR EXAMPLE :

```
FOR I=1 TO 10
LET I=I+10
NEXT I
PRINT I
END
```

REPLY WITH VALUE OF I THAT IS PRINTED

11

YOU WILL NOW BE ASKED SOME QUESTIONS ABOUT WHAT YOU HAVE JUST LEARNED .

```
1.) SAMPLE PROGRAM
LET S=0
FOR K=-5 TO 5
LET S=S+K
NEXT K
PRINT'SUM=',S
END
```

REPLY WITH VALUE OF S .

0

```
2.) SAMPLE PROGRAM
FOR I=100 TO 1 STEP 2
LET Z=I**2
LET Y= Z+10
```

```

NEXT I
PRINT I
END

```

REPLY WITH VALUE OF I

1

YOUR ANSWER IS INCORRECT . THE INDEX VALUE OF THE LOOP(I=100) IS GREATER THAN THE FINAL VALUE(1) IN THE FIRST TEST . THUS I=100 . IF THE LOOP WERE DECREMENTED IN STEPS OF -2 THEN THE LOOP WOULD BE EXECUTED AND THEN I=0 .

```

3.) SAMPLE PROGRAM
LET T=0
FOR I=5 TO 1 STEP -1
FOR J=1 TO 5
IF I=J THEN 10
GO TO 15
10 LET T=T+1
PRINT I
15 NEXT J
NEXT I
PRINT T
END

```

REPLY WITH VALUE OF T .

5

B. SUBSCRIPTED VARIABLES/LISTS AND TABLES

1. IN LESSON 1 YOU LEARNED THAT SIMPLE VARIABLES WERE A LETTER OR A LETTER FOLLOWED BY A DIGIT . SUBSCRIPTED VARIABLES CONSIST OF A LETTER FOLLOWED BY A SINGLE OR DOUBLE SUPSCRIPT IN PARENTHESIS THE SUBSCRIPTS MAY BE ANY LEGAL EXPRESSION THAT EVALUATES TO AN INTEGER VALUE . FOR EXAMPLE :

A(1) , B(3) , D(1,3) , Z'(1,J) , X(3**A) , D(A(1)) ARE LEGAL SUBSCRIPTED VARIABLE

BUT A1(1) , Z(1,2,3) ARE ILLEGAL SUBSCRIPTED VARIABLES.

A USE OF SUBSCRIPTED VARIABLES FOLLOWS :

80

2. A NUMERIC LIST , OR VECTOR , OR SINGLE DIMENSIONED ARRAY , IS A SET OF NUMERIC VALUES ARRANGED IN AN ORDERLY MANNER . FOR EXAMPLE , SUPPOSE YOU WOULD LIKE TO READ SOME NUMBERS INTO YOUR PROGRAM AND HAVE THESE NUMBERS AVAILABLE AND IDENTIFIED FOR USE AT A LATER TIME . YOU COULD ASSIGN SIMPLE VARIABLES TO EACH VALUE , BUT WHAT IF YOU HAD 100 NUMBERS ? IT IS EASIER TO THINK OF THE NUMBERS AS A LIST OF VALUES OR A VECTOR THE SIZE OF WHICH IS DETERMINED BY HOW MANY NUMBERS YOU HAVE . YOU THEN SIMPLY ASSIGN THE VALUES TO THE LIST .

80

THE FOLLOWING PROGRAM WILL ASSIGN 10 VALUES TO THE LIST 'A' WHICH CONTAINS 10 ELEMENTS :

```
DIM A(10)
FOR L=1 TO 10
  READ A(L)
NEXT L
DATA 1,3,7,5,6,9,4,2,14,4
END
```

THE LIST A , CONTAINING 10 ELEMENTS , IS REPRESENTED BY THE SUBSCRIPTED VARIABLE A(L) . READ A(L) IS IN A FOR/NEXT LOOP , AND WITHIN THE LOOP THE 11ST ELEMENTS A(1)-->A(10) ARE ASSIGNED VALUES . NOW THE LIST A HOLDS THE 10 VALUES AND EACH VALUE IS IDENTIFIABLE . CONSIDER THE FOLLOWING PROBLEM WHICH SEARCHES A LIST TO FIND THE LARGES

80

```
DIM N(10)
FOR I=1 TO 10
  LET N(I)=0
NEXT I
READ K
FOR J=1 TO K
  READ N(J)
NEXT J
LET L=N(1)
FOR J=2 TO K
  IF N(J) LT L THEN 10
  LET L=N(J)
```

```

10 NEXT J
PRINT 'LARGEST NUMBER=',L
DATA 5,3,5,9,10,6
END

```

REPLY WITH VALUE OF L

10

THE FIRST FOR/NEXT LOOP ZEROS OUT THE LIST .
 IT IS A GOOD PRACTICE TO PUT SOME VALUES IN THE LISTS YOU USE
 OTHERWISE THE COMPUTER MAY ASSIGN RANDOM VALUES . BY PUTTING
 ZERO IN EACH ELEMENT OF THE LIST YOU ARE ASSURED THAT THE LIST
 IS CLEANED UP BEFORE YOU USE IT.

80

3. IN ADDITION TO LISTS(VECTORS), BASIC ALLOWS
 YOU THE ABILITY TO USE TABLES , OR MATRICES , OR TWO DIMENSIONAL
 ARRAYS . THE SUBSCRIPTS OF A TABLE REPRESENT THE ROWS AND
 COLUMNS OF THE TABLE. THE FIRST SUBSCRIPT IS THE ROW AND THE
 SECOND SUBSCRIPT IS THE COLUMN . FOR EXAMPLE IN TABLE'D':
 D(3,4) REFERS TO THE VALUE OF THE ELEMENT IN ROW 3 , COLUMN 4
 OF TABLE'D'.

80

AS WITH A LIST , ANY ELEMENT OF A TABLE MAY BE
 REFERENCED BY DEFINING THE PAIR OF SUBSCRIPTS AS DESIRED IN
 ROW-COLUMN ORDER . IN A 3X3 TABLE(MATRIX) THE TABLE IS REFERENCED
 AS FOLLOWS :

1,1	1,2	1,3
2,1	2,2	2,3
3,1	3,2	3,3

WHERE THE FIRST SUBSCRIPT IDENTIFIES THE ROW AND THE SECOND THE COLUMN .
 USING THIS REFERENCE SYSTEM CONSIDER HOW TO FILL THE FOLLOWING TABLE

1	2	3
4	5	6
7	8	9

80

THIS TABLE'X' COULD BE FILLED AS FOLLOWS :

```

DIMX(3,3)
FOR I=1 TO 3
FOR J=1 TO 3
READ X(I,J)
NEXT J
NEXT I
DATA 1,2,3,4,5,6,7,8,9

```

END

REPLY WITH VALUE OF X(2,3)

6

NO WITH TABLE 'X' ASSIGNED VALUES CONSIDER THIS
PROGRAM (ASSUMING TABLE 'X' HAS BEEN FILLED BY THE ABOVE PROGRAM)

```
LET S=0
FOR I=1 TO 3
FOR J=1 TO 3
IF I NE J THEN 5
LET S=S+X(I,J)
5 NEXT J
NEXT I
PRINT 'SUM=',S
END
```

REPLY WITH VALUE OF S

15

YOU HAVE SEEN HOW SUBSCRIPTED VARIABLES ARE
USED TO SET UP LISTS (VECTORS) AND TABLES (MATRICES). HOWEVER TO
USE LISTS AND TABLES IN A PROGRAM, YOU MUST DIMENSION THE
MAXIMUM SIZE OF YOUR LIST OR TABLE SO THAT ENOUGH SPACE WILL
BE ALLOCATED IN THE COMPUTER MEMORY. THIS DIMENSIONING IS DONE
WITH THE DIM STATEMENT.

80

C. DIM STATEMENT

THE DIM STATEMENT TELLS THE COMPUTER THE MAXIMUM SIZE OF VECTORS
AND TABLES THAT WILL BE USED IN YOUR PROGRAM. THE DIM STATEMENT
MUST APPEAR IN THE PROGRAM BEFORE 'ANY' REFERENCE IS MADE
TO THE LIST OR TABLE. IN GENERAL PRACTICE THE DIM STATEMENT
IS USUALLY THE FIRST STATEMENT IN THE PROGRAM. THE FORM OF THE
DIM STATEMENT IS :

DIM <<LIST VARIABLE>> (<<SIZE>>)

DIM <<TABLE VARIABLE>> (<<SIZE,SIZE>>)

80

THE LIST AND TABLE VARIABLES ARE SUBSCRIPTED VARIABLES AS DEFINED EARLIER. THE SIZE IS AN UNSIGNED INTEGER IN PARENTHESIS WHICH DENOTES THE MAXIMUM SIZE OF THE LIST OR TABLE. THE DIM STATEMENT MAY CONTAIN A NUMBER OF LISTS OR TABLES WITH THEIR SIZES SEPARATED BY COMMAS, FOR EXAMPLE:

DIM A(6),X(10,3),Z(14,21)

GO

IF YOU TRY TO REFERENCE AN ELEMENT IN A LIST OR TABLE BEYOND THE MAXIMUM SIZE IN THE DIM STATEMENT YOU WILL GET AN ERROR. THE MAXIMUM SIZE LIST(VECTOR) OR TABLE(MATRIX) ALLOWED BY THE CAI-BASIC COMPILER IN ANY ONE PROGRAM IS A TOTAL OF 1600 COMPUTER MEMORY SPACES.

YOU WILL NOW BE ASKED SOME QUESTIONS:

1.) ARE THE FOLLOWING DIM STATEMENTS CORRECT, REPLY: YES OR NO

DIM A1(10)

yes

THE STATEMENT IS INCORRECT. SUBSCRIPTED VARIABLES ARE A 'SINGLE' LETTER FOLLOWED BY ONE OR TWO SUBSCRIPTS IN PARENTHESIS.

DIM X(500),B(1,10),C(5,5,5)

no

D. SUMMARY

IN THIS LESSON YOU HAVE LEARNED HOW TO USE THE FOR/NEXT STATEMENT AND HOW TO MANIPULATE LISTS(VECTORS) AND TABLES(MATRICES) BY USING SUBSCRIPTS VARIABLES AND THE DIM STATEMENT. YOU NOW HAVE ALL THE TOOLS TO BEGIN WRITING SOPHISTICATED PROGRAMS; AND AS YOU WRITE MORE COMPLICATED PROGRAMS, YOU WILL FIND A NEED FOR SUBROUTINES. SUBROUTINES ARE COMMONLY USED PROGRAM SEGMENTS THAT ARE USED OVER AGAIN IN OTHER PARTS OF YOUR PROGRAM. SUBROUTINES ALLOW YOU TO BRANCH TO THE COMMONLY USED SEGMENT AND THEN RETURN TO WHERE YOU WERE AND CONTINUE EXECUTING. THE SUBROUTINE CAN BE 'CALLED' FROM ANYWHERE IN YOUR PROGRAM. THE GOSUB STATEMENT ALLOWS SUBROUTINES IN BASIC, AND YOU WILL BE

INTRODUCED TO IT IN THE NEXT LESSON .

YOU WILL NOW BE GIVEN TWO OPTIONAL PROGRAMMING
PROBLEMS TO EXERCISE YOUR NEW TOOLS .

1.) WRITE A PROGRAM TO REVERSE THE NUMBERS IN A 10-ELEMENT LIST 'X'
IN OTHER WORDS INTERCHANGE X(1) WITH X(10), X(2) WITH X(9), ETC.
READ IN TEN VALUES AND TEST YOUR PROGRAM BY PRINTING THE LIST
BEFORE AND AFTER .

2.) WRITE A PROGRAM TO ARRANGE THE FOLLOWING LIST IN
DECENDING ORDER : 10,30,5,15,40 . ONE METHOD OF APPROACHING THIS
IS TO CHECK THE FIRST ELEMENT OF THE LIST AGAINST THE SECOND .
IF THE FIRST IS NOT LARGER THEN EXCHANGE THE TWO , OTHERWISE
GO AND COMPARE THE NEXT TWO IN THE LIST. THIS PROCESS IS REPEATED
UNTIL THERE ARE NO MORE EXCHANGES TO BE MADE . A COUNT OF
THE EXCHANGES CAN BE MADE , AND WHEN THE COUNT EQUALS 0 THE LIST IS IN ORDER.

go

IF YOU WANT TO EXECUTE THESE PROBLEMS REPLY : YES ,OTHERWISE NO ,

no

IF YOU WOULD LIKE TO SEE A SOLUTION TO THE PROBLEMS REPLY : YES , OTHERWISE NO

no

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?
(REPLY : YES OR NO)

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION
WILL CONTINUE .

no

REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO COVER

EXECUTION BEGINS...

*** LESSON 6 ***
THIS LESSON WILL INTRODUCE YOU TO SUBROUTINES AND THE PROGRAMMING
TECHNIQUE CALLED 'RECURSION'.

A. SUBROUTINES

IN MANY PROGRAMS A BLOCK OF STATEMENTS MAY BE NEEDED ON MORE
THAN ONE OCCASION. SINCE THE REPETITION OF A BLOCK OF STATEMENTS
IS BURDENSOME, A TECHNIQUE TO ECONOMIZE ON CODING INSTRUCTIONS
CALLED A SUBROUTINE, IS PRESENTED. THE FORM OF THE SUBROUTINE IS :

```
GOSUB << STATEMENT NUMBER >>
...
...
...
RETURN
```

80

EXECUTION OF THE GOSUB STATEMENT CAUSES THE
COMPUTER TO TRANSFER CONTROL TO THE STATEMENT NUMBER AFTER 'GOSUB'.
WHEN CONTROL IS TRANSFERRED, STATEMENTS ARE EXECUTED SEQUENTIALLY
UNTIL A 'RETURN' IS ENCOUNTERED. AT THAT TIME, CONTROL
IS RETURNED TO THE NEXT STATEMENT FOLLOWING THE 'GOSUB'
STATEMENT WHICH 'CALLED' THE SUBROUTINE. FOR EXAMPLE CONSIDER
THE PROGRAM SEGMENT :

80

```
...
PRINT A,B,C,D,E
GOSUB 500
FOR I=1 TO E
...
...
...
500 REM SUBROUTINE
LET X=A+B
LET Z=B+C
LET R=SQR(D)
RETURN
...
```

80

THE SUBROUTINE CONSISTS OF A SEQUENCE OF BASIC STATEMENTS ENDING WITH A 'RETURN' STATEMENT . THE SUBROUTINE MAY ONLY BE ENTERED FROM A 'GOSUB' STATEMENT , AND WILL ONLY RETURN TO ITS PROPER PLACE AFTER ENCOUNTERING A 'RETURN' STATEMENT . ALL THE VARIABLES OF THE MAIN PROGRAM ARE AVAILABLE (PASSED) TO THE SUBROUTINE , AND VICE VERSA . THE VARIABLES IN THE SUBROUTINE MUST BE CHOSEN CAREFULLY SO AS NOT TO ACCIDENTLY CONFLICT OR ALTER VARIABLES IN THE MAIN PROGRAM .

80

THE FOLLOWING PROGRAM TO COMPUTE THE GREATEST COMMON DENOMINATOR(GCD) OF THREE NUMBERS BY EUCLID'S ALGORITHM WILL DEMONSTRATE A USE OF SUBROUTINES :

```

      REM FIND GCD OF A,B,C
      PRINT 'A','B','C','GCD'
20  READ A,B,C
      IF C=9999 THEN 9999
      LET X=A
      LET Y=B
      REM G=GCD OF A,B
      GOSUB 200
      LET X=G
      LET Y=C
      REM G=GCD OF G,C
      GOSUB 200
      PRINT A,B,C,G
      GO TO 20
      200 LET Q=INT(X/Y)
          LET R=X-Q*Y
          IF R=0 THEN 300
          LET X=Y
          LET Y=R
          GO TO 200
      300 LET G=Y
      RETURN
      DATA 60,90,120
      DATA 38456,64872,98765
      DATA 0,0,9999
9999 END

```

OUTPUT PRODUCED :

A	B	C	GCD
60	90	120	30
38456	64872	98765	1

80

GOSUB STATEMENTS MAY BE NESTED TO LOGICALLY
 CALL ANOTHER GOSUB , SIMILAR TO FOR/NEXT NESTED LOOPS . YOU
 MUST ONCE MORE BE CAREFUL ABOUT THE STATUS OF VARIABLES BECAUSE
 VARIABLES WILL BE PASSED FROM ONE SUBROUTINE TO ANOTHER .
 A SUBROUTINE THAT IS NESTED SO THAT IT CALLS ITSELF IS CALLED
 RECURSION OF SUBROUTINES OR RECURSION .

80

8. RECURSION

RECURSION IS A PROGRAMMING TECHNIQUE IN WHICH SUBROUTINES CAN
 CALL THEMSELVES .FOR EXAMPLE CONSIDER THIS PROGRAM WHICH FINDS
 THE FACTORIAL OF A NUMBER USING ITERATIVE METHODS AND THEN
 USING RECURSION :

```

    REM FACTORIAL : F(N)=1*2*3*.....*(N-1)*N
    PRINT'ITERATIVE SOLUTION'
    PRINT'N','F(N)'
10  READ N
    IF N GT 0 THEN 20
    GO TO 50
20  PRINT N,
    GOSUB 100
    PRINT F
    GO TO 10
    100 REM ITERATIVE SOLUTION
        LET F=1
        FOR I=1 TO N
            LET F=F*I
        NEXT I
        REM RETURN F=FACTORIAL(N)
        RETURN
50  REM FACTORIAL : F(N)= IF N=0 THEN F(N)=1
    REM OTHERWISE , F(N)=N*F(N-1)
    RESTORE
    PRINT'RECURSIVE SOLUTION'
60  READ N
    IF N GT 0 THEN 70
    GO TO 9999
70  PRINT N,
    GO SUB 200
    PRINT F
    GO TO 60
    200 REM RECURSIVE SOLUTION
        IF N GT 0 THEN 210
        LET F=1
        RETURN
    210 LET N=N-1
        REM RECURSIVE CALL
  
```

```

      GOSUB 200
      LET N=N+1
      LET F=F*N
    RETURN
  DATA 2,5,8,6,-1
999 END

```

80

IN THE ITERATIVE METHOD THE FACTORIAL FUNCTION, $F(N)$, WAS MULTIPLIED BY ITSELF IN A LOOP FROM $I=1 \rightarrow N$. IN THE RECURSIVE SOLUTION THE FACTORIAL FUNCTION, $F(N)$, IS DEFINED IN TERMS OF ITS FINAL VALUE. WHEN $N=0$, $F(N)=1$; OTHERWISE $F(N)=N \cdot (\text{FACTORIAL OF } N-1)$.

TO KEEP TRACK OF WHERE THE SUBROUTINE CALLS RETURN IN RECURSION IT HELPS TO EVISIONING A LAST-IN-FIRST-OUT (LIFO) STACK CONTAINING THE ADDRESS OF THE GOSUB STATEMENTS. EVERY TIME A GOSUB IS ENCOUNTERED, PUT ITS ADDRESS ON TOP OF THE STACK (PUSHING DOWN ANYTHING PREVIOUSLY ON THE STACK). THEN EVERY TIME A RETURN IS ENCOUNTERED, RETURN TO THE TOP ADDRESS ON THE STACK (AND POP UP THE NEXT ADDRESS ON THE STACK).

80

TO FULLY UNDERSTAND THE CONCEPT OF RECURSION YOU SHOULD STEP THROUGH THE FACTORIAL PROBLEM BY HAND USING THE HELP OF THE STACK TO SEE HOW RECURSION WORKS. IF YOU UNDERSTAND THE CONCEPT OF RECURSION YOU ARE READY TO SOLVE THIS PROBLEM:

```

      LET X=1
      GOSUB 100
      PRINT X
      GO TO 9999
    100 LET X=X+1
      IF X GT 3 THEN 150
      GOSUB 100
    150 LET X=X+1
      RETURN
9999 END

```

REPLY WITH VALUE OF X

5

YOUR ANSWER IS INCORRECT . CONSIDER THE FOLLOWING
TABLE OF VALUES FOR X , AND ITEMS IN THE STACK(1-FIRST GOSUB, 2-SECOND GOSUB)

X=1	STACK
	1
X=2	STACK
	2
	1
X=3	STACK
	2
	2
	1
X=4	
X=5	STACK
	2
	1
X=6	STACK
	1
X=7	

YOU SHOULD STEP THROUGH THIS
EXAMPLE UNTIL YOU UNDERSTAND THE RECURSION TECHNIQUE . HOWEVER,
KNOWING HOW TO USE RECURSION IS NOT A REQUIREMENT FOR KNOWING
HOW TO USE BASIC , IT IS ONLY A CLASSIC PROGRAMMING TECHNIQUE .

C. SUMMARY
SUBROUTINES AND RECURSION ARE A VALUABLE ADDITION TO YOUR REPERTOIR
OF BASIC STATEMENTS . THEY SAVE BOTH PROGRAMMER TIME AND
COMPUTER STORAGE SPACE . THE GOSUB/RETURN COMMAND AND THE OTHER
TWELVE BASIC STATEMENTS THAT YOU HAVE ALREADY LEARNED AND USED
FORM THE BASIC LANGUAGE .
THE FACILITY THAT YOU GAIN IN PROGRAMMING BY USING THE BASIC
LANGUAGE WILL DEPEND UPON HOW OFTEN YOU EXERCISE YOUR SKILLS.
THE LAST LESSON WILL PROVIDE YOU WITH A BRIEF SUMMARY OF THE
BASIC LANGUAGE .

80

THE FOLLOWING PROBLEMS WILL TEST YOUR LATEST SKILLS :

1. COMPUTE THE NET SALARY , RETIREMENT CONTRIBUTION
AND TAX FOR N EMPLOYEES. NET PAY=GROSS SALARY-RETIREMENT-TAX
USE SUBROUTINES TO CALCULATE :

A) RETIREMENT CONTRIBUTION

IF SALARY <\$800 , R=\$0.0
IF SALARY <\$2500 , R=\$10.
IF SALARY >\$2500 , R=\$20,

B) TAX

IF SALARY <\$600 , T=\$5 ; OTHERWISE
T=.03(SALARY)

THE DATA IS:
EMPLOYEE

GROSS SALARY

JONES	3000
SMITH	5500
BROWN	475
DALEY	10,500
BERRY	4300

2. WRITE A RECURSIVE PROGRAM TO SUM THE NUMBERS
FROM X TO Y . READ IN YOUR OWN DATA AND TEST YOUR PROGRAM .

go

IF YOU WANT TO EXECUTE THESE PROGRAMS NOW , THEN
REPLY : YES ,OTHERWISE NO

no

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?
(REPLY : YES OR NO)

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION
WILL CONTINUE .

no

REPLY WITH THE NUMBER OF THE LESSON YOU WISH TO COVER

EXECUTION BEGINS...

*** LESSON 7 ***

THIS LESSON SUMMARIZES THE BASIC DEFINITIONS AND BASIC STATEMENTS THAT YOU HAVE LEARNED IN CAI-BASIC .

A. BASIC DEFINITIONS

1. ALPHA-NUMERIC CHARACTERS :

A) DIGITS : 0-->9

B) LETTERS : A-->Z

C) SPECIAL CHARACTERS : * , / + - () = ' , \$

2. STATEMENT NUMBERS : ONE TO FOUR DIGITS (0-->9999)

3. STRING : ANY SEQUENCE OF ALPHA-NUMERIC CHARACTERS ENCLOSED IN 'SINGLE' QUOTES . EG. 'HELP'

80

4. NUMBERS : (LIMITED TO 9 DIGITS)

A) INTEGERS : DIGITS WITH NO FRACTIONAL PART . EG. 5,7,10

B) REAL : DIGIT WITH A FRACTIONAL PART . EG. 5.0,7.31,-16.0
A NUMBER MAY BE PRECEDED BY A SIGN (+,-) , BUT IS ASSUMED TO BE POSITIVE IF NONE IS GIVEN .

5. VARIABLES :

A) SIMPLE VARIABLES : A SINGLE LETTER , OR A SINGLE LETTER FOLLOWED BY A DIGIT . EG A,A1,B6,Z0

B) ALPHA VARIABLES : A SINGLE LETTER FOLLOWED BY A DOLLAR SIGN (\$) EG. A\$,V\$

C) SUBSCRIPTED VARIABLE : (SINGLE LETTER)

(1) SINGLE SUBSCRIPT : <<LETTER>> (<<EXPRESSION>>)

(2) DOUBLE SUBSCRIPT : <<LETTER>> (<<EXPRESSION,EXPRESSION>>)

EG

A(5),Z(5,10),X(A*B,X**2)

80

5. OPERATORS : (LISTED IN DECENDING HIERARCHY)

- A) EXPONENTIATION **
- B) MULTIPLICATION *
- C) DIVISION /
- D) ADDITION +
- E) SUBTRACTION -

6. EXPRESSIONS :

- A) SINGLE NUMBER OR VARIABLE EG. 10,-.53,D,X(I,J)
- B) BUILT IN FUNCTION EG. SQR(10),TAN(.75)
- C) ARITHMETIC EXPRESSION : EXPRESSIONS SEPARATED BY OPERATORS AND GROUPED BY PARENTHESIS . EG. 5+6.0 , A**2(3*X-k)

80

7. RELATIONS

- | | | |
|-------|--------|------------------------------|
| A) GT | A GT B | A GREATER THAN B |
| B) GE | A GE B | A GREATER THAN OR EQUAL TO B |
| C) LT | A LT B | A LESS THAN B |
| D) LE | A LE B | A LESS THAN OR EQUAL TO B |
| E) NE | A NE B | A NOT EQUAL TO B |
| F) = | A = B | A EQUAL TO B |

B. BASIC STATEMENTS

1. REM << ANY SET OF ALPHA-NUMERIC COMMENTS >>

2. READ << VARIABLE,....,VARIABLE >>

3. END

80

4. LET << VARIABLE >> = << EXPRESSION >>

5. PRINT << 'STRING' OR EXPRESSION,....,'STRING' OR EXPRESSION >>
OR SIMPLY PRINT

6. DATA << 'STRING' OR NUMBER,....,'STRING' OR NUMBER >>

7. RESTORE OR RESTORE\$

8. IF << EXPRESSION >> << RELATION >> THEN << STATEMENT NUMBER >>

9. GO TO << STATEMENT NUMBER >>

10. ON << EXPRESSION >> GO TO << STATEMENT NUMBER,....,STATEMENT NUMBER >>

11. FOR << SIMPLE VARIABLE >> = << EXPRESSION >>
TO << EXPRESSION >> STEP << EXPRESSION >>

....

....

....

NEXT << SIMPLE VARIABLE >>

12. DIM << LETTER >> (<< INTEGER EXPRESSION >>) OR
DIM << LETTER >> (<< INTEGER EXPRESSION, INTEGER EXPRESSION >>)

13. GOSUB << STATEMENT NUMBER >>

...

...

...

RETURN

go

THAT CONCLUDES YOUR INSTRUCTION WITH CAI-BASIC .
HOWEVER , YOU ARE INVITED TO USE WHATEVER FACILITIES OF CAI-BASIC
YOU DESIRE AT ANY TIME . IF YOU HAVE NOT RUN PROGRAMS UNDER
THE OS/BATCH MODE (PUNCHING YOUR OWN CARDS AND HANDING THEM
ACROSS THE COUNTER TO BE RUN), YOU SHOULD GET THE BASIC MANUAL,
TECHNICAL NOTE NR. 0211-12 , IN ROOM I-147 TO FIND THE PROPER
JOB CONTROL CARDS REQUIRED . GOOD LUCK WITH THE COMPUTER , AND
REMEMBER IT ONLY DOES WHAT YOU TELL IT TO DO - GIGO (GARBAGE IN
GARBAGE OUT).

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?
(REPLY : YES OR NO)

no

IF YOU WANT TO EXECUTE PROGRAMS AT THIS TIME
THEN REPLY : YES ; OTHERWISE REPLY : NO AND THE INSTRUCTION
WILL CONTINUE .

yes

**** CAIBASIC COMPILER ****

THE CAIBASIC COMPILER IS A LINE BY LINE INTERPRETER .THE BASIC

INTERPRETER ACCEPTS STANDARD BASIC STATEMENTS , AND IT ANALYZES EACH BASIC STATEMENT AS IT IS INPUT . AN ADDED FEATURE OF THE CAIBASIC COMPILER IS AN EDITING AND A DEBUGGING ROUTINE THAT ALLOWS THE USER TO ADD , DELETE AND CORRECT STATEMENTS ; AND TO GET A LISTING OF ALPHA-NUMERIC AND NUMERIC DATA USED , AND A TRACE OF ALL SIMPLE VARIABLES AS THEY ARE ASSIGNED VALUES IN THE PROGRAM.

THE DEBUG FEATURE IS USED BY ADDING THE KEY WORD DEBUG AS A STATEMENT TO YOUR PROGRAM .

THE EDITING MODE IS AVAILABLE TO THE USER WHEN AN EXECUTION ERROR OCCURS AND AFTER SUCCESSFUL EXECUTION .

IN THE EVENT OF AN INPUT ERROR THE INTERPRETER WILL ANALYZE THE ERROR AND PRINT AN ERROR MESSAGE. IF THIS OCCURS FIND THE ERROR ,AND INPUT THE CORRECT BASIC STATEMENT FOR THE CURRENT LINE OF INPUT .

(NOTE : TYPING ERRORS CAN BE DELETED BY TYPING FOUR DOLLAR SIGNS (\$\$\$\$) ON THE SAME LINE AS THE ERROR , HITTING CARRIAGE RETURN , AND INPUTTING THE LINE AGAIN .)

**** CAIBASIC EXECUTION ****

INPUT BASIC PROGRAM NOW (ONE LINE AT A TIME)

```
let x=1
gosub 100
print x
go to 9999
  100 let x=x+1
    if x gt 3 then 150
    gosub 100
  150 return
9999 end
```

DO YOU WANT TO EDIT YOUR PROGRAM
REPLY : YES OR NO .

yes

**** CAIBASIC EDIT MODE ****

BY USING THE REFERENCE NUMBERS LISTED TO THE LEFT OF YOUR BASIC PROGRAM STATEMENTS YOU MAY ADD , DELETE , OR CORRECT ONE LINE OF THE PROGRAM AT A TIME . IN ALL EDITING THE FIRST STEP IS TO INPUT THE PROGRAM STATEMENT REFERENCE NUMBER AND HIT THE CARRIAGE RETURN . THE SECOND STEP DEPENDS ON WHAT EDITING YOU DO :

1. DELETE
THE BASIC STATEMENT REFERENCED IS DELETED BY TYPING THE LETTERS DEL
2. CORRECT
TO CORRECT THE BASIC STATEMENT REFERENCED TYPE IN THE COMPLETE CORRECT BASIC STATEMENT .
3. ADD
A BASIC STATEMENT IS ADDED 'AFTER' THE BASIC STATEMENT REFERENCED BY TYPING THE LETTERS ADD1 FOLLOWED BY THE BASIC STATEMENT . ALL BLANKS FOLLOWING THE LETTERS ADD1 WILL BE INCLUDED IN THE BASIC STATEMENT . TO PLACE A STATEMENT 'BEFORE' THE FIRST STATEMENT IN THE PROGRAM , USE THE REFERENCE NUMBER 0 .

```

1      LET X=1
2      GOSUB 100
3      PRINT X
4      GO TO 9999
5      100 LET X=X+1
6          IF X GT 3 THEN 150
7      GOSUB 100
8      150 RETURN
9      9999 END

```

INPUT REFERENCE NUMBER NOW
8

INPUT EDITING NOW (DEL,BASIC STATEMENT,ADD1....)

150 let x=x+1

MORE CORRECTIONS TO BE MADE ?? REPLY: YES ;
OR REPLY: NO AND THE EDITED PROGRAM WILL BE EXECUTED .

yes

```

1      LET X=1
2      GOSUB 100
3      PRINT X
4      GO TO 9999
5          100 LET X=X+1
6              IF X GT 3 THEN 150
7                  GOSUB 100
8                      150 LET X=X+1
9                          9999 END

```

INPUT REFERENCE NUMBER NOW

INPUT EDITING NOW (DEL,BASIC STATEMENT,ADD1....)

add1 return

MORE CORRECTIONS TO BE MADE ?? REPLY: YES ;
OR REPLY: NO AND THE EDITED PROGRAM WILL BE EXECUTED .

no

```

LET X=1
GOSUB 100
PRINT X
GO TO 9999
    100 LET X=X+1
        IF X GT 3 THEN 150
            GOSUB 100
                150 LET X=X+1
                    RETURN
9999 END

```

7

DO YOU WANT TO EXECUTE ANOTHER PROGRAM
REPLY : YES OR NO .

no

DO YOU WANT TO TERMINATE YOUR INSTRUCTION SESSION ?
(REPLY : YES OR NO)

yes
R; T=6.99/23.96 15.56.04

BIBLIOGRAPHY

1. FENICHEL, R. R. , WEIZENBAUM, J. , and YOCHELSON, J.C. , "Program to Teach Programming," COMMUNICATIONS OF THE ACM, v.13, March 1971.
2. FENICHEL, R. R. , "The Teach System" (unpublished work, PROJECT MAC M-388, MIT, Cambridge, Mass. , 1969).
3. HARE, V. C. , BASIC PROGRAMMING, Harcourt, Brace, and World, Inc., 1970.
4. KERR, E. G. , TING, E. G. , and WALDEN, W. M. , "A Control Program for Computer Assisted Instruction," 24TH CONFERENCE PROCEEDINGS OF THE ACM, 1969.
5. NOLAN, R. L. , INTRODUCTION TO COMPUTING THROUGH THE BASIC LANGUAGE, Holt, Rinehart, and Winston, Inc. , 1969.
6. OETTENDER, A. G. , RUN, COMPUTER, RUN, Harvard University Press, 1969.
7. SMITH, W. R. , and YOUNG, J. L. , "Computer Assisted Instruction," MANAGEMENT JOURNAL, Naval Postgraduate School, March 1971.
8. STOLUROW, L. M. , "Computer Assisted Instruction," REPORT OF A CONFERENCE, U. S. Department of HEW, Washington, 1969.
9. U. S. COMMISSION ON INSTRUCTIONAL TECHNOLOGY, "To Improve Learning," U. S. Government Printing Office, Washington, 1970.
10. U. S. NAVAL POSTGRADUATE SCHOOL TECHNICAL NOTE NR. 0211-12 "Basic Language Manual," April 1971.