

WFO FILE COPY

2

NAVAL POSTGRADUATE SCHOOL

Monterey, California



DTIC
ELECTE
FEB 17 1989
S QH D

AD-A204 362

THESIS

AN EXPERT SYSTEM INTERFACED WITH A DATABASE
SYSTEM TO PERFORM TROUBLESHOOTING OF AIRCRAFT
CARRIER PIPING SYSTEMS

by

Irving B. Clayton III and Patsy R. Boozer

December 1988

Thesis Advisor:

C. Thomas Wu

Approved for public release; distribution is unlimited.

89 2 16 010

Unclassified

Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified			1b Restrictive Markings						
2a Security Classification Authority			3 Distribution Availability of Report Approved for public release; distribution is unlimited.						
2b Declassification/Downgrading Schedule			5 Monitoring Organization Report Number(s)						
4 Performing Organization Report Number(s)			7a Name of Monitoring Organization Naval Postgraduate School						
6a Name of Performing Organization Naval Postgraduate School		6b Office Symbol (If Applicable) 37	7b Address (city, state, and ZIP code) Monterey, CA 93943-5000						
6c Address (city, state, and ZIP code) Monterey, CA 93943-5000		9 Procurement Instrument Identification Number							
8a Name of Funding/Sponsoring Organization		8b Office Symbol (If Applicable)	10 Source of Funding Numbers						
8c Address (city, state, and ZIP code)		<table border="1"> <tr> <td>Program Element Number</td> <td>Project No</td> <td>Task No</td> <td>Work Unit Accession No</td> </tr> </table>				Program Element Number	Project No	Task No	Work Unit Accession No
Program Element Number	Project No	Task No	Work Unit Accession No						
11 Title (Include Security Classification) An Expert System Interfaced with a Database System to Perform Troubleshooting of Aircraft Carrier Piping Systems									
12 Personal Author(s) Clayton III, Irving B. and Boozer, Patsy R.									
13a Type of Report Master's Thesis		13b Time Covered From To		14 Date of Report (year, month, day) December 1988					
15 Page Count 169									
16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.									
17 Cosati Codes			18 Subject Terms (continue on reverse if necessary and identify by block number)						
Field	Group	Subgroup	Expert System, Database, Modeling						
19 Abstract (continue on reverse if necessary and identify by block number)									
Maintaining and troubleshooting aircraft carrier through tank piping systems is a labor intensive, operational fleet problem. There is a clear need for a useful database and expert system to aid in fault isolation and repair planning for these systems. The multiple extensive piping systems of an aircraft carrier create an intimidating modelling problem for implementation in a database. The interface of an expert system to a large database to obtain improved execution speed, exploit a useful data model, reduce memory requirements, and enhance total system capability is examined and implemented. A flexible model for representing a large ship's piping systems in a database is presented. <i>Keywords: these, systems, management, (CR)</i>									
20 Distribution/Availability of Abstract			21 Abstract Security Classification						
☒ unclassified/unlimited ☐ same as report ☐ DTIC users			Unclassified						
22a Name of Responsible Individual C. Thomas Wu			22b Telephone (Include Area code) (408) 646-3391		22c Office Symbol/T. 52Wq				

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted

security classification of this page

All other editions are obsolete

Unclassified

Approved for public release; distribution is unlimited.

An Expert System Interfaced with a Database System to Perform
Troubleshooting of Aircraft Carrier Piping Systems

by

Irving B. Clayton III
Commander, United States Navy
B.S., University of Virginia, 1972

and

Patsy R. Boozer
Lieutenant, United States Navy
B.S., University of South Carolina, 1979

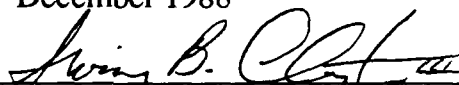
Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

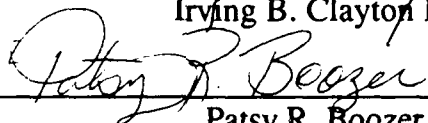
from the

NAVAL POSTGRADUATE SCHOOL
December 1988

Author:



Irving B. Clayton III



Patsy R. Boozer

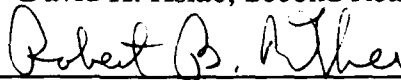
Approved by:



C. Thomas Wu, Thesis Advisor



David K. Hsiao, Second Reader



Robert B. McGhee, Chairman
Department of Computer Science



Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

Maintaining and troubleshooting aircraft carrier through tank piping systems is a labor intensive, operational fleet problem. There is a clear need for a useful database and expert system to aid in fault isolation and repair planning for these systems. The multiple extensive piping systems of an aircraft carrier create an intimidating modelling problem for implementation in a database. The interface of an expert system to a large database to obtain improved execution speed, exploit a useful data model, reduce memory requirements, and enhance total system capability is examined and implemented. A flexible model for representing a large ship's piping systems in a database is presented.

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	



TABLE OF CONTENTS

I. INTRODUCTION.....	1
II. BACKGROUND.....	3
A. INTERNAL ARRANGEMENT	3
B. TROUBLESHOOTING.....	5
III. DESIGN AND IMPLEMENTATION.....	8
A. MODELLING THE PIPING SYSTEM	8
B. EXPERT SYSTEM.....	10
C. SOFTWARE	12
D. SCOPE OF PROTOTYPE	13
1. Damage Control Void.....	14
2. Fuel Oil Service Tank	14
3. Fuel Oil Storage Tank.....	14
4. Contaminated Tank	15
5. JP-5 Tank.....	15
E. IMPLEMENTATION OF THE RELATIONAL MODEL.....	15
F. PROTOTYPE DEMONSTRATION.....	20
1. Database System Operation.....	20
2. Expert System Operation	23
IV. CONCLUSIONS.....	28
A. DATABASE SYSTEM/EXPERT SYSTEM CONNECTION.....	28
B. REFINING THE PROTOTYPE	29
C. POTENTIAL EXPERT SYSTEMS.....	30

APPENDIX A. DECISION TREES.....	31
APPENDIX B. RELATIONAL DIAGRAM.....	41
APPENDIX C. PROGRAM LISTING	42
LIST OF REFERENCES.....	159
INITIAL DISTRIBUTION LIST.....	160

LIST OF FIGURES

Figure 1.	Through Tank Piping Arrangement for No. 4. MMR.....	4
Figure 2.	Basic Relation Diagram.....	17
Figure 3.	Contains Relation.....	17
Figure 4.	Pipesystem Relation.....	17
Figure 5.	Adjacent Relation.....	18
Figure 6.	Compartment Relation.....	18
Figure 7.	String 119S Compartments.....	19
Figure 8.	Pipes Database Menu	21
Figure 9.	Database Query Menu	21
Figure 10.	Piping System Queries	22
Figure 11.	Compartment Prompt	22
Figure 12.	System Response to Query.....	23
Figure 13.	All Pipes Contained in 8-119-9-V.....	23
Figure 14.	Expert System Menu.....	24
Figure 15.	Problem Analysis Menu.....	24
Figure 16.	Fuel Oil Service Tank Problem Menu.....	25
Figure 17.	Prompt for Compartment Number	26
Figure 18.	Action Prompt.....	26
Figure 19.	Troubleshooting Solution	27
Figure A.1	D. C. Void Will Not Pump.....	31
Figure A.2	D. C. Void Pumps but Refills with Water	32
Figure A.3	Damage Control Void Will Not Flood.....	33
Figure A.4	Damage Control Void Overflowing.....	34
Figure A.5	Water in a Fuel Oil Service Tank.....	35
Figure A.6	Fuel Oil Service Tank Overflowing.....	36
Figure A.7	Fuel Oil Service Tank Losing Fuel.....	37
Figure A.8	Fuel Oil Storage Tank Overflowing.....	38

Figure A.9	Large Contaminated Tank Overflowing.....	39
Figure A.10	Fuel/Oil in a Damage Control Void.....	40
Figure B.1	Relational Diagram.....	41

I. INTRODUCTION

The failure of aircraft carrier through tank piping was identified in the mid 1970's as a difficult management problem. The deterioration of carbon steel piping, from continuous immersion in salt water, allowed the intercommunication of the ship's fuel tanks and damage control voids. The potential detriment of intercommunicating tanks includes:

- loss of boiler fires due to water in the fuel oil casualties
- inadvertent overboard discharge of fuel when pumping a contaminated D.C. void
- increased draft
- reduction in ability to counter flood
- lost reserve buoyancy

The complexity of CV's, intense operating schedules, and advancing ship age has made dealing with piping casualties time consuming and frustrating. Hours spent tracing piping diagrams and making false starts in resolving a problem has pointed out that a formalized approach to the problem would be worthwhile. The system developed in this thesis is directly applicable to CV's 59/60/61/62/63/64, and is readily adaptable to CV's 41/43 and CVN-65. The CV-67 and CVN-68 class side protection systems, construction materials(copper-nickel piping), and pipe joining techniques(socket couplings) do not experience the same failures and hence these ships would derive minimal benefit from this system.

The development of an expert system interfacing a large database to aid in troubleshooting aircraft carrier piping systems is divisible into three areas of emphasis:

- the model for the database system
- the design of the interface to the database
- the expert system itself

This thesis examines each of these issues and implements the recommended solution in an operable database interfaced expert system.

The implementation of the expert system is designed to be operated by Engineering Department personnel at sea on a micro computer likely to be readily available. The hardware limitations therefore are considered to be those of the Z-248, widely purchased for operating forces, and readily available as a GSA catalog item.

CDR Clayton developed the decision trees and provided the expertise with respect to aircraft carrier design and construction. The data model and user interface to the system were also designed by CDR Clayton.

LT Boozer developed the expert system, the interface to the data base, and the data base system.

II. BACKGROUND

The following are Naval terms which may need to be defined to assist in the comprehension of this paper:

- "through tank" refers to long sections (40 - 50 feet) of pipe which run in the bottom of the ship through several adjacent compartments. These compartments are frequently other tanks, hence the label "through tank".
- "string" refers to a group of tanks and voids all located side by side at the same frame number and on the same side of the ship. For example, string 119S refers to five adjacent voids and tanks all located at frame 119 on the starboard side of the ship.
- "frame" is a relative location along the length of the ship starting at the forward end of the ship. Aircraft carrier frames are spaced at an interval of four feet, and are numbered sequentially bow to stern.
- "wing tank" is a tank or void, outboard of the holding bulkhead, away from the center of the ship and in close proximity to the side of the ship. They are typically long (20 feet), narrow (5 feet), and very deep (37 feet).

A. INTERNAL ARRANGEMENT

Understanding the internal compartment and system arrangement in a FORRESTAL/KITTY HAWK class aircraft carrier is fundamental to comprehending the database model utilized by the expert system. An illustration of a typical arrangement [Ref. 1:p. 10] appears as Figure 1.

Piping emanating in the engineering spaces passes through several compartments before reaching its termination point. Aircraft carriers built prior to 1965 were constructed with mild carbon steel fuel oil transfer, fuel oil service, and fuel oil recirculating piping systems. The deterioration of these systems in service due to corrosion has resulted in leaks which are not readily apparent because they are inside of other tanks. This internal leakage is thus difficult to

diagnose. Other factors which contribute to the difficulty of diagnosing a problem are improper system operation by personnel, inoperative or leaking valves, foreign object blockage, and cracks in structure. Multiple combinations of the above factors can mask one problem from another and further complicate troubleshooting. This latter case presents the most difficult challenge to the expert system: successfully identifying more than one problem when multiple problems are present. The design of the expert system can accommodate this type of scenario with multiple independent user sessions, but careful decision tree structure can minimize these instances and identify more than one cause to a problem in a single session.

B. TROUBLESHOOTING

The initial identification of a tank/void system problem can come from:

- soundings
- tank level indicators
- water at the boiler front
- water paste tests
- requirement to strip excessively
- requirement to pump excessively
- overflowing air escapes/sounding tubes
- discharge from overboard piping
- excessive draft
- unable to pump

The decision making process in troubleshooting a piping system casualty begins with an input of what system is disrupted and what the initial symptom of the problem is. This information is brought to the attention of the Engineering Officer

of the Watch in Central Control, the work center supervisor in the oil/water lab, or the Damage Control Assistant.

The trouble shooting process begins based on the experience and intuition of the individual to whom the problem is addressed. This, of course, is not a constant. The approach to solving a problem can vary widely between individuals, some who may have experienced a similar casualty and remembered a prior successful solution to that problem.

A methodical process of elimination is the best approach to a solution, but thorough knowledge of the systems involved, the ship's construction, and accurate responses to questions are required to produce a least effort path to a solution. This is critical in an operating aircraft carrier because the demands of operating the ship in a normal state alone taxes the crew, correcting casualties quickly exhausts them.

Because of this last condition, the first step in troubleshooting is to get as much accurate information as possible, with as little physical effort as possible. Simply put, you conserve energy. Actions which fall in this category are:

- taking soundings
- examining logs
- making water paste tests
- reading tank level indicators
- reading drawings and system technical manuals

The next step, based on the above information, is to make simple tests using accessible equipment. These are:

- verify the correct line up of installed pumping/stripping systems in the machinery spaces
- verify use of remote operating stations

- disassemble small valves (4 inch or smaller) in machinery spaces/pump rooms
- listen to systems in operation
- inspect equipment/systems in compartments which are readily entered (no bolted access covers).

Further escalation of troubleshooting should only begin when a problem has been isolated to a likely set of causes. Action at this point is one of the following categories:

- open and gas free and inspect voids/tanks
- open and pump tanks/voids using portable equipment
- pump contaminated fuel/water out of the ship within the governing regulations for the location of the ship
- disassemble large heavy valves (greater than 4 inches) pumps, or other complex equipment

This sequence of actions is driven by conservation of assets and the need to minimize disruption of operating systems. The multiple possible paths to solving a unique problem, the varying level of experience of operators, and the importance of conserving assets, both time and personnel, clearly point to the need for an expert system capable of managing a complex object requiring a large number of facts stored in an organized database.

III. DESIGN AND IMPLEMENTATION

The design and implementation of the total "PIPES" system involved the investigation of multiple alternatives of how to structure the system to maximize its performance with respect to:

- model design and utility
- storage of data
- database query
- modification/update of database
- expert system power
- memory requirements
- speed
- connection of software

The following sections describe the evolution of the initial concept and understanding of the problem, through the decisions made in the development process, to the final configuration of the functional prototype system.

A. MODELLING THE PIPING SYSTEM

The initial concept of system design was to utilize a database management program to perform central program functions. This was primarily driven by the feeling that the difficulty in building the system would center around the details of the configuration of the model. The problem was viewed as a challenging database problem. This conclusion was drawn from the intimidating size and multiple attributes thought felt to be required to deal with a large object such as an aircraft carrier. There was also a defined set of queries which were clearly of the DBMS type. The intent was to solve the model problem, build a corresponding database

system, and then utilize an expert system to return troubleshooting problem solutions to the database system. The final configuration of the system, after evolving through the development cycle, is discussed in the Conclusions chapter.

In making the choice of how to build the database model, consideration was given to traditional database structures. A hierarchical system offered no apparent utility in exploiting the construction of the ship since there is no hierarchy among piping systems or the pieces of a piping system. The compartments within a ship could be hierarchically arranged by deck and by position from forward to aft in the ship but this did not provide any apparent advantage in dealing with the piping systems, so a hierarchical system was rejected. A network system appeared to be feasible but while attempting to establish the links in the data structure diagram for such a system it became clear that it would be easier to implement a relational model.

The alternative of implementing the relational model in the expert system could have been accomplished, but would have required using an unmanageable number of facts in a Prolog system. This was judged to be prohibitive in terms of both memory requirement and speed of execution. Building a similar system in Pascal would have required an even larger quantity of code and would have again been a poor design choice for memory and speed reasons.

A design decision was made to implement the database in a DBMS language to attempt to exploit the relational model which had been developed and which was thought to offer considerable potential because of its simplicity and apparent flexibility.

B. EXPERT SYSTEM

The expert system function of the "PIPES" system is required to return solutions for specific problems selected by the operator. This meant that at least one fact is known at the outset of the session. One control structure for this type of rule based system is referred to as forward chaining. Essentially the expert system is given a fact and it then attempts to find a chain of facts which lead to a definitive conclusion. Control structures for expert systems are often combined to take advantage of the characteristics of each structure while compromising on the limitations brought with both structures. One form of such a combined control structure is called rule-cycle hybrid. Strictly defined, rule-cycle hybrid structures cycle through rules, in order, as in backward chaining, however, as facts are asserted they are added for use in the next cycle through the rules, as in forward chaining. [Ref. 2:p. 105]

The nature of the problem solving done in troubleshooting shipboard piping systems led to the development of a system which employed a decision tree design where the entry point to a unique tree was a user selected problem. After entry into the tree, the user is directed to carry out troubleshooting action and then respond to questions as to the outcome of his investigations. In this manner virtual facts are established, as in forward chaining, through a series of user actions and responses which lead down the tree to a conclusion. Each rule which succeeds (establishing a virtual fact) thus leads to another rule which in turn must succeed (establishing another fact) to reach a conclusion. The design decision of how to connect the expert system to the database system presented the most difficult challenge in building the system. The available database programs provide no capability to make a call to another program, and return to the database program. A major

design change in the structure of the system was forced at this point of development. The details of this decision follow. The interface of a expert system to DBMS files can be accomplished by calling a specific data file from within the expert system. This would mean that none of the DBMS functions would be available for query or file modification without quitting the expert system and loading the DBMS. The alternative of loading the DBMS each time it was needed, and then reloading the expert system, though feasible, was regarded as an undesirable degradation, from an operator's performance perspective.

An obvious alternative was to utilize a more advanced machine and run the expert system and DBMS simultaneously with a multi-processor, allowing queries to the DBMS without terminating the expert system. This method was judged unsatisfactory because of the requirement to be able to operate the system on shipboard available equipment, which at best would be 80286 processor based.

Because the expert system can make calls to external data files, the feasibility of calling compiled DBMS program queries was examined. A limited number of software routines which would perform some DBMS functions were identified but not used because of the limitations on the nature of queries and prohibitive dollar cost. The potential performance improvement offered by this approach was a significant increase in speed over DBMS commands due to the machine language configuration of the already compiled routines. A further option was to write drivers in the DBMS program language to perform all of the required calls and returns from DBMS. By essentially duplicating explicit DBMS functions, the DBMS files could be queried and/or manipulated to return a response without the need to carry all of the DBMS's operating overhead and memory requirements. The drivers would, as compiled routines also did, significantly speed the response

of the database side of the system. Decomposing DBMS program code and writing appropriate routines was not in the scope of this thesis. The final design choice was a compromise to obtain the desirable modelling and data storage of the DBMS system and the efficiency of a Prolog expert system. The connection of the expert system to the DBMS was made by running the system from the DBMS system program and accessing the expert system by calling the already compiled executable Prolog file. The key to making this choice was recognition that the full Turbo-Prolog program was a compiled executable program [Ref. 3:p. 160] which could be run inside the d-BASE III program [Ref. 4:p. 208] and not exceed the 640K resident memory limitations of the hardware. An additional design decision was made to allow queries in the DBMS side of the system to be made both by using functions built for the DBMS program, and by a program feature provided to allow user built queries in d-BASE III, enabling full exploitation of the large database. This meant that some operator involvement was accepted to allow more complex DBMS queries to be made.

C. SOFTWARE

D-base III was selected as the database implementation software because it supports the relational database design. The use of the relational database was fundamental to the development of a useful model of the ship and its internal systems. D-base III is readily available to the potential users of the system and is relatively inexpensive. Prolog was chosen for the expert system because it was designed for artificial intelligence applications. Prolog solutions are arrived at by logically inferring one thing from something that is already known. A Prolog program is not a sequence of actions, but a collection of facts together with the rules for drawing conclusions from those facts. Prolog more closely follows

thinking than procedural programming languages, because it is a declarative language. A Prolog program for a given application will typically require only one tenth as many program lines as the corresponding Pascal program.

Turbo-Prolog (Version 2.0) was selected for use in the implementation of the expert system because it was the latest and apparently best product available for use on the mandated IBM compatible hardware. It is a fifth generation language and, like d-BASE III, is both economical to purchase and readily available to potential users of the system developed in this thesis.

D. SCOPE OF PROTOTYPE

The development of the expert system to do troubleshooting of through tank piping system problems first required a problem statement of those casualties which the system must be able to solve.

A decision was made to limit the scope of the expert system to those casualties experienced in the CV side protection system (wing tanks). Although the troubleshooting solution to these types of problems often extends into the machinery spaces and pump rooms, the initial problem areas dealt with by the expert system are those found among the menu items below as choices which are presented to the user:

- Damage Control Voids
- Fuel Oil Service Tanks
- Fuel Oil Storage/Ballast Tanks
- Contaminated Tanks
- JP-5 Tanks

All of these menu selection tanks are wing tanks. This selection is the first decision a user is required to make in operating the program.

The development of the logic for initial symptoms of casualties is done in decision trees which are then coded in Prolog. Appendix A contains the logic decision trees for the implemented casualties.

The menu items below appear depending on the selection of the tank type problem from the list of tanks above. Thus, the casualties handled by the system are:

1. Damage Control Void

- Will not pump
- Pumps but refills with water
- Will not flood
- Overflowing
- Oil in a void
- Sewage in a void

2. Fuel Oil Service Tank

- Water present
- Overflowing
- Foreign particles
- Losing fuel
- Gaining fuel

3. Fuel Oil Storage Tank

- Water present
- Overflowing
- Foreign particles
- Losing fuel
- Gaining fuel

4. Contaminated Tank

- overflowing
- will not pump

5. JP-5 Tank

- water present
- overflowing
- losing fuel
- will not strip
- will not pump

E. IMPLEMENTATION OF THE RELATIONAL MODEL

Modelling a complex physical object is a principle challenge in designing many database systems. The creation of a satisfactory model of the multiple and extensive piping systems in an aircraft carrier was a primary area of research for this thesis.

At the outset of implementing the expert system, one approach would have been to have used individual Prolog facts to describe the components of each system down to the requisite level of detail required to accomplish troubleshooting. This approach, while feasible, was judged to be unacceptably costly in memory requirements and execution speed. Simply stated, the number of facts was too large.

The power of a database language was needed to structure, manipulate, and query the database in a manner which would take advantage of the properties of the system being modeled.

Initial examination of Entity/Relationship models appeared to require considerable complexity to successfully model the system and its attributes. The requirements were to be able to uniquely identify each section of pipe within the

ship, to include location, system, and physical properties such as size and material composition. Because the troubleshooting function of the expert system is concerned with through tank piping, a model was developed which could be reduced to just four relationships, many fewer than was anticipated. All the requirements could be met by careful placement of the attributes with the right relationship in the model, enabling the use of a surprisingly simple scheme. Although the model employed is fully adequate for this expert system, as implemented, it would require additional refinement to be expanded to model a machinery space or pump room. Because of the size of a main machinery room, simply identifying a pipe as being in the compartment is insufficient detail to be able to constructively utilize the model. An additional attribute is needed, for example "piping segment number" (piece number). This new attribute would be made up of the forward most frame number of a pipe within a compartment coupled to a port/starboard sequence number thus accommodating multiple pipe segments within a large compartment.

The relational database developed and implemented in dBASE, uniquely identifies each pipe in the ship by the compartment number it is contained in, and the system that it is a part of.

Thus the relation diagram (Figure 2) reduces to just four relations.

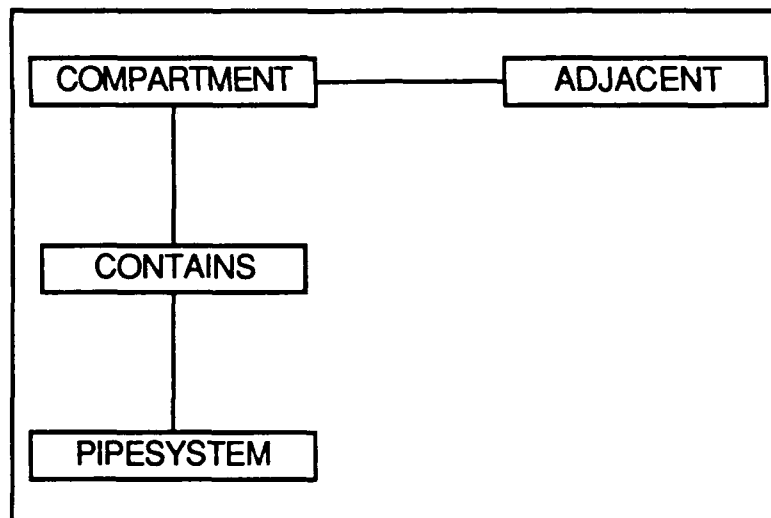


Figure 2. Basic Relation Diagram

The CONTAINS relation (Figure 3) is keyed by compartment number to each pipe within that compartment.

COMPARTMENT	SYSTEM	NUMBER
8-119-9-V	FOS	8-119-11-F
8-119-9-V	FOT	8-119-13-F
8-119-9-V	FOT	8-119-11-F

Figure 3. Contains Relation

represents a portion of the database describing which pipes are actually physically located in compartment 8-119-9-V. The PIPESYSTEM relation (Figure 4)

System Number	Pipe System	Size	Material	Couplings
FOS 8-119-11-F	Fuel Oil Service	4	steel	N/A
FOT 8-119-11-F	Fuel Oil Transfer	5	steel	N/A
FOT 8-119-13-F	Fuel Oil Transfer	5	steel	N/A

Figure 4. Pipesystem Relation

depicts the noun name of the system, the size (diameter) of a pipe in inches, the material composition, and the type of joint make up used. These attributes are used

to maintain a current database for the configuration of the ship to support long term maintenance planning. The repairs include the replacement of deteriorated carbon steel piping, hence the material attribute, with copper nickel piping and the change of troublesome sleeve couplings with those of socket design, thus the coupling attribute. The size aids in the identification of a pipe when a tank/void is opened and inspected.

The ADJACENT relation (Figure 5) locates

Compartment	Forward	Aft	Starboard	Port	Above
8-119-9-V	8-114-9-V	8-124-3-F	8-119-11-F	8-119-7-V	4-119-5-V

Figure 5. Adjacent Relation

locates the compartment in the ship with respect to the other compartments and is used in troubleshooting logic and in maintenance planning to predict access and gas free requirements.

The COMPARTMENT relation (Figure 6)

String	Usage	Compartment	Date Paint	Date Completed
119S	Void	8-119-9-V	4-83	4-85

Figure 6. Compartment Relation

identifies the string which a compartment is a member of, and records historical maintenance data pertinent to the entire compartment. The complete relational diagram appears in Appendix B.1.

Thus an apparently complex modeling problem was reduced to its fundamental relationships in a powerful relational database. The central relationship in this model and the basis for its power is the Contains relation. By subdividing the

aircraft carrier down to compartments, the common building block of the model, it becomes possible to depict the entire ship or only an area of the ship in which you are interested. In this thesis, for example, we only are interested in the fourth deck and below compartments. Thus everything above 45 feet above the keel (the height of the fourth deck) is not present in the database, because it is not relevant to through tank piping.

By utilizing the Adjacent relation and the Contains relation it is possible to trace a pipe through the entire ship. For example, if compartment 8-119-9-V is a suspected problem void, the Adjacent relation tells us that there are compartments 8-119-7-V and 8-119-11-F inboard and outboard respectively of the problem void. The Contains relation then tells us that pipe system FOT 8-119-13-F (actually a section of pipe) is contained in each of three voids/tanks. We can thus trace this pipe through at least 3 compartments. If we look at the Pipe_System relation we find that FOT 8-119-13-F is a fuel oil transfer pipe, 5 inches in diameter, and made of carbon steel. The Compartment relation tells us that 8-119-9-V, the original problem void, is in string 119S. Other compartments in string 119S appear in Figure 7.

8-119-1-W	8-119-9-V
8-119-3-W	8-119-11-F
8-119-5-F	8-119-13-F
8-119-7-V	8-119-15-V

Figure 7. String 119S Compartments

A check of Contains for these compartments reveals that pipe system FOT 8-119-13-F originates in 8-119-5-F, passes through 7-V, 9-V, 11-F, and terminates in 13-F. The value of the compartment relation is that it identifies the

compartments in a string. From a given compartment, a string could be built by multiple calls to adjacent. Providing the relation minimizes repetitive manipulation of the database to obtain a frequently needed and useful fact. The relationship that may not be apparent is that most piping runs, run athwartship within the boundary of a string. The database design takes advantage of this property easing the modeling of a piping system by speeding the location of other compartments containing a section of pipe belonging to a specific system. Summarizing, the four relations contribute to the utility of the model as follows:

- **CONTAINS:** identifies unique pipes in a compartment by pipe system
- **ADJACENT:** locates a compartment within its surrounding compartments providing the mechanism for the building block concept in the model
- **COMPARTMENT:** identifies the string a compartment is in, useful in that it relates a small group of compartments adjacent to each other within the ship
- **PIPE SYSTEM:** allows the attributes of an entire system to be carried in a single tuple, rather than repeated for each compartment

F. PROTOTYPE DEMONSTRATION

The initial step in operating the system is to start "PIPES". The following screen displays will provide a demonstration of the steps required to operate both the d- BASE III portion of the system and how to enter the troubleshooting mode of operation performed by the expert system side of the system.

1. Database System Operation

The first menu (Figure 8) presented to the operator from the database offers a choice system functions.

PIPES DATABASE	
Add/Edit Database Record	A
Query Database	Q
Print Database Records	P
Backup Database	B
PIPES - Expert System	E
Select Option	
Press ESC to EXIT	

Figure 8. Pipes Database Menu

If the user desires to query the database, for an example, he would type "Q", which would bring up the database query (Figure 9) menu

PIPES SYSTEM QUERIES			
Compartment Access	C	Strings	S
Pipe Systems	P	Tanks by type	T
Adjacent tanks	A	Inboard Tanks	I
List of paint dates	L	Unlisted Query	Q
Select Option	A		
Press ESC to EXIT			

Figure 9. Database Query Menu

From the database menu the user presses the appropriate letter key. If he desired for example, to know the pipes in a compartment he would press "P". The piping system query menu (Figure 10) would appear:

PIPE SYSTEM QUERIES	
Pipes passing through compartment	P
Compartments containing pipe system	C
Specific pipe system material	S
List of pipe systems by material	M
Select Option:	
Press ESC to EXIT	

Figure 10. Piping System Queries

If it is desired at this point to know the specific pipes in a compartment, the user presses "P", which prompts him for the compartment number desired (Figure 11).

COMPARTMENT TO QUERY
Compartment No 8-119-9-V
Press ESC to EXIT

Figure 11. Compartment Prompt

The compartment number is entered by the user, as in the example above, "8-119-9-V" has been entered. The d-BASE III program at this point has sufficient input to conduct the query and respond. (Figure 12)

PIPE SYSTEMS PASSING THROUGH COMPARTMENT 8-119-9-V
System Number: FOS 8-119-11-F Press <- to BROWSE Press ESC to EXIT Press HOME to Print

Figure 12. System Response to Query

The "Press <- to BROWSE" option allows the user to individually view the database contents for that query. The "Press HOME to Print" option is provided to print out all the pipes contained in the database for that query, as in Figure 13.

Pipes Passing Through 8-119-9-V
FOS 8-119-11-F FOT 8-119-11-F FOT 8-119-13-F BAL 8-119-13-F STR 8-119-11-F VOID 8-119-15-V FOR 8-119-11-F

Figure 13. All Pipes Contained in 8-119-9-V

2. Expert System Operation

The expert system is entered from the database system by selecting "PIPES-Expert System" from the top level database menu (Figure 8), by pressing the "E" key. The expert system will be activated and the PIPES-Expert System Menu (Figure 14) will appear.

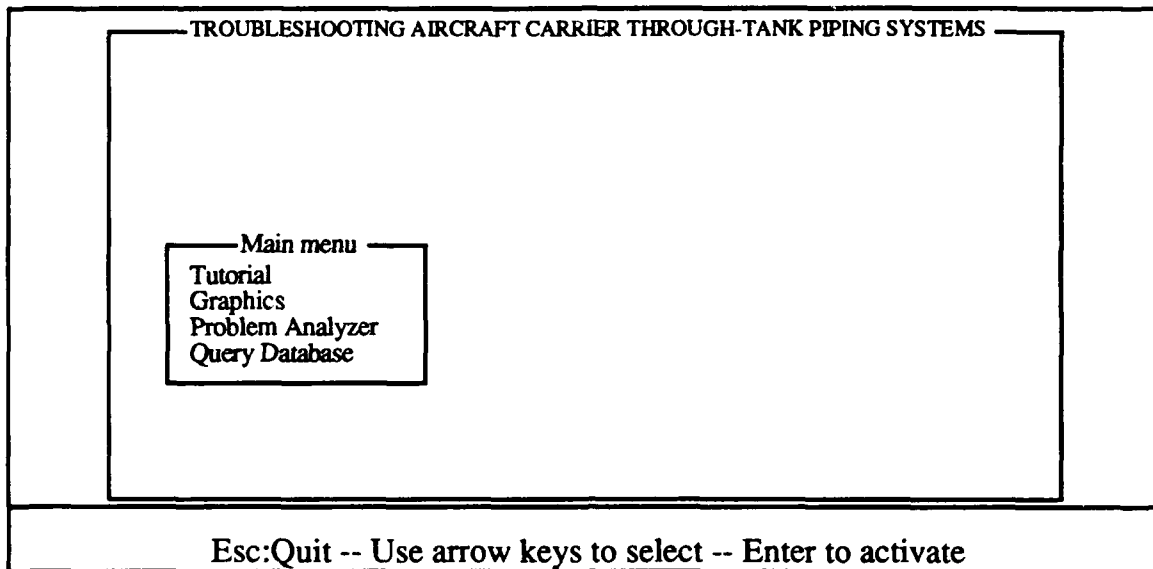


Figure 14. Expert System Menu

The operator must select "Problem Analyzer", using the arrow keys, if he desires to use the system for troubleshooting. If the "Problem Analyzer" is selected, the screen appears as in Figure 15.

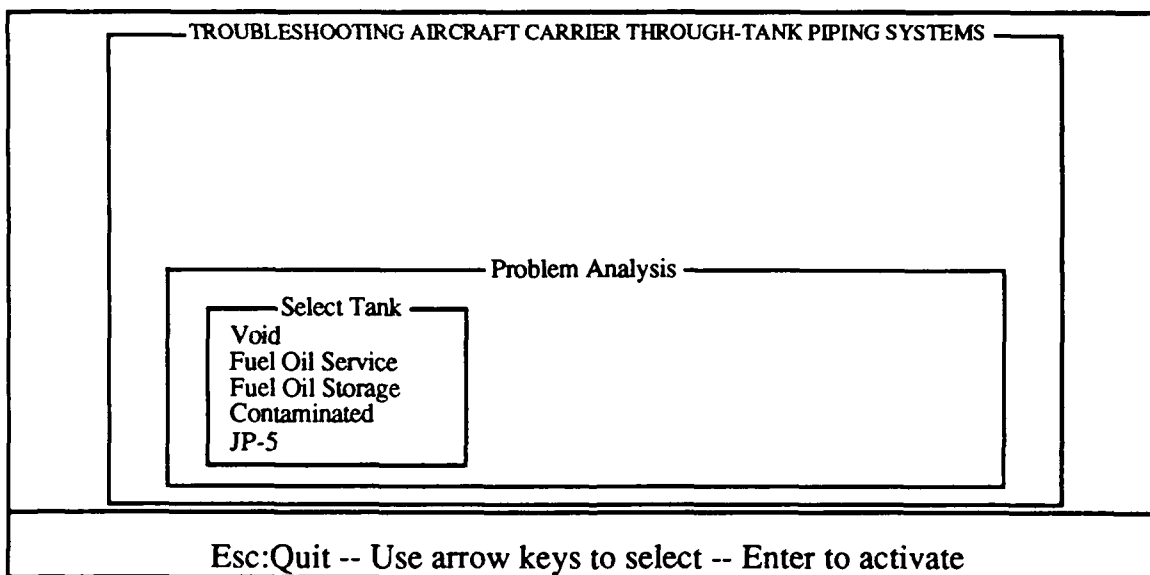


Figure 15. Problem Analysis Menu

From the above menu the user must select the type of tank in which the problem is being experienced. For example, if the problem is in a fuel oil service tank the user uses the arrow keys to select "Fuel Oil Service". The fuel oil service tank problem menu (Figure 16) will prompt the user to narrow the problem definition by selecting the nature of the casualty from the menu.

TROUBLESHOOTING AIRCRAFT CARRIER THROUGH-TANK PIPING SYSTEMS	
Action	
Problem Analysis	
Select Tank	
Water in the tank	
Overflowing	
Foreign particles	
Losing fuel	
Gaining fuel	
Esc:Quit -- Use arrow keys to select -- Enter to activate	

Figure 16. Fuel Oil Service Tank Problem Menu

If the problem is "water in the tank", the user selects this menu item with the arrow keys, and is prompted in the screen below for the compartment number of the problem tank.(Figure 17)

TROUBLESHOOTING AIRCRAFT CARRIER THROUGH-TANK PIPING SYSTEM
Problem Analysis What is the compartment number of the problem tank? 8-119-11-F
ESC: Quit --F1-- Print Action Block

Figure 17. Prompt for Compartment Number

The user must type in the correct compartment number of the problem tank, as for example, 8-119-11-F. Prompts will appear (Figure 18) directing the user to take action as indicated in the upper box labelled "ACTION", and then ask the user to respond to a question as to the outcome of the action taken. An example of such a screen is:

TROUBLESHOOTING AIRCRAFT CARRIER THROUGH-TANK PIPING SYSTEMS
Action Pump down fuel oil service tank 8-119-11-F and inspect 2 1/2 inch void suction line in service tank for cracked or leaking couplings Problem Analysis Are there any leaks in the 2 1/2 inch void suction line?
Esc: Quit --F1-- Print Action Block

Figure 18. Action Prompt

After performing the action requested, the user responded with a "yes", as indicated above, to the question asked by the system. The answer in this case is a conclusive one, and thus the system responds with the solution to the problem (Figure 19).

—TROUBLESHOOTING AIRCRAFT CARRIER THROUGH-TANK PIPING SYSTEMS— Solution The void suction line to the outboard void 8-119-15-V is ruptured in 8-19-11-F. Empty, clean, gas free, and repair the break in the 2.5 inch void suction line in 8-119-11-F Esc: Quit --F1-- Print Action Block
--

Figure 19. Troubleshooting Solution

IV. CONCLUSIONS

A. DATABASE SYSTEM/EXPERT SYSTEM CONNECTION

The development of an expert system which operates efficiently by employing a database system for data storage, retrieval, queries, and update has significant advantages. By using readily available software and hardware a valuable tool for fleet use can readily be built. The combination of d-BASE III and Turbo-Prolog, both popular economical software products, and a minimum 8086 or 80286 based, IBM compatible microcomputer produced a reliable, useful tool for assisting decision making in a complex environment. The advantages of combining the power of the database system and an expert system are:

- employment of relational data model by the expert system
- higher speed of execution for the expert system
- reduced memory requirements
- greater total system capability/flexibility
- ease of data file construction
- ease of file maintenance

The dBASE program provides the framework for building data files which enable a relational database model to be employed. The ability to build the relational model and readily implement it in the database assists understanding the organization and implementation of the expert system. Speed of execution is gained in the Prolog program by reducing the number of facts which the interpreter must process. This is accomplished by storing the facts concerning the configuration of an aircraft carrier in the database system, rather than as prolog facts. Only the

appropriate facts are called from the database and built as Prolog facts for use by the expert system.

Reduction of memory requirements is achieved by storing the facts in the database format rather than as prolog facts. The compact storage of the database files saves the replication of the predicate portion of each Prolog fact. The savings in memory storage for the configuration of the ship is 50 to 70 percent of that required for storing large numbers of Prolog facts in the predicate lists.

The utilization of dBASE queries can provide information from the database which can aid the user in decision making and fault isolation. The database by virtue of modeling the ship's piping systems has value beyond that of the expert system. The utility of the database is virtually that of a piping system diagram, and can assist in isolating systems, isolating compartments, damage control decisions, casualty control, and normal operation of systems.

The use of dBASE to maintain data files eases maintenance and construction of files. Because an instance of the database can be used to build more than one fact, updating the database for the single instance in the database, saves changing multiple predicates in the expert system. The organization of the database readily allows location and access of a specific instance needing modification.

B. REFINING THE PROTOTYPE

There are five areas in which the prototype could be further developed to expand its utility and scope of application.

- The relational model designed for the database system offers the flexibility to significantly increase the number of attributes, making the database a more powerful model of the detail of the ship. The potential exists to expand the database to store enough useful information as to define the exact configuration of the ship.

- For nine of the most important through-tank piping casualties the prototype is fully operational in a real environment. The expert system can be extended by the addition of the trouble shooting options offered in the two user problem selection menus but not installed in the system.
- The installation of this prototype requires the building of the database for each specific aircraft carrier to which it will be applied to. This capability is fully provided for in the prototype. It is accomplished by using program menus for file construction and maintenance.
- The implementation of a maintenance planning function in the expert system, to include decision making capability for establishing the priority of repairs and the development of schedules, would add a significant and useful dimension to the prototype system.
- The expansion of the graphical presentation of the ships piping systems is an important area in the design of the prototype.

C. POTENTIAL EXPERT SYSTEMS

The application of expert systems to shipboard decision making processes has extensive scope and potential for making significant impact on the reliability, correctness, and efficiency of daily shipboard operation. Expert systems, similar to the one developed in this thesis, could be built for any of the following unique engineering casualty control/management problems:

- loss of main engine vacuum
- condensate/feed system salt contamination
- evaporator troubleshooting
- O2N2 plant
- automatic combustion control system
- boiler water/feed water chemistry
- aircraft carrier multi plant engineering drills

The considerable contribution that such development and implementation would make to fleet readiness is worth further investigation.

APPENDIX A. DECISION TREES

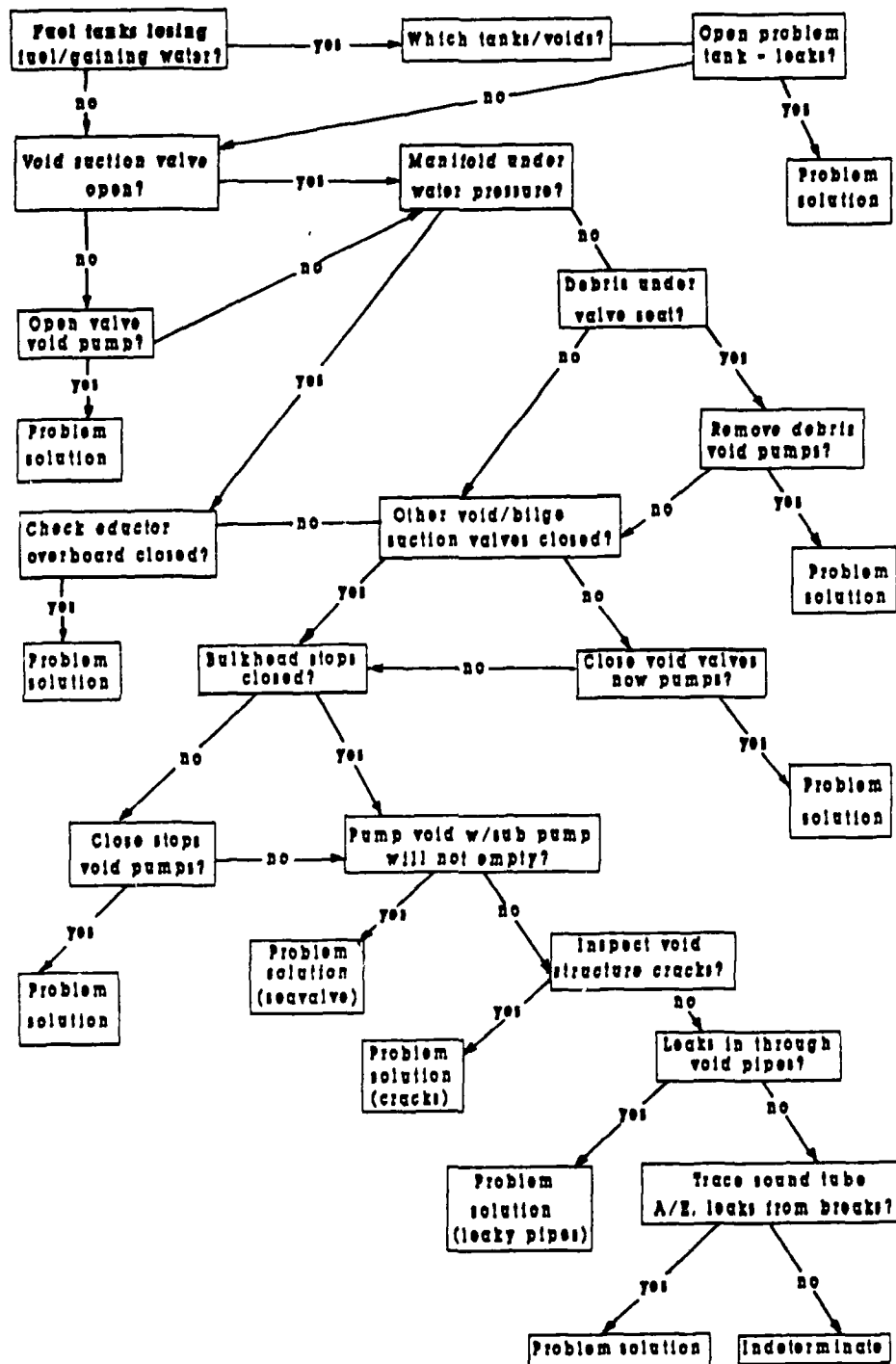


Figure A.1 D.C. Void Will Not Pump

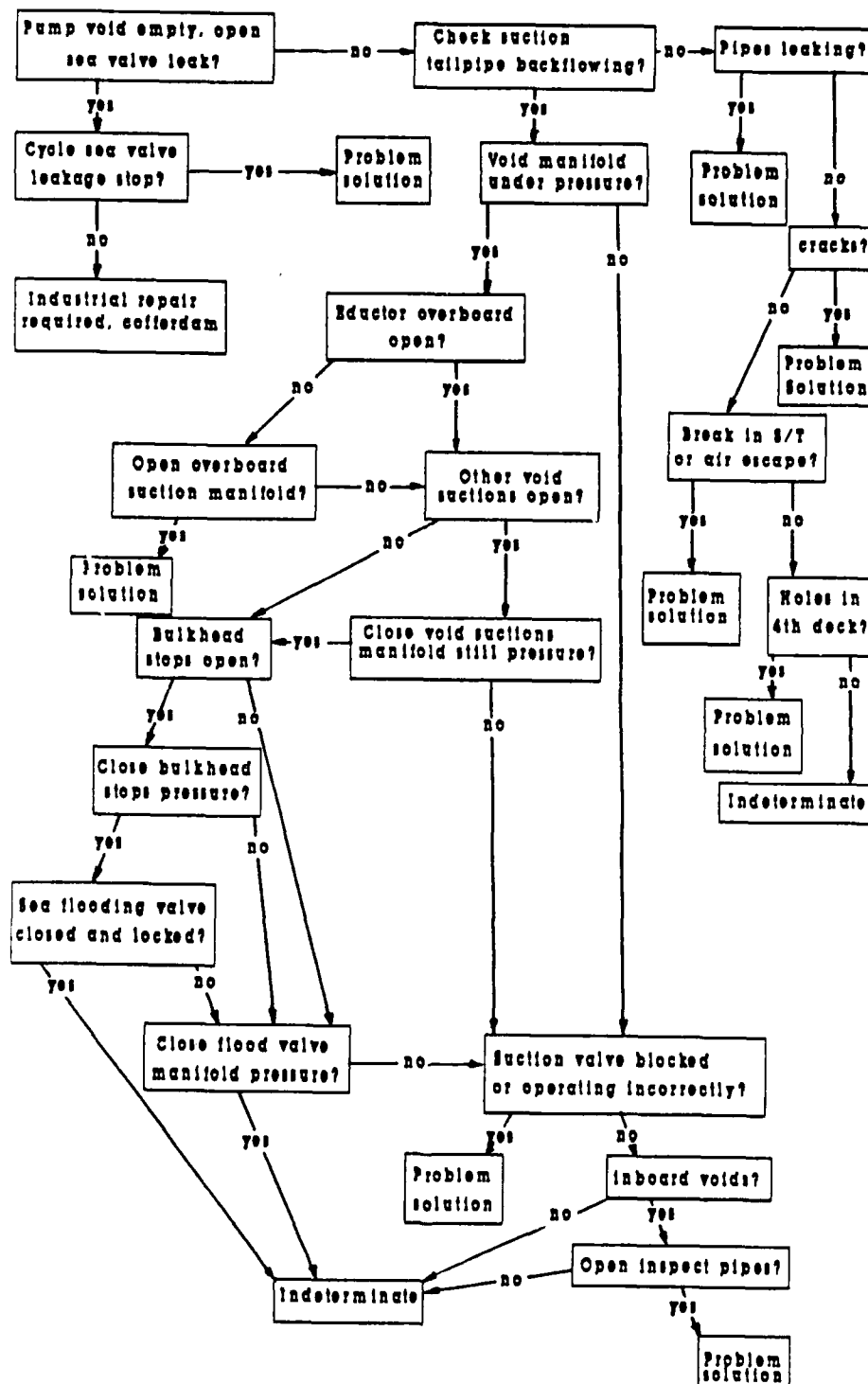


Figure A.2 D.C. Vold Pumps But Refills With Water

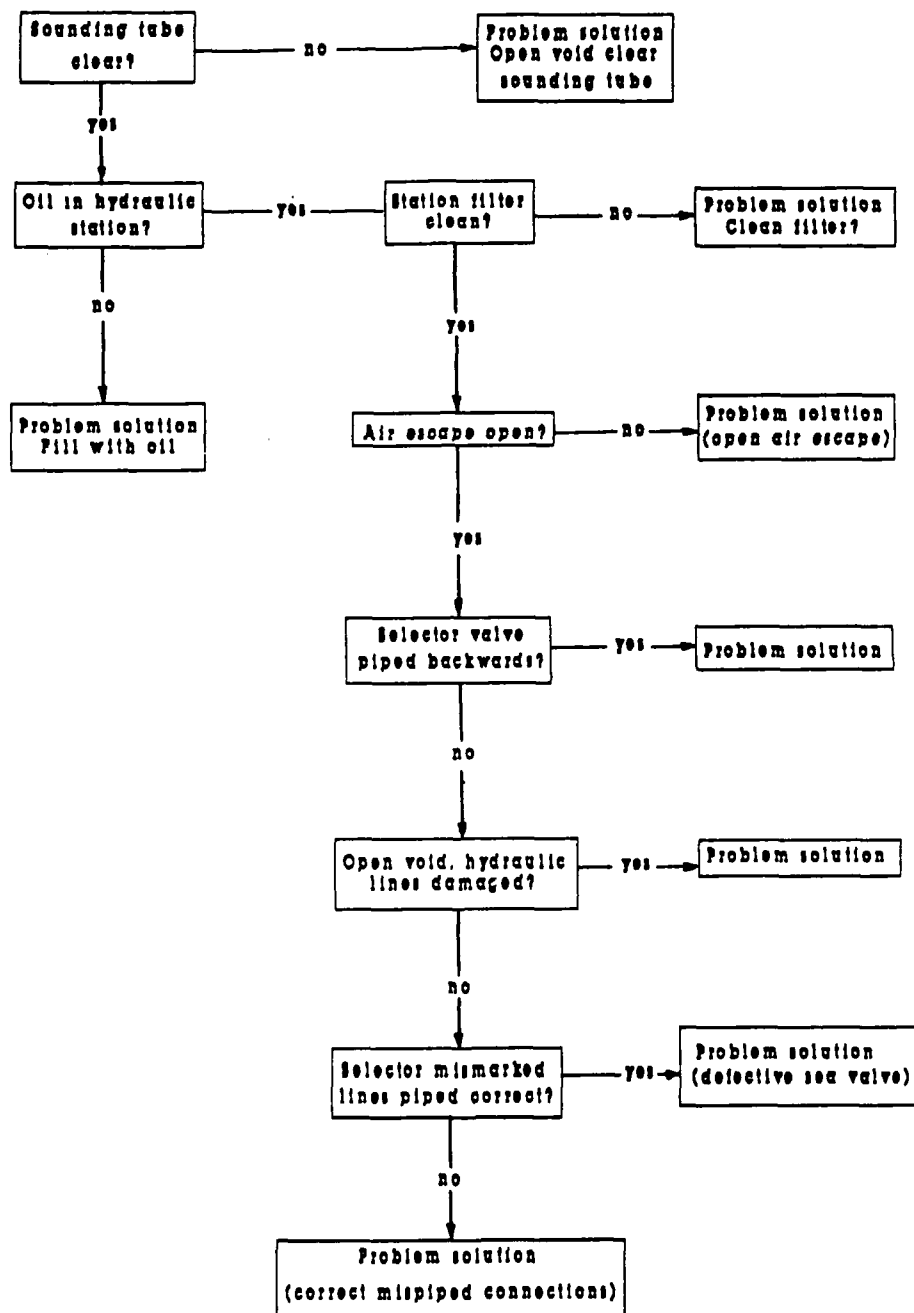


Figure A.3 Damage Control Vold Will Not Flood

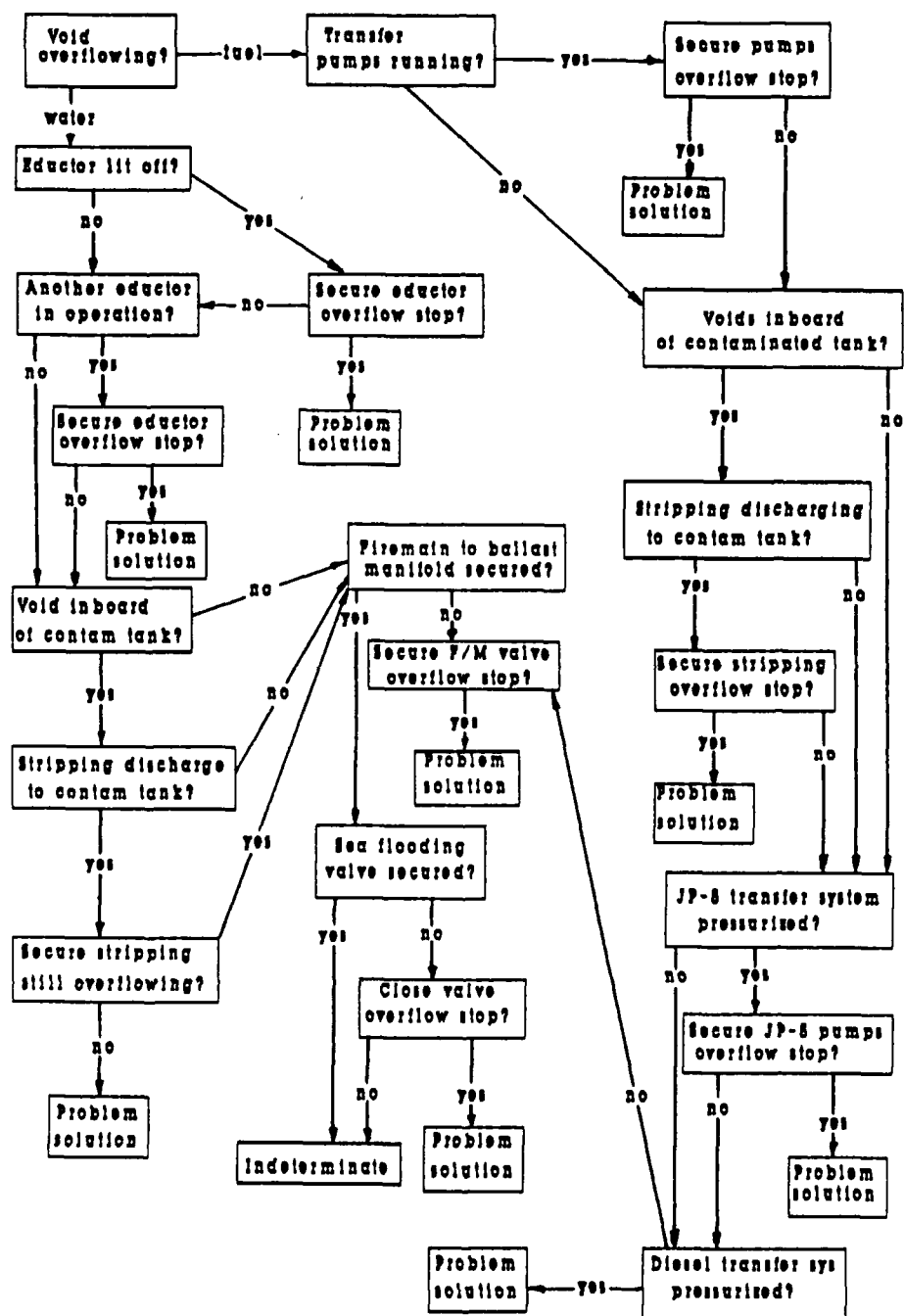


Figure A.4 Damage Control Void Overflowing

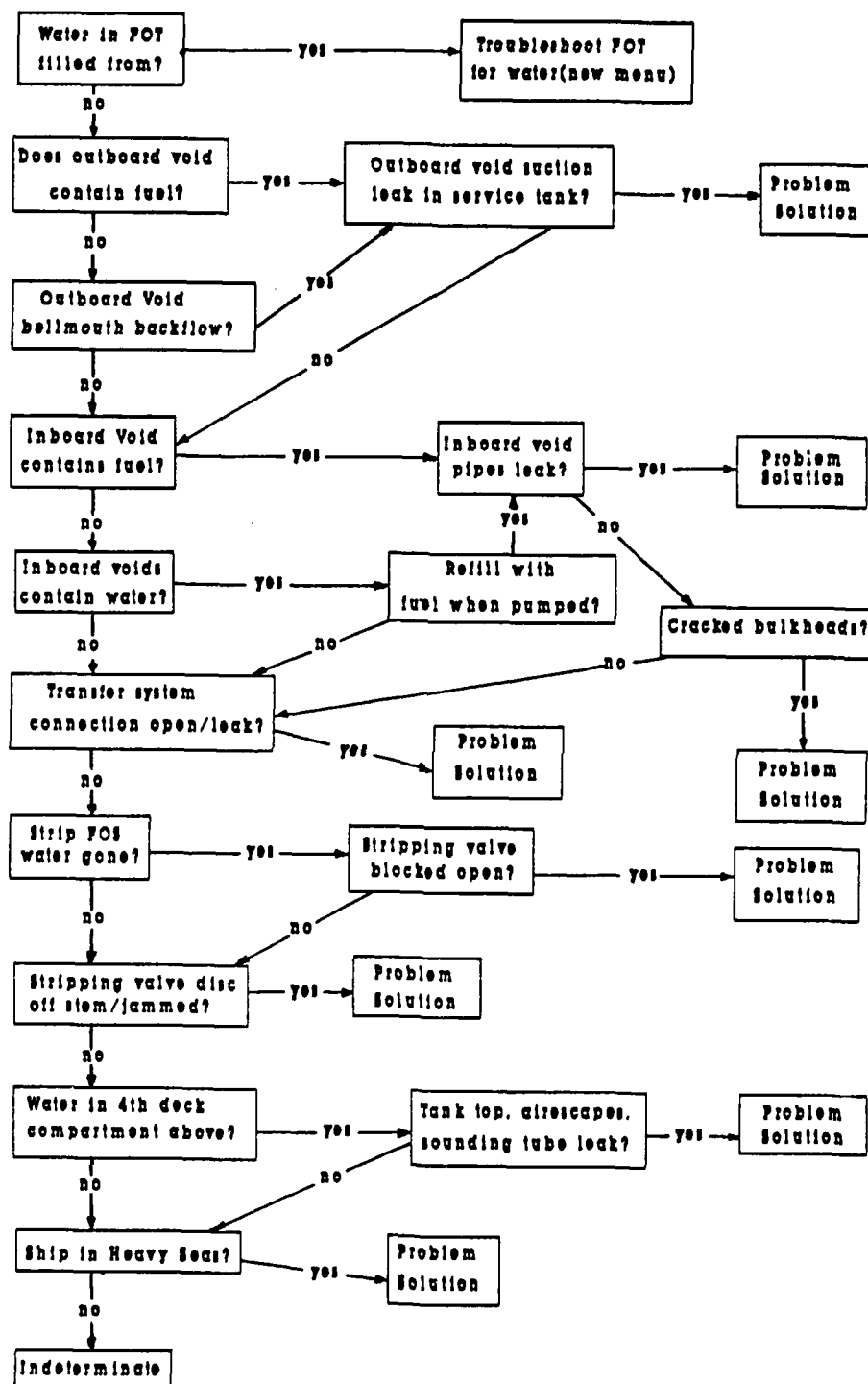


Figure A.5 Water in a Fuel Oil Service Tank

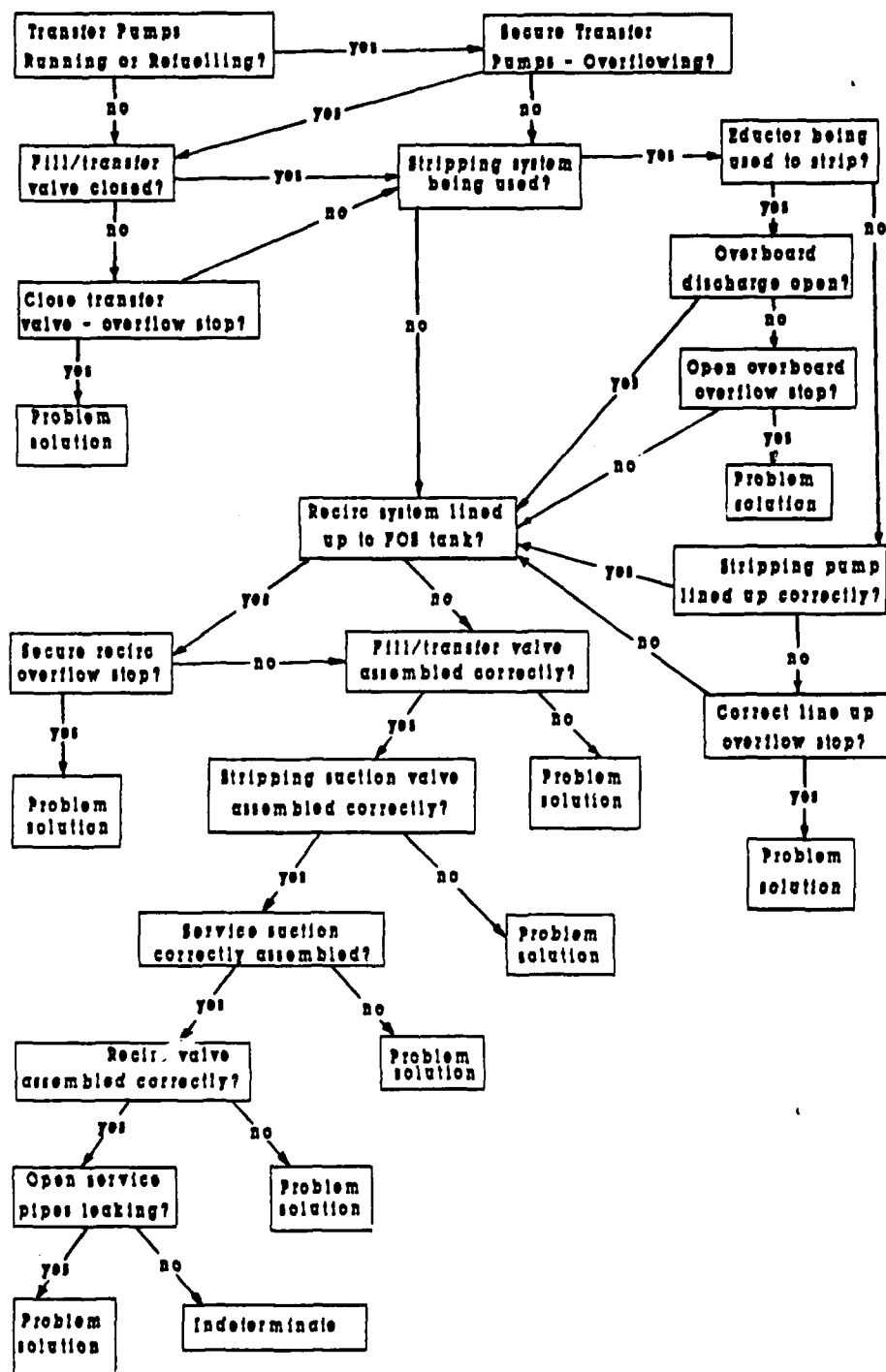


Figure A.6 Fuel Oil Service Tank Overflowing

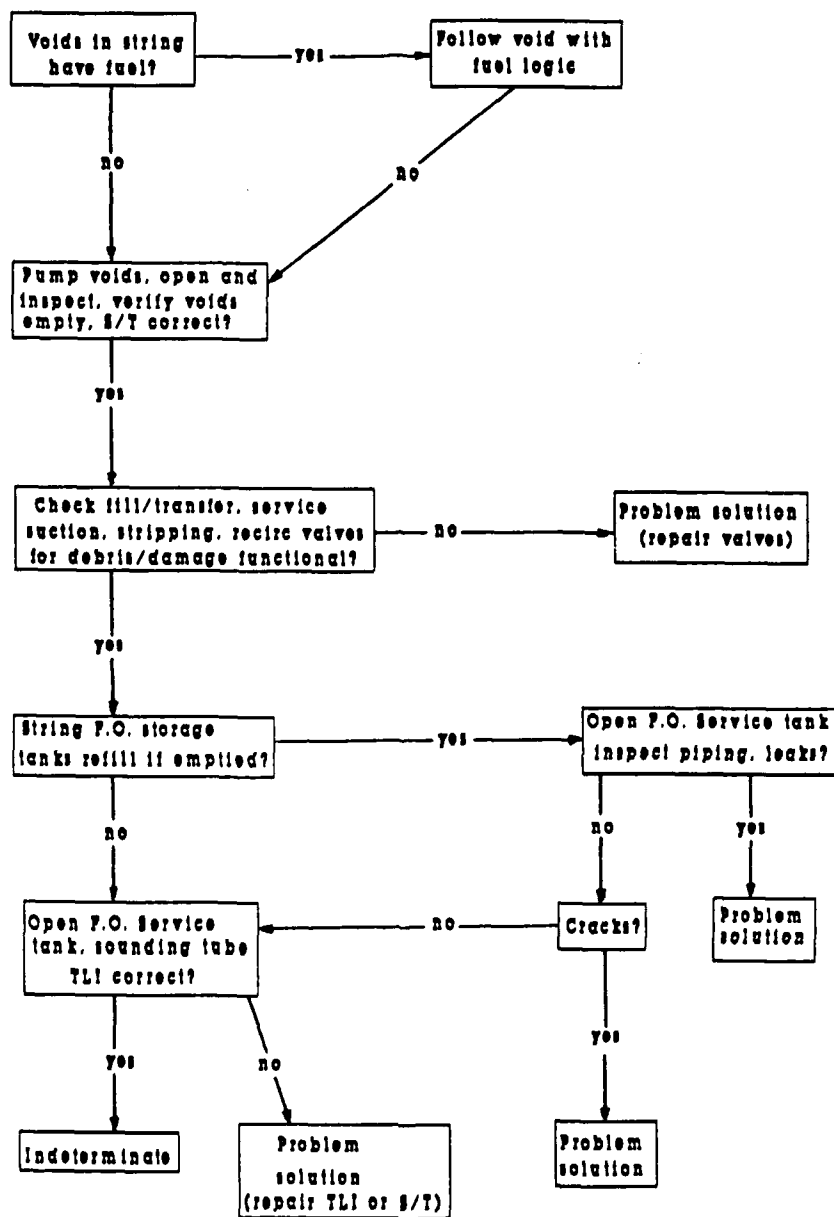


Figure A.7 Fuel Oil Service Tank Losing Fuel

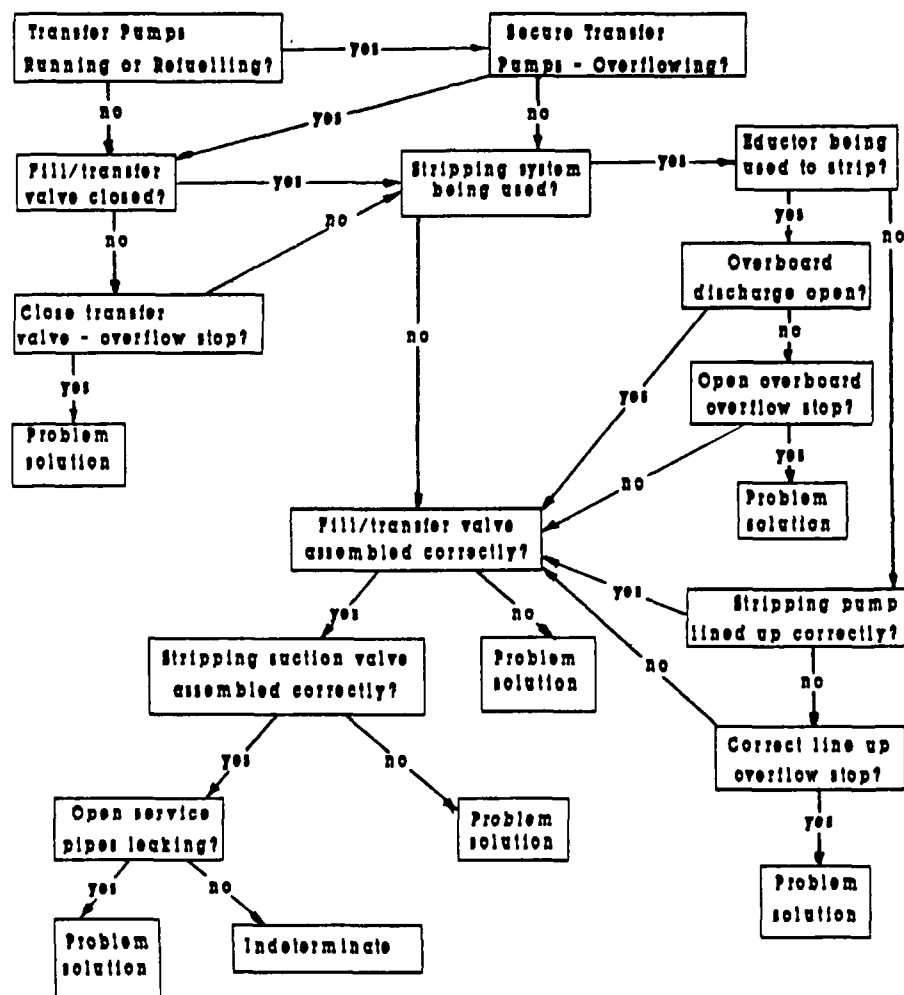


Figure A.8 Fuel Oil Storage Tank Overflowing

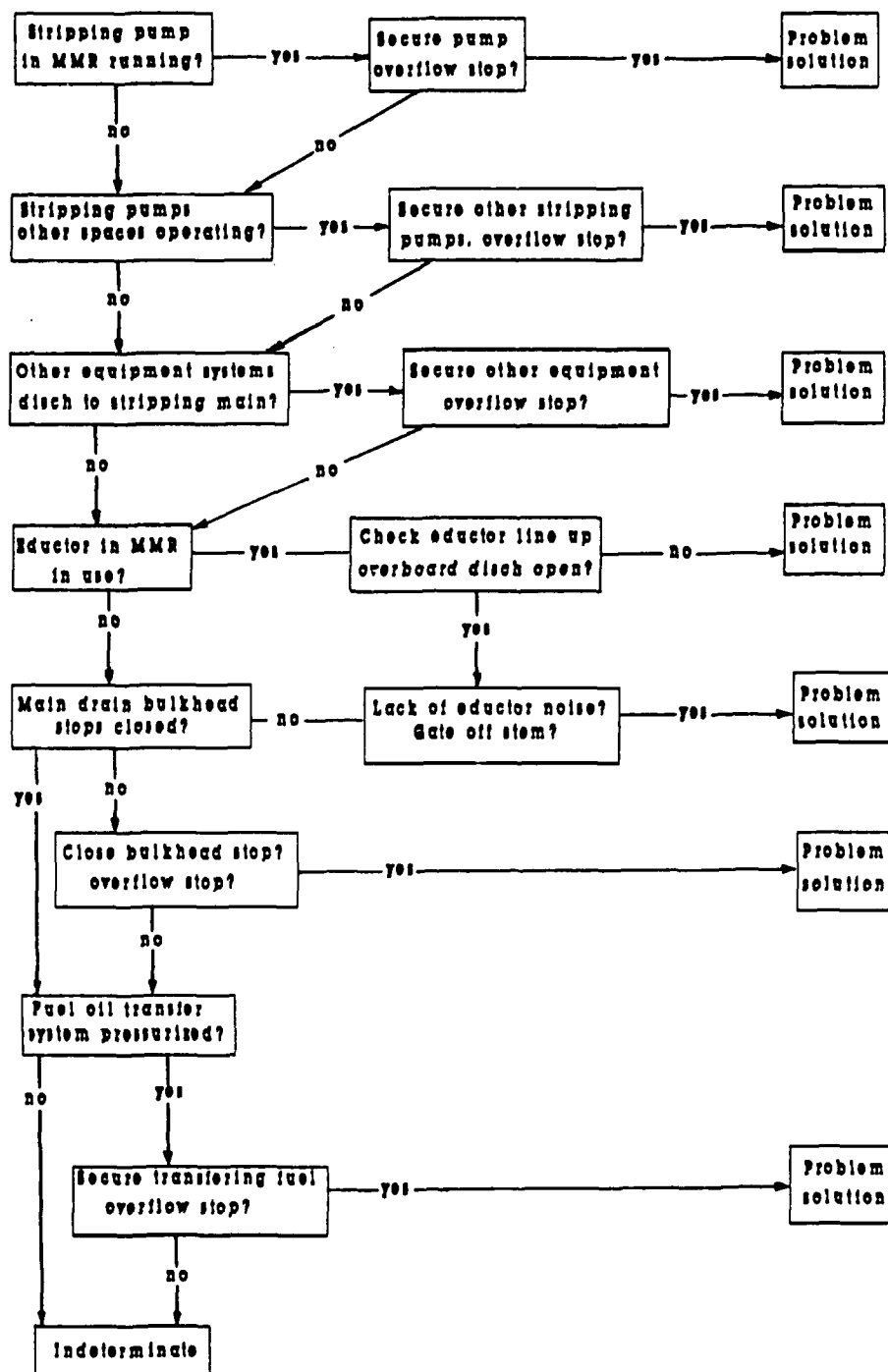


Figure A.9 Large Contaminated Tank Overflowing

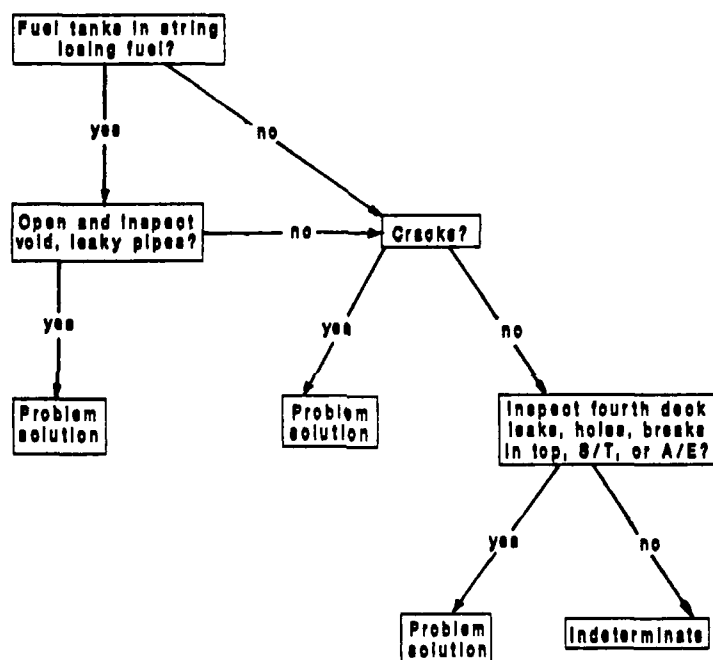


Figure A.10 Fuel/Oil In A Damage Control Void

APPENDIX B. RELATIONAL DIAGRAM

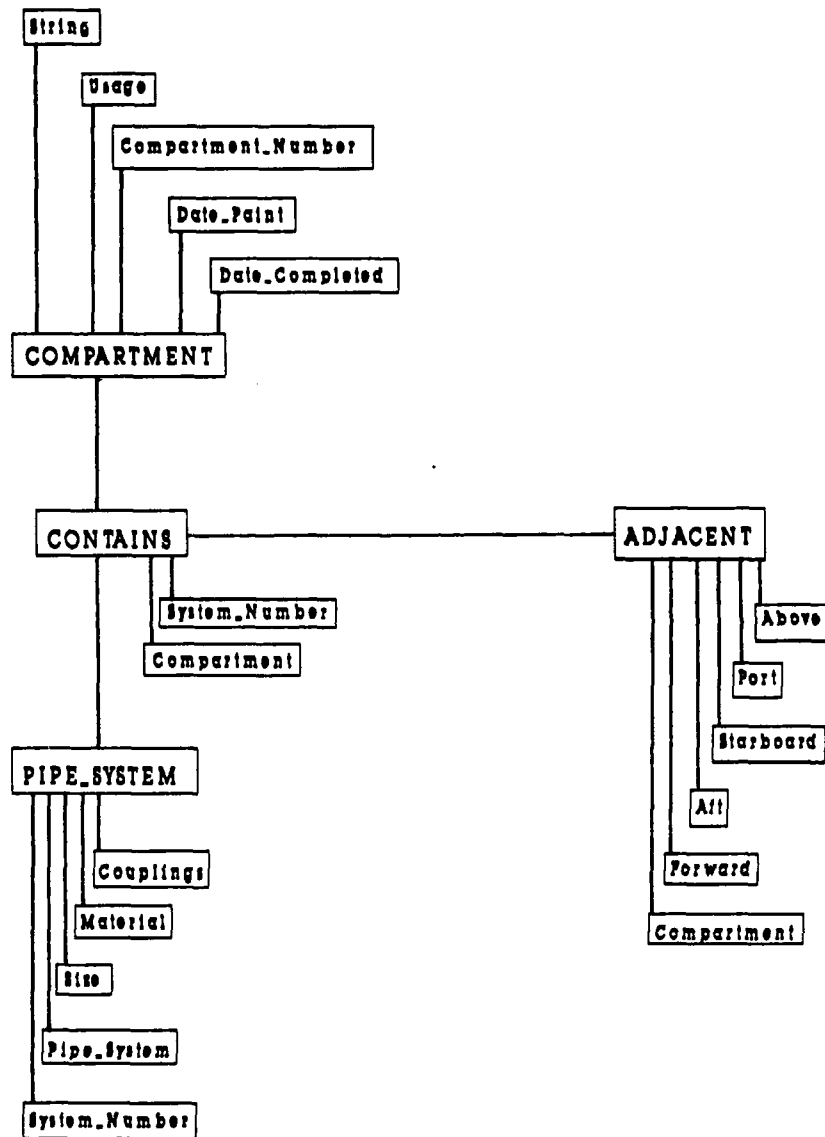


Figure B.1 Relational Diagram

APPENDIX C. PROGRAM LISTING

```

/*****
*
*      TROUBLE.PRO
*  Driver for the PIPES Expert system.
*  The following files are compiled and linked by the
*  Pipes.prj PROJECT FILE into PIPES.EXE executed from
*  PIPES BASE III PLUS System
*      DataBase.pro
*      Screen.pro
*      Question.pro
*      Solution.pro
*      VoidNoPm.pro
*      VoidOil.pro
*      VoidPump.pro
*      WaterFos.pro
*      WaterFot.pro
*      FosLosFl.pro
*      VoidNoFd.pro
*      FosOvrFl.pro
*      VoidOvrFl.pro
*      FotOvrFl.pro
*      ContmOvr.pro
*      Fact.pro
*
*
*****/

```

code = 4000

project "PIPES"

```
include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"
include "AnaGdef.PRO"
include "Pipegraph.PRO"
```

PREDICATES

```
mainmenu
proces(INTEGER)
drawtank(INTEGER)
evaluate(SYMBOL)
problemmenu(LIST,SYMBOL)
problem_type(SYMBOL,INTEGER)
determine(INTEGER)
contains_fuel
print_tanks(SYMBOL,SYMBOL)
find_string(SYMBOL)
retrieve_tank(SYMBOL,SYMBOL)
ans(CHAR)
convert_case(SYMBOL,SYMBOL)
get_problem_tank
port_stbd(SYMBOL,SYMBOL)
side(INTEGER,SYMBOL)
inboard_voids
outboard_voids
parse_tank_port_stbd(SYMBOL,SYMBOL)
find_inbd_voids(INTEGER)
find_otbd_voids(INTEGER)
assert_inboards(INTEGER)
assert_outboards(INTEGER)
assert_inboard_voids(SYMBOL)
assert_outboard_voids(SYMBOL)
troubleshoot
```

```
/******
*                               *
*          MAIN MENU          *
*****/
```

GOAL

```
pipecover,troubleshoot.
```

CLAUSES

```
troubleshoot:-
makewindow(21,78,0,"",24,0,1,80),clearwindow,write
    ("Esc:Quit -- -- Use arrow keys to
    select-- -- Enter to activate"),
```

```

    makewindow(22,78,0,"",24,0,1,80),
    write("Esc: Quit -- F1 -- Print
    Action Block"),
    makewindow(2,7,23,
    "TROUBLESHOOTING AIRCRAFT CARRIER THROUGH-TANK
    PIPING SYSTEMS",
    0,0,24,80),!,mainmenu.
mainmenu:- repeat,
shiftwindow(2),clearwindow,menu(3,14,47,14,5,"Main menu",
    [ "Tutorial",
    "Graphics",
    "Problem Analyzer",
    "Query Database"],
    CHOICE),proces(CHOICE),CHOICE=0,!,
    removewindow(2,1),removewindow(21,1).
proces(0):-retract_facts,makewindow(10,11,2,"EXIT",
    18,4,3,50),
    write("Are you sure you want to quit? (Y/N):"),
    readchar(ANS),
    ans(ANS), removewindow(10,1),!.
proces(0):-!,removewindow(10,1).

proces(1):-file_str("trouble.hlp",TXT),display(TXT),
    clearwindow,!.

proces(1):-!,write(">> trouble.hlp not in default
    directory\n").

proces(2):-repeat,shiftwindow(2),clearwindow,
    makewindow(4,78,7,"Piping
    Diagrams",13,5,10,70),
    shiftwindow(4),
    menu(5,47,7,14,7,"Diagrams",["Single Tank",
    "Tank String",
    "Zones","Tank Sections"],
    CHOICE),removewindow(4,1),
    drawtank(CHOICE),CHOICE = 0,!.

proces(3):-repeat,shiftwindow(2),clearwindow,
    makewindow(4,78,7,"Problem
    Analysis",13,5,10,70),
    shiftwindow(4),
    menu(5,47,7,14,7,"Select Tank",["Void","Fuel

```

1

4

1

2

•

evaluate(fuel_oil_service):- removewindow(7,1).

evaluate(fuel_oil_storage):-
 makewindow(7,47,7,"Action",1,5,12,70),
 assert_empty_fact!,
 problemmenu(["Water in the tank",
 "Overflowing",
 "Foreign particles",
 "Losing fuel",
 "Gaining fuel"],"fuel oil
 storage"),removewindow(7,1),!.

evaluate(fuel_oil_storage):- removewindow(7,1).

evaluate(contaminated):-
 makewindow(7,47,7,"Action",1,5,12,70),
 assert_empty_fact!,
 problemmenu(["High percentage of oil in
 tank", "Flooding with water",
 "Will not take suction",
 "Unable to fill",
 "Overflowing",
 "Draining without suction"],"contaminated"),
 removewindow(7,1),!.

evaluate(contaminated):- removewindow(7,1).

drawtank(0):-removewindow(4,1).
drawtank(1):-makewindow(5,47,7,"",5,5,11,70),
 shiftwindow(5),
 ask_ques_read_ans(TANK,"single tank"),
 SingleTank(TANK),removewindow(5,1),!.
drawtank(1):-retract_facts,removewindow(5,1).

drawtank(2).

drawtank(3).

drawtank(4).

problemmenu(LIST,MENU):- shiftwindow(4),clearwindow,
 menu(6,30,7,14,7,"",LIST,CHOICE),
 shiftwindow(4), clearwindow,
 !,problem_type(MENU,CHOICE),
 removewindow(4,1),!.

```

problemmenu(,_):-retract_facts,removewindow(7,1).

get_problem_tank:- ask_ques_read_ans(FOSERVTK,"tankno"),
    assert(problem_tank(FOSERVTK),problem).

problem_type(,_0).

problem_type(void,1):- get_problem_tank,retrieve_contains,
    add_problem(["oil in void"]),
    contains_fuel,fuel(_),
    ask_ques_read_ans(LOSINGFUEL,"losing fuel"),!,
    losingfuel(LOSINGFUEL),solution.

problem_type(void,1):- !,
    ask_ques_read_ans(FOURTH,"fourth deck"),!,
    fourthdeck(FOURTH),solution.

problem_type(void,2):-
    get_problem_tank,!,add_problem(["unable to pump"]),
    !,find_string(String_num),retrieve_string(String_num),!,
    ask_ques_read_ans(LOSING_FUEL_WATER,"void losing fuel
    water"),!, losing_fuel_water(LOSING_FUEL_WATER),
solution.

problem_type(void,3).

problem_type(void,4):-get_problem_tank,!,
    find_string(String_num),retrieve_string(String_num),
    inboard_voids,
    add_problem(["pumps but refills"]),!,
    retrieve_adjacent(above),!,
    ask_ques_read_ans(SEA_VALVE,"sea valve
    leak"),sea_valve_leak(SEA_VALVE), solution.

problem_type(void,5).

problem_type(void,6).

problem_type("fuel oil service",1):- !,

```

```

    get_problem_tank,!,add_problem(["water in
    fuel"]),!,

find_string(String_NUM),retrieve_string(String_NUM),
    inboard_voids,outboard_voids,!,
    retrieve_adjacent(above),!,
    ask_ques_read_ans(FILLED,"fuel filled
    from"),!,asserta(filled_from(FILLED),
    problem),!,paste_test("Y"),!,
    solution.

problem_type("fuel oil service",2):- !,
    ask_ques_read_ans(_,"overflowing"),!,
    retrieve_contains,solution,!.
problem_type("fuel oil service",3).

problem_type("fuel oil service",4):- get_problem_tank,
    find_string(String_NUM),retrieve_string(String_NUM),
    ask_ques_read_ans(OIL,"voids have oil"),
    voids_oil(OIL),!,solution.*/

problem_type("fuel oil service",4):- !,
    ask_ques_read_ans(FOURTH,"fourth deck"),!,
    fourthdeck(FOURTH),solution.

problem_type("fuel oil service",5).

problem_type("fuel oil storage",1):- !,
    get_problem_tank,!,add_problem(["water in
    fuel"]),!,

find_string(String_NUM),retrieve_string(String_NUM),
    inboard_voids,outboard_voids,!,
    retrieve_adjacent(above),!,
    from"),assert_filled(FILLED),
    solution.

problem_type(contaminated,1).

problem_type("JP-5",1).

assert_filled(FILLED):- str_len(FILLED,LEN),LEN <> 0,
    asserta(filled_from(FILLED),problem),!,

```



```

fot_paste_test("Y"),!.
assert_filled(_):- !,fot_paste_test("N").

contains_fuel:-contains(_,SYSTEM),
frontchar(SYSTEM,_,REST),
frontstr(3,REST,SYS,_),
enter_fuel_fact(SYS,REST).

contains_fuel.

ask_ques_read_ans(ANSWER,QUES_NO):- !,
question(QUES_NO),
!readln(INPUT),
convert_case(INPUT,ANSWER),clearwindow.

convert_case(INPUT,ANSWER):- upper_lower(ANSWER,INPUT).
convert_case(INPUT,ANSWER):- ANSWER = INPUT.

print(TANK_TYPE,WINDOW):-retrieve_tank(TANK_TYPE,WINDOW),
nl.
print(TANK_TYPE,_):- error(TANK_TYPE).

retrieve_tank("adjacent fuel",WINDOW):-
adjacent_fuel_tanks(TANKS),
print_tanks(WINDOW,TANKS),fail.

retrieve_tank("adjacent fuel",_):- adjacent_fuel_tanks(_).

retrieve_tank("fuel tanks",WINDOW):-
fuel(TANKS),print_tanks(WINDOW,TANKS),fail.

retrieve_tank("fuel tanks",_):- fuel(_).

retrieve_tank("adjacent tanks",WINDOW):-
adjacent_tanks(TANKS),
print_tanks(WINDOW,TANKS),fail.

retrieve_tank("adjacent tanks",_):- adjacent_tanks(_).

retrieve_tank("string fuel tanks",WINDOW):-
string_fuel_tanks(TANKS),
print_tanks(WINDOW,TANKS),fail.

```

```

retrieve_tank("string fuel tanks",_-):-
    string_fuel_tanks(_).

retrieve_tank("voids in string",WINDOW):-
    string_void_tanks(TANKS),
    print_tanks(WINDOW,TANKS),fail.

retrieve_tank("voids in string",_-):- string_void_tanks(_).

retrieve_tank("string voids in tank",WINDOW):-
    string_void_in_tank(TANKS),
    print_tanks(WINDOW,TANKS),fail.

retrieve_tank("string voids in tank",_-):-
    string_void_in_tank(_).

retrieve_tank("inboard voids",WINDOW):-inbd_void(TANK),
    print_tanks(WINDOW,TANK),fail.

retrieve_tank("inboard voids",_-):- inbd_void(_).

retrieve_tank("outboard voids",WINDOW):- otbd_void(TANK),
    print_tanks(WINDOW,TANK),fail.

retrieve_tank("outboard voids",_-):- otbd_void(_).

retrieve_tank(,_):- fail.
    print_tanks(WINDOW,TANKS):-!,
    position_in_window(WINDOW),write(TANKS).

error("adjacent fuel"):- write("AT ERROR"),readchar(_).

error("fuel tank"):- write("AT ERROR"),readchar(_).

error("adjacent tanks"):- write("AT ERROR"),readchar(_).

error("string fuel tanks"):- write("AT
    ERROR"),readchar(_).

error("voids in string"):- write("AT ERROR"),readchar(_).

error("string voids in tank"):- write("AT

```

ERROR"),readchar(_).

error("inboard voids"):- indeterminate,solution,!,fail.

error("outboard voids"):- write("AT ERROR"),readchar(_).
find_string(String_Num):-problem_tank(TANK),
fronttoken(TANK,_,REST),
fronttoken(REST,_,STR1),
fronttoken(STR1,String_No,STR2),
fronttoken(STR2,_,STR3),
fronttoken(STR3,PORT_STBD,_),
port_stbd(PORT_STBD,SIDE),
concat(String_No,SIDE,String_Num).

inboard_voids:- problem_tank(TANK),
parse_tank_port_stbd(TANK, PROB_TANK),
str_int(PROB_TANK,PROB_PORT_STBD),
find_inbd_voids(PROB_PORT_STBD).

outboard_voids:- problem_tank(TANK),
parse_tank_port_stbd(TANK,PROB_TANK),
str_int(PROB_TANK,PROB_PORT_STBD),
find_otbd_voids(PROB_PORT_STBD).
parse_tank_port_stbd(TANK,PORT_STBD):-
fronttoken(TANK,_,REST),
fronttoken(REST,_,STR1), fronttoken(STR1,_,STR2),
fronttoken(STR2,_,STR3),
fronttoken(STR3,PORT_STBD,_).

find_inbd_voids(PROB_PORT_STBD):- string_void_tanks(_),
assert_inboards(PROB_PORT_STBD).
find_inbd_voids(_):- write("error").

find_otbd_voids(PROB_PORT_STBD):- string_void_tanks(_),
assert_outboards(PROB_PORT_STBD).
find_otbd_voids(_):- error("voids in string").

assert_inboards(PROB_PORT_STBD):- string_void_tanks(TANK),
parse_tank_port_stbd(TANK,TANK_PORT_STBD),
str_int(TANK_PORT_STBD,PORT_STBD),
PORT_STBD < PROB_PORT_STBD,
assert_inboard_voids(TANK),fail.
assert_inboards(_).

```
assert_inboard_voids(TANK):-
    assertz(inbd_void(TANK),problem).
```

```
assert_outboards(PROB_PORT_STBD):-
    string_void_tanks(TANK),
    parse_tank_port_stbd(TANK,TANK_PORT_STBD),
    str_int(TANK_PORT_STBD,PORT_STBD),
    PORT_STBD > PROB_PORT_STBD,
    assert_outboard_voids(TANK),fail.
assert_outboards(_).
```

```
assert_outboard_voids(TANK):-
    assertz(otbd_void(TANK),problem).
```

```
port_stbd(PORT_STBD,SIDE):- str_int(PORT_STBD,INT),ANS =
    INT mod 2,
    !,side(ANS,SIDE).
side(0,SIDE):- SIDE = "P".
side(_,SIDE):- SIDE = "S".
```

```
indeterminate:- retract_facts,assert_empty_fact,!,
    add_problem(["indeterminate data"]).
```

```
*****
*
*          PIPEGDOM.PRO          *
*          GLOBAL DOMAINS       *
*
*****/
```

GLOBAL DOMAINS

LIST = SYMBOL*

file = datafile

QUES_VAR = SYMBOL

WNO,SCRATTR,FRAMATTR,ROW,COL,LEN = INTEGER

KEY = cr ; esc ; break ; tab ; btab ; del ;

bdel ; ins ; end ; home ; function(INTEGER) ; up ; down ;

left ; right ;

ctrlleft; ctrlright; ctrlend; ctrlhome; pgup; pgdn;

chr(CHAR) ; otherspec

```

*****
*
*          GLOBDEF.PRO          *
*        GLOBAL DECLARATIONS   *
*
*****/

```

GLOBAL PREDICATES

```

nondeterm question(SYMBOL) - (i)
nondeterm write_solution(LIST) -(i)
nondeterm menu(WNO,SCRATTR,FRAMATTR,
  ROW,COL,STRING,LIST,INTEGER)-(i,i,i,i,i,i,o)
nondeterm write_screen(LIST)- (i)
nondeterm repeat
solution
position_in_window(SYMBOL)-(i)
nondeterm retrieve_contains
nondeterm retrieve_adjacent(SYMBOL)-(i)
nondeterm retrieve_string(SYMBOL)-(i)
nondeterm add_problem(LIST)-(i)
append(LIST,LIST,LIST)-(i,i,o)
nondeterm error(SYMBOL)-(i)
retract_facts
nondeterm print(SYMBOL,SYMBOL)-(i,i)
enter_fuel_fact(SYMBOL,SYMBOL)-(i,i)
compare_string_void_to_contains
nondeterm assert_fact(SYMBOL,SYMBOL)-(i,i)
assert_empty_fact
nondeterm ask_ques_read_ans(SYMBOL,SYMBOL)-(o,i)

```

```

*****
*
*          ANAGDEF.PRO          *
*        GLOBAL DECLARATIONS   *
*
*****/

```

GLOBAL PREDICATES

```

nondeterm indeterminate
nondeterm losingfuel(SYMBOL)-(i)
nondeterm paste_test(SYMBOL)-(i)

```

```

nondeterm fot_paste_test(SYMBOL)-(i)
nondeterm fourthdeck(SYMBOL)-(i)
nondeterm losing_fuel_water(SYMBOL)-(i)
nondeterm voids_oil(SYMBOL)-(i)
nondeterm sea_valve_leak(SYMBOL)-(i)
nondeterm fot_paste(SYMBOL)-(i)
nondeterm assert_filled(SYMBOL)-(i)
nondeterm fos_pump_run_refuel(SYMBOL)-(i)
nondeterm fot_pump_run_refuel(SYMBOL)-(i)
nondeterm mmr_strip_pump(SYMBOL)-(i)
nondeterm clear_sounding_tube(SYMBOL)-(i)
nondeterm void_ovrflow(SYMBOL)-(i)

```

```

*****
*
*          PIPEGDBASE.PRO
*
*****/

```

GLOBAL DATABASE - problem

```

contains(SYMBOL,SYMBOL)
problem_tank(SYMBOL)
user_data
first_char(CHAR)
adjacent_tanks(SYMBOL)
adjacent_fuel_tanks(SYMBOL)
string_tanks(SYMBOL)
string_fuel_tanks(SYMBOL)
string_void_tanks(SYMBOL)
string_void_in_tank(SYMBOL)
which_tank_losing_fuel(SYMBOL)
above_tank(SYMBOL)
filled_from(SYMBOL)
otbd_void(SYMBOL)
inbd_void(SYMBOL)
nondeterm fuel(SYMBOL)
count(INTEGER)
problem(LIST)

```

```

/*****
*
*      DATABASE.PRO
*
*****/

```

project "PIPES"

```

include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"

```

PREDICATES

```

search_contains(SYMBOL,INTEGER)
search_adjacent(SYMBOL,INTEGER)
search_string(SYMBOL)
compare_contains(SYMBOL,SYMBOL,SYMBOL)
compare_adjacent(SYMBOL,SYMBOL,SYMBOL,INTEGER)
compare_string(SYMBOL,SYMBOL,SYMBOL)
compare_string_to_problem_tank
moredata(file)
find_space(SYMBOL,INTEGER)
split_string(INTEGER,SYMBOL,SYMBOL,SYMBOL)
is_space(INTEGER,INTEGER,SYMBOL)
check_adjacent(SYMBOL)
check_string(SYMBOL)
which_adjacent(SYMBOL)
which_tanks_in_string
find_adj_tanks(SYMBOL)
find_above_tank
find_string_tanks(SYMBOL)
check_empty_list(LIST,LIST)
assert_string_void_in_tank(SYMBOL)

```

CLAUSES

```

retrieve_adjacent(WHICH):-
    openread(datafile,"adjacent.txt"), readdevice(datafile),

    problem_tank(COMP_NUM), str_len(COMP_NUM,LENGTH),
    search_adjacent(COMP_NUM,LENGTH), closefile(datafile),
    readdevice(keyboard),

```

```
which_adjacent(WHICH).
retrieve_adjacent(_):-
closefile(datafile).
```

```
which_adjacent(WHICH):- WHICH =
    "all",adjacent_tanks(TANKS),
    find_adj_tanks(TANKS).
which_adjacent(WHICH):- WHICH = "above",find_above_tank.
```

```
find_above_tank:-
    adjacent_tanks(TANK),str_len(TANK,LEN),LENGTH = LEN -
    13,frontstr(LENGTH,TANK,_,LAST), find_space(LAST,0),
    count(COUNT),frontstr(COUNT,LAST,_,ABOVE),!,
    assertz(above_tank(ABOVE),problem).
find_above_tank:- !.
```

```
search_adjacent(COMP_NUM,LENGTH):-
    readln(PIPESYS),!,moredata(datafile),
    split_string(LENGTH,PIPESYS,COMPNO,SYSTEM),
    compare_adjacent(COMP_NUM,COMPNO,SYSTEM,LENGTH).
search_adjacent(_,_):- !.
```

```
compare_adjacent(COMP_NUM,COMPNO,REST,_):-
    COMP_NUM = COMPNO,!,assertz(adjacent_tanks(REST),
    problem).
compare_adjacent(COMP_NUM,_,_,LENGTH):-
    search_adjacent(COMP_NUM,LENGTH).
```

```
find_adj_tanks(TANKS):-
    frontchar(TANKS,_,REST),find_space(REST,0),
    count(LENGTH),
    frontstr(LENGTH,REST,COMPNO,STR1),
    !,check_adjacent(COMPNO),find_adj_tanks(STR1).
find_adj_tanks(_):- !.
```

```
check_adjacent(COMPNO):- str_len(COMPNO,LEN),
    LAST = LEN - 1,
    frontstr(LAST,COMPNO,_,USAGE),USAGE = "F",
    assertz(adjacent_fuel_tanks(COMPNO),problem).
```

```
check_adjacent(_):- !.
```


find_space(ADJ_TANK,COUNT):- frontchar(ADJ_TANK,FRT,REST),
char_int(FRT,VAL),is_space(VAL,COUNT,REST).

is_space(VAL,COUNT,REST):- VAL <> 32,INCCOUNT =
COUNT + 1,!,
find_space(REST,INCCOUNT),!.
is_space(_,COUNT,_):- asserta(count(COUNT),problem).

retrieve_string(String_Num):-
openread(datafile,"compartment.txt"),
readdevice(datafile),
search_string(String_Num), closefile(datafile),
readdevice(keyboard),
which_tanks_in_string.
retrieve_string(_):-
closefile(datafile),readdevice(keyboard),readchar(_).

search_string(String_Num):-
readln(DATA),moredata(datafile),
frontstr(4,DATA,STR_NO,REST),
compare_string(String_Num,STR_NO,REST),
search_string(String_Num).
search_string(_):- !.

compare_string(String_Num,STRNO,REST):-
String_Num = STRNO,assertz(string_tanks(REST),problem).
compare_string(_,_,_):- !.

which_tanks_in_string:- string_tanks(TANKS),
find_string_tanks(TANKS),fail.
which_tanks_in_string:- !.

find_string_tanks(TANKS):- !,frontchar(TANKS,_,REST),
check_string(REST).

check_string(TANKS):- str_len(TANKS,LEN), LAST = LEN - 1,
frontstr(LAST,TANKS,_,USAGE),USAGE = "F",
assertz(string_fuel_tanks(TANKS),problem).

check_string(TANKS):- str_len(TANKS,LEN), LAST = LEN - 1,

```

frontstr(LAST,TANKS,_,USAGE),USAGE = "V",
assertz(string_void_tanks(TANKS),problem).
check_string(_):- !.

retrieve_contains:-
    openread(datafile,"contains.txt"),
    readdevice(datafile), problem_tank(COMP_NUM),
    str_len(COMP_NUM,LENGTH),
    search_contains(COMP_NUM,LENGTH), closefile(datafile),
    readdevice(keyboard).

retrieve_contains:-
    closefile(datafile),readdevice(keyboard).
search_contains(COMP_NUM,LENGTH):-
    readln(PIPESYS),moredata(datafile),
    split_string(LENGTH,PIPESYS,COMPNO,SYSTEM),
    compare_contains(COMP_NUM,COMPNO,SYSTEM),
    search_contains(COMP_NUM,LENGTH).
search_contains(_,_):- !.

split_string(LENGTH,PIPESYS,COMPNO,SYSTEM):-!,
    frontstr(LENGTH,PIPESYS,COMPNO,SYSTEM).

compare_contains(COMP_NUM,COMPNO,SYSTEM):-
    COMP_NUM = COMPNO,
    assertz(contains(COMPNO,SYSTEM),problem).

moredata(FILE):- not(eof(FILE)).

add_problem(PROBLEM):- problem(LIST),
    check_empty_list(LIST,PROBLEM).

check_empty_list(["_"],PROBLEM):-
    !,asserta(problem(PROBLEM),problem).
check_empty_list(LIST,PROBLEM):-
    !,append(LIST,PROBLEM,LIST2),
    asserta(problem(LIST2),problem).

compare_string_to_problem_tank:- string_void_tanks(_),
    compare_string_void_to_contains.
compare_string_to_problem_tank:-error("voids in string").

compare_string_void_to_contains:- string_void_tanks(VOID),

```

```

contains(_,PIPESYS), frontchar(PIPESYS,_,REST),
fronttoken(REST,_,STR1),
frontchar(STR1,_,PIPE),VOID = PIPE,
assert_string_void_in_tank(VOID),fail.
compare_string_void_to_contains.

```

```

assert_string_void_in_tank(SYMBOL):- !,
    assertz(string_void_in_tank(SYMBOL),problem).

```

```

/*****
*
*          SCREEN.PRO
*
*****/

```

```

project "PIPES"

```

```

include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"

```

PREDICATES

```

readkey(KEY)
readkey1(KEY,CHAR,INTEGER)
readkey2(KEY,INTEGER)
answer(CHAR)

```

CLAUSES

```

readkey(KEY):-readchar(T),answer(T),char_int(T,VAL),
readkey1(KEY,T,VAL).

```

```

readkey1(KEY,_,0):-!,readchar(T),char_int(T,VAL),
readkey2(KEY,VAL).
readkey1(cr,_,13):-!.
readkey1(esc,_,27):-!.
readkey1(chr(T),T,_) .

```

```

readkey2(up,72):-!.
readkey2(down,80):-!.
readkey2(function(N),VAL):- VAL>58,VAL<70,N=VAL-58,!.
readkey2(otherspec,_) .

```

```

answer(T):- user_data,asserta(first_char(T)),write(T).
answer(_):- !.

```

PREDICATES

```

maxlen(LIST,INTEGER,INTEGER)
listlen(LIST,INTEGER)
writelist(INTEGER,INTEGER,LIST)
index(LIST,INTEGER,SYMBOL)
scroll_screen(LIST)
check_end_of_question(LIST)
check_question_length(LIST,INTEGER)

```

```

/*****
*
*          MENUS
*
*
*****/

```

PREDICATES

```

menu1(ROW,SCRATTR,LIST,ROW,INTEGER,INTEGER)
menu2(ROW,SCRATTR,LIST,ROW,INTEGER,INTEGER,KEY)
request(INTEGER,INTEGER,INTEGER,SYMBOL)
user_key(INTEGER,INTEGER,SYMBOL)
user_request(INTEGER,INTEGER,SYMBOL)
user_input(KEY,INTEGER,INTEGER,INTEGER)

```

CLAUSES

```

menu(WN,SN,FN,LI,KOL.TXT,LIST,CHOICE):-
    shiftwindow(21),
    maxlen(LIST,0,ANTKOL),
    listlen(LIST,LEN),ANTLI=LEN,LEN>0,
    HH1=ANTLI+2,HH2=ANTKOL+2,

    makewindow(WN,SN,FN,TXT,LI,KOL,HH1,HH2),
    HH3=ANTKOL,
    writelist(0,HH3,LIST),cursor(0,0),
    menu1(0,SN,LIST,ANTLI,ANTKOL,CH),
    CHOICE=1+CH,
    removewindow,
    shiftwindow(22),
    shiftwindow(2).

```

```

menu1(LI,SN,LIST,MAXLI,ANTKOL,CHOICE):-
    field_attr(LI,0,ANTKOL,112),
    cursor(LI,0),
    readkey(KEY),
    menu2(LI,SN,LIST,MAXLI,ANTKOL,CHOICE,KEY).

menu2(_,_,-1,esc):-!.
menu2(LI,_,-1,CH,function(10)):-!,CH=LI.
menu2(LI,_,-1,CH,cr):-!,CH=LI.
menu2(LI,SN,LIST,MAXLI,ANTKOL,CHOICE,up):-
    LI>0,!,
    field_attr(LI,0,ANTKOL,SN),
    LI1=LI-1,
    menu1(LI1,SN,LIST,MAXLI,ANTKOL,CHOICE).

menu2(LI,SN,LIST,MAXLI,ANTKOL,CHOICE,down):-
    LI<MAXLI-1,!,
    field_attr(LI,0,ANTKOL,SN),
    LI1=LI+1,
    menu1(LI1,SN,LIST,MAXLI,ANTKOL,CHOICE).

menu2(LI,SN,LIST,MAXLI,ANTKOL,CHOICE,_):-
    menu1(LI,SN,LIST,MAXLI,ANTKOL,CHOICE).

user_key(ROW,COL,ANSWER):-
    asserta(user_data),user_request(ROW,COL,ANSWER),
    retract(user_data).
user_request(ROW,COL,ANSWER):- readkey(KEY),
    user_input(KEY,REQUEST,ROW,COL),
    !,request(REQUEST,ROW,COL,ANSWER).

user_input(esc,-1,_):-!.
user_input(cr,1,_):-!.
user_input(down,1,ROW,COL):- cursor(R,C),R <> ROW,
    NROW = R + 1,

scr_char(R,C,CHAR),cursor(NROW,COL),scr_char(R,C,CHAR).
user_input(down,1,ROW,COL):-!,cursor(ROW,COL).
user_input(up,1,_COL):-cursor(R,C),R <> 0,NROW = R
- 1,scr_char(R,C,CHAR),
    cursor(NROW,COL),scr_char(R,C,CHAR).
user_input(up,1,ROW,COL):-!,cursor(ROW,COL).
user_input(_,0,_):-!.

```

```

request(1,ROW,COL,ANSWER):- user_request(ROW,COL,ANSWER).
request(0,_,_,ANSWER):- readln(ANS),first_char(T),
frontchar(ANSWER,T,ANS),
write(ANSWER),retract(first_char(T)).
request(-1,_,_,_):- fail.

```

CLAUSES

```

index([X|_],1,X):- !.
index([_|L],N,X):- N>1,N1=N-1,index(L,N1,X).

```

```

append([],L,L).
append([A|At],B,[A|C]):-append(At,B,C).

```

```

maxlen([H|T],MAX,MAX1):-
    str_len(H,LEN),
    LEN>MAX,!,
    maxlen(T,LEN,MAX1).
maxlen([_|T],MAX,MAX1):-maxlen(T,MAX,MAX1).
maxlen([],LEN,LEN).

```

```

listlen([],0).
listlen([_|T],N):-
    listlen(T,X),
    N=X+1.

```

```

writelist(_,_,[]).
writelist(LI,ANTKOL,[H|T]):-field_str(LI,0,ANTKOL,H),
    LI1=LI+1,writelist(LI1,ANTKOL,T).

```

```

write_screen([]).
write_screen([H|T]):-
    scroll_screen([H|T]),
    write_screen(T).

```

```

scroll_screen([H|T]):-
    cursor(_,COL),
    check_question_length([H],COL),
    write(H," "),check_end_of_question(T).

```

```

scroll_screen([H|T]):-
    scroll(1,0),write(H),
    check_end_of_question(T).

```

```

check_question_length([Hl_],LENGTH):-
    str_len(H,LEN),MAXLEN = LEN + LENGTH,
    MAXLEN < 65.

```

```

check_question_length(_,_):- nl.

```

```

check_end_of_question([]).
check_end_of_question([Hl_]):- not(isname(H)).
check_end_of_question(_):- nl.

```

```

position_in_window("problem analysis"):-
    shiftwindow(4),nl.
position_in_window(action):-
    shiftwindow(7),clearwindow,nl.
position_in_window(solution):- shiftwindow(12),nl.

```

```

repeat. repeat:-repeat.

```

```

/*****
*
*          FACT.PRO
*
*****/

```

```

project "PIPES"

```

```

include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"

```

```

CLAUSES

```

```

enter_fuel_fact(SYS,COMPNO2):-
    assert_fact(SYS,COMPNO2),fail.

```

```

assert_fact(SYS,COMPNO2):-
    SYS="FOT",assertz(fuel(COMPNO2),problem).

```

```

assert_fact(SYS,COMPNO2):-
    SYS="FOS",assertz(fuel(COMPNO2),problem).

```

assert_empty_fact:- asserta(problem(["_"]),problem).

retract_facts:- retractall(_,problem).

```

/*****
*
*
*   PIPEGRAPH.PRO
*
*****/
```

include "GrapDecl.PRO"

CONSTANTS

intlist = BGI_ilst

Domains

PointList = intlist*

Constants

PaletteList = intlist

```

/*****
*
*   Local data base
*
*****/
```

Database - graphics

Determ driver(Integer,Integer,String)

Determ maxcolors(Integer)

Determ maxX(Integer)

Determ maxY(Integer)

Determ graphCoord(Integer,Integer)

```

/*****
*
*           Return Type of Tank
*
*****/
```

PREDICATES

TankType(String,String)

CLAUSES

TankType("8-119-9-V","8-119-9-V DC VOID").
TankType("8-119-7-V","8-119-7-V DC VOID").
TankType("8-119-11-F","8-119-11-F FOS").
TankType("8-119-7-V","8-119-7-V DC VOID").

```
/******  
*  
*           Return pipes in compartment           *  
*  
*****
```

PREDICATES

Pipes(SYMBOL,INTEGER,INTEGER,INTEGER,INTEGER,INTEGER,SYMBOL)

CLAUSES

Pipes("8-119-7-V",1,-1,0,-1,0,"RED").
Pipes("8-119-7-V",4,-3,0,-3,0,"BLUE").
Pipes("8-119-7-V",1,-6,0,-6,0,"RED").
Pipes("8-119-7-V",0,-8,0,-8,0,"ORANGE").
Pipes("8-119-7-V",2,-15,0,-15,0,"GREEN").
Pipes("8-119-7-V",0,6,0,6,0,"ORANGE").
Pipes("8-119-7-V",2,11,0,11,-28,"GREEN").
Pipes("8-119-7-V",2,13,0,13,0,"GREEN").
Pipes("8-119-7-V",2,15,0,15,0,"GREEN").
Pipes("8-119-9-V",1,-1,0,-1,0,"RED").
Pipes("8-119-9-V",4,-3,0,-3,0,"BLUE").
Pipes("8-119-9-V",1,-6,0,-6,0,"RED").
Pipes("8-119-9-V",0,-8,0,-8,0,"ORANGE").
Pipes("8-119-9-V",2,-15,0,-15,0,"GREEN").
Pipes("8-119-9-V",0,6,0,6,0,"ORANGE").
Pipes("8-119-9-V",2,13,0,13,-23,"GREEN").
Pipes("8-119-9-V",2,15,0,15,0,"GREEN").
Pipes("8-119-11-F",1,-1,0,-1,0,"RED").
Pipes("8-119-11-F",4,-3,0,-3,-10,"BLUE").
Pipes("8-119-11-F",1,-6,0,-6,-25,"RED").
Pipes("8-119-11-F",0,-8,0,-8,-28,"ORANGE").
Pipes("8-119-11-F",2,-15,0,-15,0,"GREEN").
Pipes("8-119-11-F",0,6,0,6,-30,"ORANGE").

```
Pipes("8-119-11-F",2,15,0,15,0,"GREEN").
Pipes("8-119-15-V",2,15,0,15,-26,"GREEN").
```

```

/*****
*
*           Return pipename
*
*****/
```

PREDICATES

```
Pipename(SYMBOL,SYMBOL)
```

CLAUSES

```
Pipename("RED","Fuel Oil Transfer").
Pipename("BLUE","Fuel Oil Stripping").
Pipename("GREEN","Ballasting and Main Drain").
Pipename("ORANGE","Fuel Oil Service & Recirc").
Pipename("MAGENTA","OverFlow").
```

```

/*****
*
*           Return name of driver
*
*****/
```

PREDICATES

```
GetDriverName2(Integer,String)
```

CLAUSES

```
GetDriverName2(0,"Detect").
GetDriverName2(1,"CGA").
GetDriverName2(2,"MCGA").
GetDriverName2(3,"EGA").
GetDriverName2(4,"EGA64").
GetDriverName2(5,"EGAMono").
GetDriverName2(6,"Reserved").
GetDriverName2(7,"HercMono").
GetDriverName2(8,"ATT400").
GetDriverName2(9,"VGA").
GetDriverName2(10,"PC3270").
```

PREDICATES

GetMode(Integer,Integer,String)

CLAUSES

```
GetMode(cga,cgaHi,"CGAHi"):-!.
GetMode(cga,GraphMode,S):-!,format(S,"CGA%",GraphMode).
GetMode(mcga,mcgaMed,"MCGAMed"):-!.
GetMode(mcga,mcgaHi,"MCGAHi"):-!.
GetMode(mcga,GraphMode,S):-!,format(S,"MCGA%",GraphMode).
GetMode(ega,egaLo,"EGALo"):-!.
GetMode(ega,egaHi,"EGAHi"):-!.
GetMode(ega64,ega64Lo,"EGA64Lo"):-!.
GetMode(ega64,ega64Hi,"EGA64Hi"):-!.
GetMode(hercMono,_, "HercMonoHi"):-!.
GetMode(egaMono,_, "EGAMonoHi"):-!.
GetMode(pc3270,_, "PC3270Hi"):-!.
GetMode(att400,att400Med,"ATT400Med"):-!.
GetMode(att400,att400Hi,"ATT400Hi"):-!.
GetMode(att400,GraphMode,S):-
!,format(S,"ATT400%",GraphMode).
GetMode(vga,vgaLo,"VGALo"):-!.
GetMode(vga,vgaMed,"VGAMedo"):-!.
GetMode(vga,vgaHi,"VGAHi"):-!.
GetMode(_,_, "UnKnown"):-!.
```

```
/*
*
*           Return name of font
*
*/
```

PREDICATES

GetFontName(Integer,String)

CLAUSES

```
GetFontName(1,"TriplexFont").
GetFontName(2,"SmallFont").
GetFontName(3,"SansSerifFont").
GetFontName(4,"GothicFont").
```

```

/*****
*
*      Implementation of the C loop: for(I=Cur, i<Max, I++)
*
*****/

```

PREDICATES

nondeterm for(Integer,Integer,Integer)

CLAUSES

for(Cur,_,Cur).
for(Cur,Max,I):- Cur2=Cur+1, Cur2<Max, for(Cur2,Max,I).

```

/*****
*
*      Mode switching
*
*****/

```

PREDICATES

ToGraphic
ToText
KeepColor(integer,integer,integer)

CLAUSES

ToGrapHic:-
/* Detect graphic equipment */
DetectGraph(G_Driver, G_Mode1),
KeepColor(G_Driver,G_Mode1,G_Mode),
GetDriverName2(G_Driver,G_Name),
assert(driver(G_Driver,G_Mode,G_Name)),
envsymbol("BGIDIR",SetValue),
InitGraph(G_Driver,G_Mode, _, _,SetValue),!.

ToText:-
closegraph().

KeepColor(1,_,0).
KeepColor(_,Mode,Mode).

```

/*****
*
*           Display a status line at the bottom of the screen
*
*****/

```

PREDICATES

StatusLine(String)

CLAUSES

StatusLine(Msg):-

```

    maxX(MaxX), maxY(MaxY), SetViewPort(0,0,MaxX,MaxY, 1),
    maxColors(MaxColors), MaxCol2=MaxColors,
    SetColor(MaxCol2),
    SetBkColor(0),
    SetTextStyle(default_Font, horiz_Dir, 1),
    SetTextJustify(center_Text, top_Text),
    SetLineStyle(solid_Line,0,norm_Width),
    SetFillStyle(empty_Fill,0),
    TextHeight("H",Height), MaxYH = MaxY-(Height+4),
    Bar(0, MaxYH,MaxX,MaxY),
    Rectangle(0,MaxYH,MaxX,MaxY),
    MaxX2 = MaxX div 2, MaxY2 = MaxY-(Height+2),
    OutTextXY(MaxX2,MaxY2, Msg),
    Height5 = Height+5, MaxX1=MaxX-1, MaxY5 =
    MaxY-(Height+5),
    SetViewPort(1,Height5,MaxX1,MaxY5,1).

```

```

/*****
*
*           Pause until the user enters a keystroke
*
*****/

```

PREDICATES

Pause

CLAUSES

Pause:-

```

    readChar(_).

```

```

/*****
*
*           Draw a solid line around the current viewport
*
*****/

```

PREDICATES

DrawBorder

CLAUSES

DrawBorder:-

```

    maxColors(MaxColors), MaxCol2 = MaxColors,
SetColor(MaxCol2),
    SetBkColor(0),
    SetLineStyle(solid_Line,0,thick_Width),
    GetViewSettings(Left,Top,Right,Bottom,_),
    RL=Right-Left, BT=Bottom-Top,
    Rectangle(0,0,RL,BT).

```

```

/*****
*
*           Establish the main window and set a viewport
*
*****/

```

PREDICATES

FullScreen(String)

CLAUSES

FullScreen(Header):-

```

    ClearDevice,
    maxColors(MaxColors),
    SetColor(MaxColors),           % Set current color to
white
    SetBkColor(0),                 % Set background to black
    TextHeight("H",Height),       % Get basic text height
    Height5=Height+5, Height4=Height+4,
    maxX(MaxX), MaxX1=MaxX-1,MaxX2=MaxX div 2,
    maxY(MaxY), MaxY4=MaxY-(Height4),MaxY5=MaxY-(Height5),
    SetViewPort(0,0,MaxX,MaxY,1), % Open port to full
screen
    SetTextStyle(small_font,horiz_dir,0),
    SetTextJustify(center_text, top_text),

```

```

OutTextXY(MaxX2,2,Header),
SetViewport( 0, Height4, MaxX, MaxY4, 1),
DrawBorder,
SetViewport(1, Height5, MaxX1,MaxY5, 1).

```

```

/*****

```

Initialize video and Global flags

```

****

```

PREDICATES

Initialize

CLAUSES

Initialize:-

```

    retractall(_, graphics),
    ToGraphic,
    GetMaxColor(MaxColors), assert(maxcolors(MaxColors)),
    GetMaxX(MaxX),         assert(maxX(MaxX)),
    GetMaxY(MaxY),         assert(maxY(MaxY)).

```

```

/*****

```

Display PIPES system screen

```

****

```

PREDICATES

PipesScreen

CLAUSES

PipesScreen:-

FullScreen("Piping Improvement And Planning Expert
System"),

```

    GetViewSettings(Left,Top,Right,Bottom,_),
    SetTextStyle(gothic_FONT,hORIZ_Dir,5),
    SetTextJustify(center_Text,center_Text),
    maxColors(MaxColor), Color = 1+round(MaxColor / 2.5),
    SetColor(Color),
    H = Bottom - Top, W = Right - Left,

```

```

W2 = W div 2, H2 = H div 2,
OutTextXY(W2,H2,"P I P E S"),
StatusLine("Press any key to Continue").

```

```

/*****

```

Display a pattern of random dots on the screen

```

****

```

DOMAINS

```

Pixel    = p(Integer,Integer,Integer)
PixelList = Pixel*

```

PREDICATES

```

WriteDot
PutPixels(Pixellist,Integer,Integer,Integer,Integer)
OutPixels(Pixellist)
DelPixels(Pixellist)
Delay(Integer)

```

CLAUSES

```

WriteDot:-
    GetViewSettings(Left,Top,Right,Bottom,_),
    H = Bottom - Top,
    W = Right - Left,
    maxColors(MaxColors),
    PutPixels(Points,1200,H,W,Maxcolors),
    DelPixels(Points),
    for (0,2,I),
        OutPixels(Points),
        I<1,
        DelPixels(Points),
    fail.

```

```

WriteDot:- pause,ToText.

```

```

PutPixels([],0,_,_,):- !.

```

```

PutPixels([p(X,Y,Color)|Points],I,H,W,Maxcolors):-
    random(W,X),
    random(H,Y),

```



```

    random(MaxColors,Color),
    PutPixel(X,Y,Color),
    I2 = I - 1,!,
    PutPixels(Points,I2,H,W,Maxcolors).

OutPixels([p(X,Y,Color)|Points]):-!,
    PutPixel(X,Y,Color),
    OutPixels(Points).
OutPixels(_).

DelPixels([p(X,Y,_)|Points]):-
!,PutPixel(X,Y,black),DelPixels(Points).
DelPixels(_).

Delay(N):- N > 0,!,N1 = N-1, Delay(N1).
Delay(0).

/*****

    DISPLAY COVER
    *****/

PREDICATES

    PipeCover

CLAUSES

    PipeCover:- !,Initialize,      % Set system into
    graphic mode
        GraphDefaults,
        PipesScreen,
        WriteDot.

/*****

    DRAW PipeSystems
    *****/

PREDICATES

```

SingleTank(SYMBOL)
 PlotLine(SYMBOL)
 WriteName(SYMBOL,INTEGER,INTEGER)

CLAUSES

SingleTank(CompNo):-
 !,Initialize, % Set system into
 graphic mode
 GraphDefaults,TankType(CompNo,TYPE),
 FullScreen(TYPE),
 StatusLine("Press any Key to
 Continue"),
 setwritemode(0),

 setlinestyle(solid_LINE,0,norm_WIDTH),
 PlotLine(CompNo),
 pause,ToText.

PlotLine(CompNo):-

 pipes(CompNo,COLOR,XPOS,YPOS,XPOS1,YPOS1,PIPETYPE),
 maxColors(MaxColor),
 LineColor=
 MaxColor-COLOR,setcolor(LineColor),

 GetViewSettings(Left,Top,Right,Bottom,_),
 H=Bottom-Top,W=Right-Left,
 X = W div 2,
 XCORR = X div 16 ,XREAL = XCORR *
 XPOS,
 XPLOT = X + XREAL,
 XREAL1 = XCORR * XPOS1,
 XPLOT1 = X + XREAL1,
 YCORR = H div 32 ,YREAL = YCORR *
 YPOS,
 YPLOT = LEFT + YREAL,
 YREAL1 = YCORR * YPOS1,
 YPLOT1 = Bottom + YREAL1,

 line(XPLOT,YPLOT,XPLOT1,YPLOT1),APLOTY=YPLOT1-3,

```

        arc(XPLOT1,APLOTY,180,0,4),
        setttextstyle(small_Font,vert_DIR,0),
        SetTextJustify(top_TEXT,top_TEXT),
        TX = XPLOT -
5,TY=Top,WriteName(PIPETYPE,TX,TY),
    fail.
    PlotLine(_).

    WriteName(PIPETYPE,TX,TY):-
    !,Pipename(PIPETYPE,PIPENAME),
        outtextxy(TX,TY,PIPENAME).
    WriteName(_,_,_).
/*****
*
*      Oil in the Void Tank
*
*****/

project "PIPES"

include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "AnaGdef.PRO"

PREDICATES

    nondeterm cracks(SYMBOL)

CLAUSES

    losingfuel("Y"):- add_problem(["losing fuel"]).
    losingfuel("N"):- ask_ques_read_ans(CRACKS,"cracks in
        tank"),cracks(CRACKS).

    cracks("Y"):- !,add_problem(["cracks"]).
    cracks("N"):- ask_ques_read_ans(FOURTHDECK,"fourth deck"),
        !,fourthdeck(FOURTHDECK),!.

    fourthdeck("Y"):- !,add_problem(["4th deck"]),!.
    fourthdeck("N"):- !,indeterminate,!.

```

```

/*****
*
*          Unable to pump Void Tank
*
*****/

```

Project "PIPES"

```

include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"
include "AnaGdef.Pro"

```

PREDICATES

```

nondeterm pump_leaks(SYMBOL)
nondeterm valve_open(SYMBOL)
nondeterm debris(SYMBOL)
nondeterm void_pumps(SYMBOL)
nondeterm clean_valve(SYMBOL)
nondeterm manifold(SYMBOL)
nondeterm eductor(SYMBOL)
nondeterm submersible(SYMBOL)
nondeterm bilge(SYMBOL)
nondeterm bulkhd
nondeterm bulkhd_stops(SYMBOL)
nondeterm close_valve(SYMBOL)
nondeterm bulkhd_stops_close(SYMBOL)
nondeterm pump_cracks(SYMBOL)
nondeterm sounding(SYMBOL)
nondeterm string_leaks(SYMBOL)

```

CLAUSES

```

losing_fuel_water("Y"):- ask_ques_read_ans(WHICH,"which
    tank losing fuel"),
    asserta(which_tank_losing_fuel(WHICH)),
    ask_ques_read_ans(LEAKS,
        "unable pump losing fuel"),pump_leaks(LEAKS).
losing_fuel_water("N"):-
    ask_ques_read_ans(VALVE_OPEN,"check valve"),

```

```

valve_open(VALVE_OPEN).

pump_leaks("Y"):- add_problem(["leaks"]).
pump_leaks("N"):- losing_fuel_water("N").

valve_open("Y"):- ask_ques_read_ans(MANIFOLD,"manifold"),
    manifold(MANIFOLD).
valve_open("N"):- ask_ques_read_ans(VOID_PUMP,"void
    pump"),void_pumps(VOID_PUMP).

debris("Y"):- ask_ques_read_ans(CLEAN,"clean valve"),
    clean_valve(CLEAN).
debris("N"):- eductor("N").

void_pumps("Y"):- add_problem(["pumps"]).
void_pumps("N"):- valve_open("Y").

clean_valve("Y"):- add_problem(["clean valve"]).
clean_valve("N"):- eductor("N").

manifold("Y"):- ask_ques_read_ans(EDUCTOR,"eductor"),
    eductor(EDUCTOR).
manifold("N"):- ask_ques_read_ans(DEBRIS,"check
    debris"),debris(DEBRIS).

eductor("Y"):- add_problem(["eductor"]).
eductor("N"):-ask_ques_read_ans(BILGE,bilge),bilge(BILGE).

submersible("Y"):- add_problem(["submersible"]).
submersible("N"):- ask_ques_read_ans(CRACKS,"cracks in
    tank"),pump_cracks(CRACKS).

bilge("Y"):-bulkhd.
bilge("N"):- ask_ques_read_ans(CLOSE_VALVE,"close valve"),
    close_valve(CLOSE_VALVE).

bulkhd:- ask_ques_read_ans(BULKHD_STOPS,"bulkhd stops"),
    bulkhd_stops(BULKHD_STOPS).

bulkhd_stops("Y"):-
    ask_ques_read_ans(SUBMERSIBLE,"submersible"),
    submersible(SUBMERSIBLE).
bulkhd_stops("N"):- ask_ques_read_ans(STOPS_CLOSE,"bulkhd

```

```

stops close"),
bulkhd_stops_close(STOPS_CLOSE).

close_valve("Y"):- add_problem(["valves closed"]).
close_valve("N"):- bulkhd.

bulkhd_stops_close("Y"):- add_problem(["bilge"]).
bulkhd_stops_close("N"):- bulkhd_stops("Y").

pump_cracks("Y"):- add_problem(["cracks"]).
pump_cracks("N"):- ask_ques_read_ans(LEAKS,"leaky pipes"),
    string_leaks(LEAKS).

sounding("Y"):- add_problem(["sounding tube"]).
sounding("N"):- indeterminate.

string_leaks("Y"):- add_problem(["string leaks"]).
string_leaks("N"):- ask_ques_read_ans(SOUNDING,"sounding
    tube"),sounding(SOUNDING).

```

```

/*****
*
*      Void Tank Pumps but Refills with Water
*
*
*****/

```

project "PIPES"

```

include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "AnaGdef.PRO"

```

PREDICATES

```

cycle_sea_valve(SYMBOL)
void_backflow(SYMBOL)
manifold_pressure(SYMBOL)
void_cracks(SYMBOL)
overboard(SYMBOL)
void_debris(SYMBOL)
void_breaks(SYMBOL)
void_suction(SYMBOL)

```

open_ovbd(SYMBOL)
holes(SYMBOL)
close_suction(SYMBOL)
bulkhd_stops_open(SYMBOL)
close_bulkhd_stops(SYMBOL)
flooding_valve(SYMBOL)
close_flood_valve(SYMBOL)
inboard_void(SYMBOL)

CLAUSES

sea_valve_leak("Y"):- ask_ques_read_ans(CYCLE,"cycle sea valve"),cycle_sea_valve(CYCLE).
sea_valve_leak("N"):-ask_ques_read_ans(BACKFLOW,"void backflow"),void_backflow(BACKFLOW).

cycle_sea_valve("Y"):- add_problem(["cycle sea valve"]).
cycle_sea_valve("N"):- add_problem(["industrial repair"]).

void_backflow("Y"):- ask_ques_read_ans(MANIFOLD,"manifold pressure"),manifold_pressure(MANIFOLD).
void_backflow("N"):- ask_ques_read_ans(CRACKS,"cracks in tank"),void_cracks(CRACKS).

manifold_pressure("Y"):-ask_ques_read_ans(OVERBOARD,"overboard"),overboard(OVERBOARD).
manifold_pressure("N"):- ask_ques_read_ans(DEBRIS,"check debris"),void_debris(DEBRIS).

void_cracks("Y"):- add_problem(["void cracks"]).
void_cracks("N"):- ask_ques_read_ans(BREAKS,"sounding tube"),void_breaks(BREAKS).

overboard("Y"):- ask_ques_read_ans(SUCTION,"void suction"),void_suction(SUCTION).
overboard("N"):- ask_ques_read_ans(OPEN,"open ovbd"),open_ovbd(OPEN).

void_debris("Y"):- add_problem(["debris"]).
void_debris("N"):- ask_ques_read_ans(INBOARD,"inboard void"),inboard_void(INBOARD).

inboard_void("Y"):- add_problem(["inboard"]).

inboard_void("N"):- indeterminate.

void_breaks("Y"):- add_problem(["breaks"]).

void_breaks("N"):- ask_ques_read_ans(HOLES,"holes"),
holes(HOLES).

void_suction("Y"):- ask_ques_read_ans(CLOSE,"close
suction"),close_suction(CLOSE).

void_suction("N"):- ask_ques_read_ans(OPEN,"bulkhd
stops open"),bulkhd_stops_open(OPEN).

open_ovbd("Y"):- add_problem(["open ovbd"]).

open_ovbd("N"):- overboard("Y").

holes("Y"):- add_problem(["holes 4th deck"]).

holes("N"):- indeterminate.

close_suction("Y"):- void_suction("N").

close_suction("N"):- manifold_pressure("N").

bulkhd_stops_open("Y"):- ask_ques_read_ans(STOPS,"close
bulkhd stops"),close_bulkhd_stops(STOPS).

bulkhd_stops_open("N"):- manifold_pressure("N").

close_bulkhd_stops("Y"):- ask_ques_read_ans(FLOODING,
"flooding valve"),flooding_valve(FLOODING).

close_bulkhd_stops("N"):- manifold_pressure("N").

flooding_valve("Y"):- indeterminate.

flooding_valve("N"):- ask_ques_read_ans(FLOOD,
"close flood valve"),close_flood_valve(FLOOD).

close_flood_valve("Y"):- indeterminate.

close_flood_valve("N"):- manifold_pressure("N").

```
/******  
*                                                                 *  
*      Damage Control Void Will Not Flood                      *  
*                                                                 *  
*****/
```

project "PIPES"


```
include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "AnaGdef.PRO"
```

PREDICATES

```
oil_hydro_station(SYMBOL)
clean_station_fltr(SYMBOL)
air_escape_open(SYMBOL)
sel_valve_bkwrđ(SYMBOL)
damage_hydro_line(SYMBOL)
sel_mismarked(SYMBOL)
```

CLAUSES

```
clear_sounding_tube("Y"):- ask_ques_read_ans(OIL,"oil hydro
station"),oil_hydro_station(OIL).
clear_sounding_tube("N"):- add_problem(["sounding tube"]).
```

```
oil_hydro_station("Y"):- ask_ques_read_ans(FLTR,"clean
station fltr"),clean_station_fltr(FLTR).
oil_hydro_station("N"):- add_problem(["oil hydro station"]).
```

```
clean_station_fltr("Y"):- ask_ques_read_ans(AIR,"air escape
open"),air_escape_open(AIR).
clean_station_fltr("N"):- add_problem(["clean station
filter"]).
```

```
air_escape_open("Y"):- ask_ques_read_ans(SEL,"sel valve
bkwrđ"),sel_valve_bkwrđ(SEL).
air_escape_open("N"):- add_problem(["air escape open"]).
```

```
sel_valve_bkwrđ("Y"):- add_problem(["sel valve bkwrđ"]).
sel_valve_bkwrđ("N"):- ask_ques_read_ans(DAMAGE,"damage
hydro line"),damage_hydro_line(DAMAGE).
```

```
damage_hydro_line("Y"):- add_problem(["damage hydro line"]).
damage_hydro_line("N"):- ask_ques_read_ans(SEL,
"sel_mismarked"),sel_mismarked(SEL).
```

```
sel_mismarked("Y"):- add_problem(["sel mismarked"]).
```

```
sel_mismatched("N");- add_problem(["correct mismatch"]).
```

```
/******  
*  
*          Void Overflowing  
*  
*****/  
-
```

```
project "PIPES"
```

```
include "PipeGdoms.PRO"  
include "GlobDef.PRO"  
include "AnaGdef.PRO"
```

PREDICATES

```
trans_pump_run(SYMBOL)  
close_ovrfl_stop(SYMBOL)  
eductor_lit_off(SYMBOL)  
another_eductor(SYMBOL)  
close_edu_ovrfl_stop(SYMBOL)  
void_inb(SYMBOL)  
ballast_manifold(SYMBOL)  
strip_dischrg(SYMBOL)  
close_FM_valve(SYMBOL)  
close_strip(SYMBOL)  
close_sea_fld_valve(SYMBOL)  
close_valve_ovrfl_stop(SYMBOL)  
void_inb_contm(SYMBOL)  
dischr_contm(SYMBOL)  
cls_strip_ovrfl_valve(SYMBOL)  
jp5_trans(SYMBOL)  
close_jp5(SYMBOL)  
diesel_trans(SYMBOL)
```

CLAUSES

```
trans_pump_run("fuel");- losingfuel("Y").  
trans_pump_run("water");- ask_ques_read_ans(LIT,"eductor lit
```

```

off"),educator_lit_off(LIT).

/*****
*
*           Overflowing Water
*
*****/
educator_lit_off("Y"):- ask_ques_read_ans(CLOSE,"close
    edu ovrfl stop"),close_edu_ovrfl_stop(CLOSE).
educator_lit_off("N"):- ask_ques_read_ans(EDUCTOR,"another
    educator"),another_educator(EDUCTOR).

another_educator("Y"):- educator_lit_off("Y").
another_educator("N"):- ask_ques_read_ans(INBD,"void inb"),
    void_inb(INBD).

close_edu_ovrfl_stop("Y"):- add_problem(["close educator"]).
close_edu_ovrfl_stop("N"):- ask_ques_read_ans(BALLAST,
    "ballast manifold"),ballast_manifold(BALLAST).

void_inb("Y"):- ask_ques_read_ans(STRIP,"strip dischrg"),
    strip_dischrg(STRIP).
void_inb("N"):- close_edu_ovrfl_stop("Y")

ballast_manifold("Y"):- ask_ques_read_ans(CLOSE,"close sea
    fld valve"),close_sea_fld_valve(CLOSE).
ballast_manifold("N"):- ask_ques_read_ans(CLOSE,"close FM
    valve"),close_FM_valve(CLOSE).

strip_dischrg("Y"):- ask_ques_read_ans(CLOSE,"close strip"),
    close_strip(CLOSE).
strip_dischrg("N"):- close_edu_ovrfl_stop("Y").

close_FM_valve("Y"):- add_problem(["close FM valve"]).
close_FM_valve("N"):-

close_strip("Y"):- close_edu_ovrfl_stop("Y").
close_strip("N"):- add_problem(["close strip"]).

close_sea_fld_valve("Y"):-
close_sea_fld_valve("N"):- ask_ques_read_ans(CLOSE,
    "close valve ovrfl stop"),close_valve_ovrfl_stop(CLOSE).

```

```
close_valve_ovrfl_stop("Y"):- add_problem(["close ovrfl
stop"]).
close_valve_ovrfl_stop("N"):- indeterminate.
```

```

/*****
*
*           Overflowing Fuel
*
*****/

```

```
close_ovrfl_stop("Y"):- add_problem(["close ovrfl stop"]).
close_ovrfl_stop("N"):- ask_ques_read_ans(INBD,"void inb
contm"),void_inb_contm(INBD).
```

```
void_inb_contm("Y"):- ask_ques_read_ans(DIS,"dischr contm"),
dischr_contm(DIS).
void_inb_contm("N"):- ask_ques_read_ans(JP5,"jp5 trans"),
jp5_trans(JP5).
```

```
dischr_contm("Y"):- ask_ques_read_ans(CLOSE,"cls strip ovrfl
valve"),cls_strip_ovrfl_valve(CLOSE).
dischr_contm("N"):- void_inb_contm("N").
```

```
cls_strip_ovrfl_valve("Y"):- add_problem(["close strip
ovrfl"]).
cls_strip_ovrfl_valve("N"):- void_inb_contm("N").
```

```
jp5_trans("Y"):- ask_ques_read_ans(CLOSE,"close jp5"),
close_jp5(CLOSE).
jp5_trans("N"):- ask_ques_read_ans(DIESEL,"diesel trans"),
diesel_trans(DIESEL).
```

```
close_jp5("Y"):- add_problem(["close JP5"]).
close_jp5("N"):- jp5_trans("N").
```

```
diesel_trans("Y"):- add_problem(["diesel trans"]).
diesel_trans("N"):- ballast_manifold("N").
```

```

/*****
*
* Water in the Fuel Oil Service/ Fuel Oil Storage Tank
*
*****/

```

project "PIPES"

include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"
include "AnaGdef.PRO"

PREDICATES

nondeterm fuel_paste_test(SYMBOL)
nondeterm known_tank(INTEGER)
nondeterm outbd_fuel(SYMBOL)
nondeterm void_suction(SYMBOL)
nondeterm backflow(SYMBOL)
nondeterm fuel_pipes(SYMBOL)
nondeterm fuel_inboard(SYMBOL)
nondeterm water(SYMBOL)
nondeterm fuel_leaks(SYMBOL)
nondeterm pumps_refills_fuel(SYMBOL)
nondeterm fuel_cracks(SYMBOL)
nondeterm x_fer(SYMBOL)
nondeterm stripp_paste(SYMBOL)
nondeterm stripp_blockage(SYMBOL)
nondeterm fuel_stripp_open(SYMBOL)
nondeterm water_above(SYMBOL)
nondeterm skin_of_ship(SYMBOL)
nondeterm heavy_seas(SYMBOL)

CLAUSES

paste_test("Y"):- filled_from(TANK),
str_len(TANK,LEN),known_tank(LEN).
paste_test("N"):- ask_ques_read_ans(OTBD,"fuel outbd
fuel"),outbd_fuel(OTBD).

known_tank(0):- fot_paste_test("N").
known_tank(_):- ask_ques_read_ans(WATER_PASTE,"fuel
waterpaste"),!,fuel_paste_test(WATER_PASTE),!.

fuel_paste_test("Y"):-filled_from(TANK),
asserta(problem_tank(TANK),problem),

```

fot_paste("Y").
fuel_paste_test("N"):- paste_test("N").

outbd_fuel("Y"):- ask_ques_read_ans(VOID_SUCTION,
    "fuel void suction"),void_suction(VOID_SUCTION).
outbd_fuel("N"):- ask_ques_read_ans(BACKFLOW,"fuel
    backflow"),backflow(BACKFLOW).

void_suction("Y"):- add_problem(["void suction"]).
void_suction("N"):- position_in_window("action"),
    clearwindow,backflow("N").

backflow("Y"):- ask_ques_read_ans(CK_PIPES,
    "fuel check pipes"),fuel_pipes(CK_PIPES).
backflow("N"):- ask_ques_read_ans(FUEL,"fuel fuel
    inboard"),fuel_inboard(FUEL).

fuel_pipes("Y"):- add_problem(["void suction"]).
fuel_pipes("N"):- backflow("N").

fuel_inboard("Y"):- ask_ques_read_ans(LEAKS,
    "fuel pipes inboard"),fuel_leaks(LEAKS).
fuel_inboard("N"):- ask_ques_read_ans(WATER,"fuel
    water"),water(WATER).

water("Y"):- ask_ques_read_ans(REFILLS,
    "pumps refills fuel"),pumps_refills_fuel(REFILLS).
water("N"):- ask_ques_read_ans(XFER,"fuel trans
    conn"),x_fer(XFER).

fuel_leaks("Y"):- add_problem(["leaks"]).
fuel_leaks("N"):- ask_ques_read_ans(CRACKS,"fuel cracks"),
    fuel_cracks(CRACKS).

pumps_refills_fuel("Y"):- ask_ques_read_ans(LEAKS,"fuel
    pipes inboard"),fuel_leaks(LEAKS).
pumps_refills_fuel("N"):- water("N").

fuel_cracks("Y"):- add_problem(["cracks"]).
fuel_cracks("N"):- water("N").

x_fer("Y"):- add_problem(["x-fer"]).
x_fer("N"):- ask_ques_read_ans(STRIPP,"stripp paste"),

```

```

stripp_paste(STRIPP).

stripp_paste("Y"):- ask_ques_read_ans(BLOCK,"stripp
blockage"),stripp_blockage(BLOCK).
stripp_paste("N"):- ask_ques_read_ans(VALVE,"fuel stripp
open"),fuel_stripp_open(VALVE).

stripp_blockage("Y"):- add_problem(["stripp blockage"]).
stripp_blockage("N"):- ask_ques_read_ans(WATER,
"water above"),water_above(WATER).

fuel_stripp_open("Y"):- add_problem(["stripp open"]).
fuel_stripp_open("N"):- ask_ques_read_ans(WATER,
"water above"),water_above(WATER).

water_above("Y"):- ask_ques_read_ans(SKIN,"skin of
ship"),skin_of_ship(SKIN).
water_above("N"):-ask_ques_read_ans(HEAVY_SEAS,"heavy
seas"),heavy_seas(HEAVY_SEAS).

skin_of_ship("Y"):- add_problem(["skin of ship"]).
skin_of_ship("N"):- indeterminate.

heavy_seas("Y"):-add_problem(["heavy seas"]).
heavy_seas("N"):- indeterminate.

```

```

/*****
*
*      Water in the Fuel Oil Storage Tank
*
*****/

```

project "PIPES"

```

include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "AnaGdef.PRO"
include "PipeGdbase.PRO"

```

PREDICATES

```

nondeterm fot_outbd_fuel(SYMBOL)
nondeterm fot_void_suction(SYMBOL)

```

```

nondeterm fot_pipes(SYMBOL)
nondeterm fot_inboard(SYMBOL)
nondeterm fot_water(SYMBOL)
nondeterm fot_pumps_refills_fuel(SYMBOL)
nondeterm fot_cracks(SYMBOL)
nondeterm fot_water_above(SYMBOL)
nondeterm fot_skin_of_ship(SYMBOL)
nondeterm fot_backflow(SYMBOL)
nondeterm fot_leaks(SYMBOL)
nondeterm fot_stripp_paste(SYMBOL)
nondeterm fot_stripp_blockage(SYMBOL)
nondeterm fot_stripp_open(SYMBOL)
nondeterm fot_heavy_seas(SYMBOL)

```

CLAUSES

```

fot_paste_test("Y"):- filled_from(TANK),!,
    asserta(problem_tank(TANK),problem),
    ask_ques_read_ans(WATER_PASTE,"fot waterpaste"),
    fot_paste(WATER_PASTE),!.
fot_paste_test("N"):- ask_ques_read_ans(OTBD,"fuel outbd
    fuel"),fot_outbd_fuel(OTBD).
fot_paste("Y"):- ask_ques_read_ans(FILLED,"fot filled
    from"),!,assert_filled(FILLED).
fot_paste("N"):- fot_paste_test("N").

fot_outbd_fuel("Y"):- ask_ques_read_ans(VOID_SUCTION,
    "fuel void suction"),fot_void_suction(VOID_SUCTION).
fot_outbd_fuel("N"):- ask_ques_read_ans(BACKFLOW,"fot
    backflow"),fot_backflow(BACKFLOW).

fot_void_suction("Y"):- add_problem(["void suction"]).
fot_void_suction("N"):- position_in_window("action"),
    clearwindow,fot_backflow("N").

fot_backflow("Y"):- ask_ques_read_ans(CK_PIPES,
    "fot check pipes"),fot_pipes(CK_PIPES).
fot_backflow("N"):- ask_ques_read_ans(FUEL,"fuel fuel
    inboard"),fot_inboard(FUEL).

fot_pipes("Y"):- add_problem(["void suction"]).
fot_pipes("N"):- fot_backflow("N").

```



```

fot_inboard("Y"):- ask_ques_read_ans(LEAKS,
    "fuel pipes inboard"),fot_leaks(LEAKS).
fot_inboard("N"):- ask_ques_read_ans(WATER,"fuel
    water"),fot_water(WATER).

fot_water("Y"):- ask_ques_read_ans(REFILLS,
    "pumps refills fuel"),fot_pumps_refills_fuel(REFILLS).
fot_water("N"):- ask_ques_read_ans(STRIPP,"fot stripp
    paste"),fot_stripp_paste(STRIPP).

fot_leaks("Y"):- add_problem(["fot leaks"]).
fot_leaks("N"):- ask_ques_read_ans(CRACKS,"fuel cracks"),
    fot_cracks(CRACKS).

fot_pumps_refills_fuel("Y"):- ask_ques_read_ans(LEAKS,
    "fuel pipes inboard"),fot_leaks(LEAKS).
fot_pumps_refills_fuel("N"):- fot_water("N").

fot_cracks("Y"):- add_problem(["cracks"]).
fot_cracks("N"):- fot_water("N").

fot_stripp_paste("Y"):- ask_ques_read_ans(BLOCK,"fot
    stripp blockage"),fot_stripp_blockage(BLOCK).
fot_stripp_paste("N"):- ask_ques_read_ans(VALVE,"fot
    stripp open"),fot_stripp_open(VALVE).

fot_stripp_blockage("Y"):- add_problem(["fot stripp
    blockage"]).
fot_stripp_blockage("N"):- ask_ques_read_ans(WATER,
    "water above"),fot_water_above(WATER).

fot_stripp_open("Y"):- add_problem(["stripp open"]).
fot_stripp_open("N"):- ask_ques_read_ans(WATER,
    "water above"),fot_water_above(WATER).

fot_water_above("Y"):- ask_ques_read_ans(SKIN,"skin of
    ship"),fot_skin_of_ship(SKIN).
fot_water_above("N"):-ask_ques_read_ans(HEAVY_SEAS,"fot
    heavy seas"),fot_heavy_seas(HEAVY_SEAS).

fot_skin_of_ship("Y"):- add_problem(["skin of ship"]).
fot_skin_of_ship("N"):- indeterminate.

```

```
fot_heavy_seas("Y"):-add_problem(["heavy seas"]).
fot_heavy_seas("N"):- indeterminate.
```

```

/*****
*
*      Fuel Oil Service Tank Overflowing
*
*****/

```

```
project "PIPES"
```

```
include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"
include "AnaGdef.PRO"
```

PREDICATES

```

pump_overflow(INTEGER)
fill_valve_closed(SYMBOL)
strip_sys_used(SYMBOL)
eductor_strip(SYMBOL)
trs_valve_ovrfl(SYMBOL)
recirc_fostank(SYMBOL)
ovbd_dis_open(SYMBOL)
strip_lineup(SYMBOL)
ovbd_ovrfl_stop(SYMBOL)
close_recirc(SYMBOL)
fill_x_fer(SYMBOL)
correct_ovrfl_stop(SYMBOL)
strip_suct_valve(SYMBOL)
serv_suct_valve(SYMBOL)
recirc_valve(SYMBOL)
pipe_leak(SYMBOL)

```

CLAUSES

```

fos_pump_run_refuel("Y"):- ask_ques_read_ans(PUMP,"pump
overflow"), pump_overflow(PUMP).
fos_pump_run_refuel("N"):- ask_ques_read_ans(CLOSE,"fill
valve closed"),fill_valve_closed(CLOSE).

pump_overflow("Y"):- pump_run_refuel("N").

```

pump_overflow("N"):- ask_ques_read_ans(SYS,"strip sys
used"),strip_sys_used(SYS).

fill_valve_closed("Y"):- pump_overflow("N").
fill_valve_closed("N"):- ask_ques_read_ans(TRANS,"trs
valve ovrfl"),trs_valve_ovrfl(TRANS).

strip_sys_used("Y"):- ask_ques_read_ans(EDUCTOR,"eductor
strip"),eductor_strip(EDUCTOR).
strip_sys_used("N"):- ask_ques_read_ans(RECIRC,"recirc
fostank"),recirc_fostank(RECIRC).

eductor_strip("Y"):- ask_ques_read_ans(OVBD,"ovbd dis
open"),ovbd_dis_open(OVBD).
eductor_strip("N"):- ask_ques_read_ans(LINEUP,"strip
lineup"),strip_lineup(LINEUP).

trs_valve_ovrfl("Y"):- add_problem(["trs valve ovrfl"]).
trs_valve_ovrfl("N"):- fill_valve_closed("Y").

recirc_fostank("Y"):- ask_ques_read_ans(STOP,"close
recirc"),close_recirc(STOP).
recirc_fostank("N"):- ask_ques_read_ans(FILL,"fill
x-fer"),fill_x-fer(FILL).

ovbd_dis_open("Y"):- strip_sys_used("N").
ovbd_dis_open("N"):- ask_ques_read_ans(STOP,"ovbd ovrfl
stop"),ovbd_ovrfl_stop(STOP).

strip_lineup("Y"):- strip_sys_used("N").
strip_lineup("N"):- ask_ques_read_ans(STOP,"correct ovrfl
stop"),correct_ovrfl_stop(STOP).

ovbd_ovrfl_stop("Y"):- add_problem(["ovbd ovrfl stop"]).
ovbd_ovrfl_stop("N"):- strip_sys_used("N").

close_recirc("Y"):- add_problem(["close recirc"]).
close_recirc("N"):- recirc_fostank("N").

fill_x-fer("Y"):- ask_ques_read_ans(STRIPP,"strip suct
valve"),strip_suct_valve(STRIPP).
fill_x-fer("N"):- add_problem(["fill x-fer"]).

```

correct_ovrfl_stop("Y"):- add_problem(["correct ovrfl
stop"]).
correct_ovrfl_stop("N"):-strip_sys_used("N").

strip_suct_valve("Y"):- ask_ques_read_ans(SERV,"serv suct
valve"),serv_suct_valve(SERV).
strip_suct_valve("N"):- add_problem(["strip suct valve"]).

serv_suct_valve("Y"):- ask_ques_read_ans(RECIRC,"recirc
valve"),recirc_valve(RECIRC).
serv_suct_valve("N"):- add_problem(["serv suct valve"]).

recirc_valve("Y"):- ask_ques_read_ans(LEAK,"pipe
leak"),pipe_leak(LEAK).
recirc_valve("N"):- add_problem(["recirc valve"]).

pipe_leak("Y"):- add_problem(["pipe leak"]).
pipe_leak("N"):- indeterminate.

```

```

/*****
*
*      Fuel Oil Service Tank Losing Fuel
*
*****/

```

project "PIPES"

```

include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "AnaGdef.PRO"

```

PREDICATES

```

pump_pipe_leaks(SYMBOL)
fill_debris(SYMBOL)
refill(SYMBOL)
inspect_leaks(SYMBOL)
open_sound_tube(SYMBOL)
fos_los_fuel_cracks(SYMBOL)

```

CLAUSES

```

voids_oil("Y"):- ask_ques_read_ans(VOID,"which void has
oil"),retract_facts(assert(problem_tank(VOID),problem),
retrieve_contains,add_problem(["oil in void"])),

contains_fuel,fuel(_),ask_ques_read_ans(LOSINGFUEL,"losing
fuel"),!,losingfuel(LOSINGFUEL).

voids_oil("N"):- add_problem(["fos losing fuel"]),
ask_ques_read_ans(LEAKS,"pump pipe
leaks"),pump_pipe_leaks(LEAKS).

pump_pipe_leaks("Y"):- ask_ques_read_ans(FILL,"fill
debris"),fill_debris(FILL).
pump_pipe_leaks("N"):- add_problem(["leaks"]).

fill_debris("Y"):- ask_ques_read_ans(REFILL,"refill"),
refill(REFILL).
fill_debris("N"):- add_problem(["refills"]).

refill("Y"):- ask_ques_read_ans(LEAKS,"inspect leaks"),
inspect_leaks(LEAKS).
refill("N"):- ask_ques_read_ans(CRACKS,"fos los fuel
cracks"),fos_los_fuel_cracks(CRACKS).

inspect_leaks("Y"):- add_problem(["leaks"]).
inspect_leaks("N"):- refill("N").

fos_los_fuel_cracks("Y"):- add_problem(["cracks"]).
fos_los_fuel_cracks("N"):- ask_ques_read_ans(OPEN,"open
sound tube"),open_sound_tube(OPEN).

open_sound_tube("Y"):- indeterminate.
open_sound_tube("N"):- add_problem(["sound tube"]),

```

```

/*****
*
*      Fuel Oil Storage Tank Overflowing
*
*****/

```

project "PIPES"

```
include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"
include "AnaGdef.PRO"
```

PREDICATES

```
pump_overflow(INTEGER)
fill_valve_closed(SYMBOL)
strip_sys_used(SYMBOL)
educator_strip(SYMBOL)
trs_valve_ovrfl(SYMBOL)
ovbd_dis_open(SYMBOL)
strip_lineup(SYMBOL)
ovbd_ovrfl_stop(SYMBOL)
fill_x_fer(SYMBOL)
correct_ovrfl_stop(SYMBOL)
strip_suct_valve(SYMBOL)
pipe_leak(SYMBOL)
```

CLAUSES

```
fot_pump_run_refuel("Y"):- ask_ques_read_ans(PUMP,"fot
pump overflow"),pump_overflow(PUMP).
fot_pump_run_refuel("N"):- ask_ques_read_ans(CLOSE,"fot
fill valve closed"),fill_valve_closed(CLOSE).

pump_overflow("Y"):- pump_run_refuel("N").
pump_overflow("N"):- ask_ques_read_ans(SYS,"fot strip sys
used"),strip_sys_used(SYS).

fill_valve_closed("Y"):- pump_overflow("N").
fill_valve_closed("N"):- ask_ques_read_ans(TRANS,"for trs
valve ovrfl"),trs_valve_ovrfl(TRANS).

strip_sys_used("Y"):- ask_ques_read_ans(EDUCTOR,"fot
educator strip"),educator_strip(EDUCTOR).
strip_sys_used("N"):- ask_ques_read_ans(FILL,"fot fill
x_fer"),fill_x_fer(FILL).

educator_strip("Y"):- ask_ques_read_ans(OVBD,"fot ovbd dis
open"),ovbd_dis_open(OVBD).
educator_strip("N"):- ask_ques_read_ans(LINEUP,"fot strip
lineup"),strip_lineup(LINEUP).
```

```
trs_valve_ovrfl("Y"):- add_problem(["trs valve ovrfl"]).
trs_valve_ovr("N"):- fill_valve_closed("Y").
```

```
ovbd_dis_open("Y"):- strip_sys_used("N").
ovbd_dis_open("N"):- ask_ques_read_ans(STOP,"fot ovbd
ovrfl stop"),ovbd_ovrfl_stop(STOP).
```

```
strip_lineup("Y"):- strip_sys_used("N").
strip_lineup("N"):- ask_ques_read_ans(STOP,"fot correct
ovrfl stop"),correct_ovrfl_stop(STOP).
```

```
ovbd_ovrfl_stop("Y"):- add_problem(["ovbd ovrfl stop"]).
ovbd_ovrfl_stop("N"):- strip_sys_used("N").
```

```
fill_x_fer("Y"):- ask_ques_read_ans(STRIPP,"fot strip suct
valve"),strip_suct_valve(STRIPP).
fill_x_fer("N"):- add_problem(["fill x-fer"]).
```

```
correct_ovrfl_stop("Y"):- add_problem(["correct ovrf
stop"]).
correct_ovrfl_stop("N"):-strip_sys_used("N").
```

```
strip_suct_valve("Y"):- ask_ques_read_ans(LEAK,"fot pipes
leak"),pipes_leak(LEAK).
strip_suct_valve("N"):- add_problem(["strip suct valve"]).
```

```
pipe_leak("Y"):- add_problem(["pipe leak"]).
pipe_leak("N"):- indeterminate.
```

```
/******
*
*      Large Contaminated Tank Overflowing
*
*****/
```

```
project "PIPES"
```

```
include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"
include "AnaGdef.PRO"
```

PREDICATES

cont_pump_overflow(INTEGER)
cont_ovrfl_closed(SYMBOL)
other_sys_used(SYMBOL)
cont_ovbd_open(SYMBOL)
other_dischrg(SYMBOL)
other_ovrfl_closed(SYMBOL)
mmr_eductor(SYMBOL)
main_drain(SYMBOL)
gate_off_stem(SYMBOL)
cont_blkhd_stop(SYMBOL)
fot_pressure(SYMBOL)
close_trans_fuel(SYMBOL)

CLAUSES

mmr_strip_pump("Y"):- ask_ques_read_ans(PUMP,"cont pump overflow"),cont_pump_overflow(PUMP).
mmr_strip_pump("N"):- ask_ques_read_ans(OP,"other sys sed"),other_sys_used(OP).

cont_pump_overflow("Y"):- add_problem(["pump overflow"]).
cont_pump_overflow("N"):- mmr_strip_pump("N").

cont_ovrfl_closed("Y"):- add_problem(["overflow stops closed"]).
cont_ovrfl_closed("N"):- other_sys_used("N").

other_sys_used("Y"):- ask_ques_read_ans(CLOSE,"cont ovrfl closed"),cont_ovrfl_closed(CLOSE).
other_sys_used("N"):- ask_ques_read_ans(DISCH,"other dischrg"),other_dischrg(DISCH).

other_dischrg("Y"):- ask_ques_read_ans(OVBD,"other ovrfl closed"),other_ovrfl_closed(OVBD).
other_dischrg("N"):- ask_ques_read_ans(EDUCTOR,"mmr eductor"),mmr_eductor(EDUCTOR).

other_ovrfl_closed("Y"):- add_problem(["other ovrfl closed"]).
other_ovrfl_closed("N"):- other_dischrg("N").


```
mmr_eductor("Y"):- ask_ques_read_ans(EDUCTOR,"cont ovrbd
open"),cont_ovrbd_open(EDUCTOR).
mmr_eductor("N"):- ask_ques_read_ans(MAIN,"main drain"),
main_drain(MAIN).
```

```
cont_ovbd_open("Y"):- ask_ques_read_ans(STEM,"gate off
stem"),gate_off_stem(STEM).
cont_ovbd_open("N"):- add_problem(["ovbd open"]).
```

```
main_drain("Y"):- ask_ques_read_ans(PRESSURE,"fot
pressure"),fot_pressure(PRESSURE).
main_drain("N"):- ask_ques_read_ans(STOP,"cont bulkhd
stop"),cont_bulkhd_stop(STOP).
```

```
gate_off_stem("Y"):- add_problem(["gate off stem"]).
gate_off_stem("N"):- mmr_eductor("N").
```

```
cont_bulkhd_stop("Y"):- add_problem(["bulkhd stop"]).
cont_bulkhd_stop("N"):- main_drain("Y").
```

```
fot_pressure("Y"):- ask_ques_read_ans(CLOSE,"close trans
fuel"),close_trans_fuel(CLOSE).
fot_pressure("N"):- indeterminate.
```

```
close_trans_fuel("Y"):- add_problem(["trans pressure"]).
close_trans_fuel("N"):- indeterminate.
```

```
/******
*
*          QUESTION
*  Contains the questions for the PIPES system.
*
*****/
```

project "PIPES"

```
include "PipeGdoms.PRO"
include "GlobDef.PRO"
include "PipeGdbase.PRO"
```

CLAUSES

```

/*****
* The problem tank number is a mandatory piece of
* information and is entered in the following format:
*      8-119-9-V
*
*****/

```

```

question("tankno):-
  position_in_window("problem analysis"),
  write_screen(["What is the compartment number of the
  problem tank?"]).

```

```

/*****
* The tank number is required to locate the tank to be
* drawn and is entered in the following format:
*      8-119-11-F
*
*****/

```

```

question("single tank):- nl,
  write_screen(["Enter compartment number
  to be drawn: "]).

```

```

/*****
*      Water in the Fuel Oil Storage Tank
*****/

```

```

question("fot filled from):-
  position_in_window("action"),
  problem_tank(COMP_NUM),position_in_window("problem
  analysis"),
  write_screen(["Which fuel oil storage tank was",
  "fuel oil storage tank", COMP_NUM,"filled from?","(press
  enter if unknown)"]).

```

```

question("fot backflow):-
  otbd_void(QUES_VAR),position_in_window("action"),
  write_screen(["As a precaution to eliminate D.C. Void",
  QUES_VAR,
  "as a possible source", "of contamination, pump",
  "it to zero and leave empty", "until actual source of",
  "contamination to", "Fuel Oil Storage tank is",

```

"determined. ", "Monitor level of", QUES_VAR, "to see if",
 "its level rises from fuel", "leakage back through",
 "void suction tail pipe. ", "Alternative is to open",
 "gas free and visually", "inspect suction tail pipe in",
 QUES_VAR]), position_in_window("problem analysis"),
 write_screen(["Does oil flow back into the void?"]),
 question("fot check pipes"), QUES_VAR),
 position_in_window("action"),
 write_screen(["The tailpipe in D.C. void", QUES_VAR,
 "is backflowing fuel, pump down Fuel Oil storage",
 "tank and check the 2 1/2 inch copper-nickle void",
 "suction line passing through the service tank."]),
 position_in_window("problem analysis"),
 write_screen(["Are any pipes leaking?"])).

question("fot void suction"):- problem_tank(TANK),
 position_in_window("action"),
 write_screen(["Pump down fuel oil storage
 tank", TANK, "and inspect 2 1/2",
 "inch void suction line in storage tank for cracked or
 leaking couplings"]),
 position_in_window("problem analysis"),
 write_screen(["Are there any leaks in the 2 1/2 inch
 void suction line?"])).

question("fot heavy seas"):- position_in_window("action"),
 position_in_window("problem analysis"), clearwindow, nl,
 write_screen(["Has the ship been operating recently in
 heavy seas, which",
 "could have backflowed through the overflow piping into
 the service tank?"])).

question("fot stripp open"):-
 problem_tank(TANK), position_in_window("action"),
 write_screen(["Disassemble the Fuel Oil Storage
 Ballast/", "stripping valve for", TANK, "
 Inspect for debris under the seat",
 "disc off stem, and correct, free operation"]),
 position_in_window("problem analysis"),
 write_screen(["Was the ballast/ stripping valve damaged
 or inoperative?"])).

question("fot stripp paste"):-

```

problem_tank(TANK),position_in_window("action"),nl,
write_screen(["Strip Fuel oil storage tank",TANK,
"and perform water paste test"]),
position_in_window("problem
analysis"),write_screen(["Was",TANK,
"successfully stripped of water?"])).

```

```

question("fot stripp blockage):-
problem_tank(TANK),position_in_window("action"),
write_screen(["Disassemble the Fuel Oil Storage
Ballast/","stripping valve for",TANK,".
Check for blockage holding the valve in the",
"open position, allowing backflow of water into the
service tank."]),
position_in_window("problem
analysis"),write_screen(["Was the ballast/ stripping",
"valve blocked open or partially open?"])).

```

```

/*****
*           Water in the Fuel Oil Service Tank           *
*****/

```

```

question("fuel filled from):-
position_in_window("action"),
problem_tank(COMP_NUM),position_in_window("problem
analysis"),
write_screen(["Which fuel oil storage tank was",
"fuel oil service tank", COMP_NUM,"filled from?"])).

```

```

question("fuel waterpaste):- filled_from(QUES_VAR),
position_in_window("action"),
write_screen(["Do a waterpaste test on fuel oil storage
tank ",QUES_VAR,
"."]),position_in_window("problem analysis"),
write_screen(["Is the test positive for water?"])).

```

```

question("fuel outbd fuel):-
position_in_window("action"),
otbd_void(QUES_VAR),
write_screen(["Sound the outboard D.C. void",QUES_VAR]),
position_in_window("problem analysis"),
write_screen(["Does the sounding tape indicate the

```

presence of fuel",
"(May require a waterpaste test)?"]).

question("fuel backflow"):-
otbd_void(QUES_VAR),position_in_window("action"),
write_screen(["As a precaution to eliminate D.C. Void",
QUES_VAR,"as a possible source of contamination,
pump","it to zero and leave", "empty until actual
source of","contamination to", "Fuel Oil Service tank
is","determined. ","Monitor level of",QUES_VAR,"to see
if","its level rises from","fuel leakage back through",
"void suction tail pipe. ","Alternative is to open",
"gas free and visually","inspect suction tail pipe in",
QUES_VAR]),position_in_window("problem analysis"),
write_screen(["Does oil flow back into the void?"])).

question("fuel check pipes"):- otbd_void(QUES_VAR),
position_in_window("action"),
write_screen(["The tailpipe in D.C. void", QUES_VAR,
"is backflowing fuel,pump down Fuel Oil service",
"tank and check the 2 1/2 inch copper-nickle void",
"suction line passing through the service tank."]),
position_in_window("problem analysis"),
write_screen(["Are any pipes leaking?"])).

question("fuel fuel inboard"):-
position_in_window("action"),
position_in_window("problem analysis"),
print("inboard voids","problem analysis"),
write_screen(["Do soundings or waterpaste test indicate
there is fuel","in any of these voids?"])).

question("fuel pipes inboard"):-
position_in_window("action"),
write_screen(["The inboard D.C. void indicates fuel is
present,"]),question("leaking").

question("fuel void suction"):- problem_tank(TANK),
position_in_window("action"),
write_screen(["Pump down fuel oil service
tank",TANK,"and inspect 2 1/2",
"inch void suction line in service tank for cracked or
leaking couplings"])),

```
position_in_window("problem analysis"),
write_screen(["Are there any leaks in the 2 1/2 inch
void suction line?"])).
```

```
question("pumps refills fuel"):-
position_in_window("action"),
write_screen(["Pump inboard void empty", "Take soundings
and do waterpaste",
"test after serveral hours to see if voids refill with
fuel."]),position_in_window("problem analysis"),
write_screen(["Do any voids in the string refill with
fuel?"])).
```

```
question("leaking"):- position_in_window("action"),
write_screen(["pump down void, open, gas free, and check
for leaks",
"on: \n",
"  4 inch fuel oil service suction \n",
"  5 inch fuel oil transfer to FOS tank \n",
"  2 inch recirc pipe to FOS tank \n",
"  2 1/2 inch stripping pipe to FOS tank \n",
"  heating coils \n"]),position_in_window("problem
analysis"),
write_screen([" Are any pipes leaking?"])).
```

```
question("fuel cracks"):- problem_tank(COMP_NUM),
position_in_window("action"),
write_screen(["Piping is not leaking. As a precaution, ",
"do not flood inboard D.C. voids. ", "Pump, open, gas",
"free, and inspect", COMP_NUM, " for cracks in the",
"longitudinal bulkheads near", "the weld to the
transverse",
"bulkhead. Check for leaks", "at cracks in pipe and
heating",
"coil penetrations in the", "lower part of the void."]),
position_in_window("problem analysis"),
write_screen(["Are there any cracks?"])).
```

```
question("fuel water"):- position_in_window("action"),
position_in_window("problem analysis"),
print("inboard voids", "problem analysis"),
write_screen(["Is there water in these voids?"])).
```

question("fuel pump"):- position_in_window("action"),
write_screen(["Pump void empty as a precaution"]),
position_in_window("problem analysis"),
write_screen(["Does void refill with fuel?"])).

question("fuel trans conn"):-
position_in_window("action"),
position_in_window("problem analysis"),
write_screen(["Is the emergency connection from the Fuel
Oil Service",
"pump suction piping to the Fuel Oil Transfer system ",
"open (normally a locked closed valve)?"])).

question("heavy seas"):- position_in_window("action"),
position_in_window("problem analysis"),clearwindow,nl,
write_screen(["Has the ship been operating recently in
heavy seas, which",
"could have backflowed through the overflow piping into
the service tank?"])).

question("fuel stripp open"):- problem_tank(TANK),
position_in_window("action"),
write_screen(["Disassemble the Fuel Oil Service",
"stripping valve for",TANK,". Inspect for debris under
the seat",
"disc off stem, and correct, free operation"])),
position_in_window("problem analysis"),
write_screen(["Was the stripping valve damaged or
inoperative?"])).

question("stripp paste"):-
problem_tank(TANK),position_in_window("action"),
write_screen(["Strip Fuel oil service tank",TANK,
"and perform water paste test"]),
position_in_window("problem
analysis"),write_screen(["Was",TANK,
"successfully stripped of water?"])).

question("stripp blockage"):-
problem_tank(TANK),position_in_window("action"),
write_screen(["Disassemble the Fuel Oil Service",
"stripping valve for",TANK,". Check for blockage
holding the valve in the",

```
"open position, allowing backflow of water into the
service tank."]),
position_in_window("problem
analysis"),write_screen(["Was the stripping",
"valve blocked open or partially open?"])).
```

```
question("water above"):- position_in_window("action"),
problem_tank(TANK),above_tank(ABOVE),
write_screen(["Open and inspect the compartment
above",TANK,"which is",
ABOVE,"and check for water"]),position_in_window("problem
analysis"),
write_screen(["Is there water in",ABOVE])).
```

```
question("skin of ship"):- position_in_window("action"),
write_screen(["Inspect manhole cover, flange coaming,
gasket, deck,",
"pipe penetratings, and sounding tube/ air escape piping
on decks above",
"for deterioration and possible avenues of leakage"]),
position_in_window("problem analysis"),
write_screen(["Was a path for water to leak into the
service tank found",
"on the fourth deck?"])).
```

```
/******
*
*      Fuel Oil Service Tank Losing Fuel
*
*****/
```

```
question("voids have oil"):- position_in_window("action"),
position_in_window("problem analysis"),
print("voids in string","problem analysis"),
write_screen(["Do any of the above Voids contain
fuel?"])).
```

```
question("which void has oil"):-
position_in_window("action"),
position_in_window("problem analysis"),
print("voids in string","problem analysis"),
write_screen(["Which Void contains fuel?"])).
```



```

/*****
*
*      Void Pumps but Refills with Water
*
*****/
question("sea valve leak):-
  problem_tank(TANK),above_tank(ABOVE),
  position_in_window("action"),
  write_screen(["Empty",TANK,"remove tank",
  "top cover in",ABOVE,"gas free and visually observe sea
  valve","for leakage."]),position_in_window("problem
  analysis"),write_screen([
  "Does sea valve leak?"]);

question("void backflow):-
  position_in_window("action"),write_screen([
  "While void is empty and suction secured","gas free and
  enter void",
  "check void suction tailpipe for backflow."]),
  position_in_window("problem analysis"),
  write_screen(["Is there backflow?"]);

question("cycle sea valve):-
  problem_tank(TANK),position_in_window("action"),
  write_screen(["Hydraulically cycle sea valve
  for",TANK,"and then pump",
  TANK,"empty."]),position_in_window("problem analysis"),
  write_screen(["Observe sea valve",
  "did cycling the sea valve stop leakage?"]);

question("manifold
  pressure):-position_in_window("action"),write_screen([
  "Break flange of void suction valve in machinery
  space","Do not remove",
  "all fasteners."]),position_in_window("problem
  analysis"),write_screen([
  "Is manifold under water pressure?"]);

```

question("void cracks"):- position_in_window("action"),
position_in_window("problem
analysis"),write_screen(["Are there",
"any cracks in welds at transverse bulkheads",
"or in piping penetrations?"])).

question("overboard"):-position_in_window("action"),
write_screen([
"Check line up of machinery space or pumproom
educator.", " If the",
"firemain supply in open"]),position_in_window("problem
analysis"),
write_screen(["Is the overboard discharge valve",
"of the educator open?"])).

question("void suction"):- position_in_window("action"),
position_in_window("problem analysis"),
write_screen(["Are other void suction valves open?"])).

question("open ovbd"):-
position_in_window("action"),write_screen([
"Open the overboard discharge."]),
position_in_window("problem analysis"),
write_screen(["Is pressure at the void valve manifold
removed?"])).

question("close suction"):-position_in_window("action"),
write_screen(["Close all other suction valves in the
machinery space or pumproom."]),
position_in_window("problem analysis"),
write_screen(["Is the void manifold under pressure?"])).

question("bulkhd stopsopen"):-
position_in_window("action"),
write_screen(["Verify that bulkhead stops in the main",
"drain system",
"for the affected machinery space/ pumproom are
closed."]),
position_in_window("problem analysis"),
write_screen(["Are bulkhead stops closed?"])).

question("flooding valve"):-position_in_window("action"),
position_in_window("problem analysis"),

```
write_screen(["Is the sea flooding valve to the  
ballast system closed and locked?"])).
```

```
question("close bulkhd stops):-  
position_in_window("action"),  
write_screen(["Close main drainage systems bulkhead",  
"stops."]),position_in_window("problem analysis"),  
write_screen(["Is the",  
"void manifold under pressure?"])).
```

```
question("close flood valve):-  
position_in_window("action"),  
write_screen(["Close sea flooding valve."]),  
position_in_window("problem analysis"),  
write_screen(["Is the void manifold under pressure?"])).
```

```
question("holes):-  
above_tank(ABOVE),position_in_window("action"),  
write_screen(["Open, gas free (if necessary)and",  
"inspect the 4th deck compartment",ABOVE,"for holes in  
the deck coaming, tank top, gasket or penetrations."]),  
position_in_window("problem analysis"),  
write_screen(["Were holes/",  
"breaks found in ",ABOVE,"?"])).
```

```
question("inboard void):- problem_tank(TANK),  
position_in_window("action"),print("inboard  
voids", "action"),  
write_screen(["The above voids are located",  
"inboard of",TANK,". Pump down these voids, open, gas  
free", "and inspect",  
"2 1/2 inch copper nickle void suction line to",TANK]),  
position_in_window("problem analysis"),write_screen([  
"Are any lines leaking?"])).
```

```
/*  
*  
*      Oil in the Void Tank      *  
*  
*  
*/
```

```
question("losing fuel):- position_in_window("action"),  
position_in_window("problem analysis"),
```

```
print("fuel tanks","problem analysis"),
!,problem_tank(COMP_NUM),
write_screen(["The above are fuel oil system
associated pipes contained in ",COMP_NUM,". Are any of
these tanks losing fuel?"])).
```

```
question("cracks in tank"):- problem_tank(COMP_NUM),
position_in_window("action"),
write_screen(["Pump, open, gas free,",
"and inspect", COMP_NUM, "for cracks in the
longitudinal bulkheads near the weld", "to the
transverse bulkhead.",
"Check for leaks at cracks", "in pipe and heating coil",
"penetrations in the lower part of the void."]),
position_in_window("problem analysis"),
write_screen(["Are there any cracks in ",COMP_NUM,"?"])).
```

```
question("fourth deck"):- retrieve_adjacent(above),
above_tank(TANK),
position_in_window("action"),
write_screen(["Inspect fourth deck compartment for oil",
"on deck, loose or deteriorated tank top to void",
"below holes in deck, holes in pipe", "penetrations (air
escape, ", "sounding tube, or TLI."]),
position_in_window("problem analysis"),
write_screen(["Are there any leaks in",TANK,"?"])).
```

```
question("void losing fuel water"):-
position_in_window("action"),
position_in_window("problem analysis"),
print("string fuel tanks","problem analysis"),
write_screen(["Are any of the above fuel tanks",
"losing fuel/filling with water?"])).
```

```
question("which tank losing fuel"):-
position_in_window("action"),position_in_window("problem
analysis"),write_screen(["Which tank is losing fuel?"])).
question("check valve"):- position_in_window("action"),
position_in_window("problem analysis"),
write_screen(["Is the void suction valve open?"])).
```

```
question("check debris"):- position_in_window("action"),
write_screen(["Disassemble the void suction valve and",
```

"remove bonnet from manifold. Inspect debris/ blockage",
"under the seat.")),
position_in_window("problem analysis"),
write_screen(["Was there blockage/ debris under the
valve?"])).

question("void pump):- position_in_window("action"),
write_screen(["Open void suction valve."]),
position_in_window("problem analysis"),
write_screen(["Does the void pump?"])).

question("clean valve):- position_in_window("action"),
write_screen(["Clean suction valve."]),
position_in_window("problem analysis"),
write_screen(["Will the void pump?"])).

question(manifold):- position_in_window("action"),
position_in_window("problem analysis"),
write_screen(["Is manifold under water pressure at
suction valve?"])).

question(educator):- position_in_window("action"),
write_screen(["Check educator lineup.", "Check overboard
discharge suction valve firemain supply.", "(Note: Check
valve clattering",
"means educator is drawing air)"]),
position_in_window("problem analysis"),
write_screen(["Is educator lined up incorrectly?"])).

question(bilge):- position_in_window("action"),
write_screen(["Check other valves in the machinery",
"space or pumproom."]),
position_in_window("problem analysis"),
write_screen(["Are bilgewells and other void suction",
"valves all closed?"])).

question(submersible):- problem_tank(TANK),
position_in_window("action"),
write_screen(["Open", TANK, "pump with submersible pump",
"(Note: NSTM permits pumping oil)"]),
position_in_window("problem analysis"),
write_screen(["Will tank not pump down?"])).

```
question(tailpipe):- position_in_window("action"),
    position_in_window("problem analysis"),
    write_screen(["Is there backflooding through the",
        "tailpipe?"])).
```

```
question("ovbd discharge):- position_in_window("action"),
    position_in_window("problem analysis"),
    write_screen(["Is the eductor overboard discharge",
        "open?"])).
```

```
question("sounding tube):- position_in_window("action"),
    write_screen(["Trace out sounding tube and air",
        "escape."]),
    position_in_window("problem analysis"),
    write_screen(["Is void flooding through breaks in",
        "sounding tube or air escapes?"])).
```

```
question("unable pump losing fuel):- problem_tank(TANK),
    position_in_window("action"), write_screen([
        "Pump, open, and gas free",TANK,".", "Inspect fuel oil",
        "transfer/stripping/suction",
        "line for leaks."]),
    position_in_window("problem analysis"),
    write_screen(["Are pipes leaking?"])).
```

```
question("bulkhd stops):- position_in_window("action"),
    position_in_window("problem analysis"),
    write_screen(["Are the bulkhead stops closed?"])).
```

```
question("close valve):- position_in_window("action"),
    write_screen(["Close all valves."]),
    position_in_window("problem analysis"),
    write_screen(["Will void pump now?"])).
```

```
question(bulkhd_stops_close):-
    position_in_window("action"),
    write_screen(["Close the bulkhead stops."]),
    position_in_window("problem analysis"),
    write_screen(["Will void pump now?"])).
```

```
question("leaky pipes):- position_in_window("action"),
    problem_tank(TANK),!,retrieve_contains,!,
```

```

compare_string_void_to_contains,
print("string voids in tank","action"),nl,
write_screen(["Pump down, open, gas free, and check",
"each of the above",
"voids for leaks on the 2 1/2 inch stripping pipe to",
TANK]),position_in_window("problem analysis"),
write_screen(["Are any pipes leaking?"]);
/*****
*
*          Void Overflowing
*
*****/

question("pump run refuel") :- position_in_window("problem
analysis"),
write_screen(["Are any fuel oil transfer pumps running",
"or is the","fuel oil transfer system pressurized due to",
"refueling?"]);

question("pump overflow") :- problem_tank(TANK),
position_in_window("action"),
write_screen(["Secure fuel oil transfer pumps or isolate
the transfer",
"system to the section serving service tank",TANK]),
position_in_window("problem analysis"),
write_screen(["Is the service tank still overflowing?"]);

/*****
*
*          Overflowing Water
*
*****/

question("strip sys used") :- position_in_window("problem
analysis"),write_screen(["Is the fuel oil stripping
system being used?"]);

question("trs valve ovrf") :- problem_tank(TANK),
position_in_window("action"),
write_screen(["Close the fill/transfer valve to ",TANK]),
position_in_window("problem analysis"),
write_screen(["Did service tank",TANK,"stop

```

overflowing?"])).

question("eductor strip") :- position_in_window("problem analysis"),write_screen(["Is the eductor being used to strip?"])).

question("recirc fostank") :- problem_tank(TANK), position_in_window("problem analysis"), write_screen(["Are the fuel oil service pumps recirculating to",TANK,"?"])).

question("ovbd dis open") :- position_in_window("problem analysis"),write_screen(["Is the eductor overboard discharge open?"])).

question("strip lineup") :- position_in_window("problem analysis"),write_screen(["Is the stripping pump lined up correctly?"])).

question("close recirc") :- problem_tank(TANK), position_in_window("action"), write_screen(["Secure recirc to",TANK]), position_in_window("problem analysis"), write_screen(["Did the overflow from service tank",TANK,"stop?"])).

question("fill x_fer") :- problem_tank(TANK), position_in_window("action"), write_screen(["Disassemble and inspect the fill/transfer valve to",TANK]), position_in_window("problem analysis"), write_screen(["Was the fill/transfer valve operating correctly?"])).

question("ovbd ovrfl stop") :- problem_tank(TANK), position_in_window("action"), write_screen(["Open the eductor overboard discharge"]), position_in_window("problem analysis"), write_screen(["Is service tank",TANK,"still overflowing?"])).


```

/*****
*
*           Overflowing Fuel
*
*****/
question("correct ovrfl stop") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Verify correct line up of stripping
pump"]),position_in_window("problem analysis"),
    write_screen(["Is service tank",TANK,"still
overflowing?"]).

question("strip suct valve") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Disassemble and inspect stripping valve to
service tank",TANK]), position_in_window("problem
analysis"),write_screen(["Is the stripping valve
to",TANK,"operating correctly?"]).

question("serv suct valve") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Disassemble and inspect the service suction
valve to",TANK]), position_in_window("problem analysis"),
    write_screen(["Is the service suction valve to",TANK,
"operating correctly?"]).

question("recirc valve") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Disassemble and inspect the recirc valve
to",TANK]),position_in_window("problem analysis"),
    write_screen(["Is the recirc valve to",TANK,"operating
correctly?"]).

question("pipe leak") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Open, gas free, and inspect piping and
structure in",TANK]), position_in_window("problem
analysis"),write_screen(["Are any pipes or is any
structure in",TANK,"ruptured or","leaking?"]).

```

```

/*****
*
*      Fuel Oil Storage Tank Overflowing
*
*****/

```

```

question("fot pump run refuel") :-
  position_in_window("problem analysis"),
  write_screen(["Are any fuel oil transfer pumps running or
is the","fuel oil transfer system pressurized due to
refueling?"]).

```

```

question("fot pump overflow") :- problem_tank(TANK),
  position_in_window("action"),
  write_screen(["Secure fuel oil transfer pumps or isolate
the transfer",
"system to the section serving storage tank",TANK]),
  position_in_window("problem analysis"),
  write_screen(["Is the storage tank still overflowing?"]).

```

```

question("fot strip sys used") :-
  position_in_window("problem analysis"),
  write_screen(["Is the fuel oil stripping system being
used?"]).

```

```

question("fot trs valve ovrfl") :- problem_tank(TANK),
  position_in_window("action"),
  write_screen(["Close the fill/transfer valve to ",TANK]),
  position_in_window("problem analysis"),
  write_screen(["Did storage tank",TANK,"stop
overflowing?"]).

```

```

question("fot eductor strip") :- position_in_window("problem
analysis"),write_screen(["Is the eductor being used to
strip?"]).

```

```

question("fot ovbd dis open") :- position_in_window("problem
analysis"),write_screen(["Is the eductor overboard
discharge open?"]).

```

```

question("fot strip lineup") :- position_in_window("problem
analysis"),write_screen(["Is the stripping pump lined up
correctly?"]).

```

```
question("fot fill x_fer") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Disassemble and inspect the fill/transfer
valve to",TANK]), position_in_window("problem analysis"),
    write_screen(["Was the fill/transfer valve operating
correctly?"])).
```

```
question("fot ovbd ovrfl stop") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Open the eductor overboard discharge"]),
    position_in_window("problem analysis"),
    write_screen(["Is storage tank",TANK,"still
overflowing?"])).
```

```
question("fot correct ovrfl stop") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Verify correct line up of stripping
pump"]),position_in_window("problem analysis"),
    write_screen(["Is storage tank",TANK,"still
overflowing?"])).
```

```
question("fot strip suct valve") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Disassemble and inspect stripping valve to
storage tank",TANK]), position_in_window("problem
analysis"),write_screen(["Is the stripping valve
to",TANK,"operating correctly?"])).
```

```
question("fot pipe leak") :- problem_tank(TANK),
    position_in_window("action"),
    write_screen(["Open, gas free, and inspect piping and
structure in",TANK]), position_in_window("problem
analysis"),write_screen(["Are any pipes or is any
structure in",TANK,"ruptured or","leaking?"])).
```

```
/******
*
*          SOLUTION.PRO
*
*****/
```

project "PIPES"

include "PipeGdoms.PRO"

include "PipeGdbase.PRO"

include "GlobDef.PRO"

```

/*****
*
*   write_solution predicate is used to determine the
*   appropriate solution to the queries from the user
*   based on the responses given to the
*   proposed questions and facts collected from the
*   pipesystem database. These facts are asserted in
*   the problem(LIST_OF_PROBLEMS) fact.
*   The first object in the write_solution predicate
*   is the original problem selected from the problem
*   menu and the clauses are grouped by this
*   object for program clarity.
*
*   Solution builds the screen display and writes the
*   appropriate solution
*
*****/
```

CLAUSES

```

solution:- position_in_window("action"),clearwindow,
            makewindow(12,23,7,"Solution",8,5,10,70,1,255,
            "\201\187\200\188\205\186"),problem(PROBLEMLIST),!,
            position_in_window("solution"),
            write_solution(PROBLEMLIST),retract_facts,
            readchar(_),removewindow(12,1),!.
solution:- retract_facts.
```

```

/*****
*
*           OIL IN A VOID TANK
*
*****/
```

```

write_solution(["oil in void","losing fuel"]):-
    problem_tank(COMP_NUM),
```

```

fuel(COMPNO2),write_screen(["A fuel oil",
"transfer/stripping/suction line to",COMPNO2,
"is cracked/holed in",COMP_NUM,".", "Pump, open, clean,",
"gas free, and repair ruptured pipe."])).

```

```

write_solution(["oil in void","cracks"]):-
problem_tank(COMP_NUM),
retrieve_adjacent(all),print("adjacent
fuel",solution),!,
write_screen(["The cause of the fuel in void",
COMP_NUM,"is a crack in the bulkhead into",
"one of the above tanks"])).

```

```

write_solution(["oil in void","4th deck"]):-
above_tank(FOURTH_DECK),
problem_tank(COMP_NUM),write_screen(["Fuel in",COMP_NUM,
"is coming from a break in",FOURTH_DECK,
"above."])).

```

```

/*****
*
*      UNABLE TO PUMP VOID TANK
*
*****/

```

```

write_solution(["unable to pump","leaks"]):-
problem_tank(TANK),
write_screen(["The cause of ",TANK," not pumping",
"is that it is refilling from a fuel oil tank in the",
"string through a leak in the",
"fill/transfer/stripping/service piping inside",TANK])).

```

```

write_solution(["unable to pump","pumps"]):-
problem_tank(TANK),
write_screen(["The cause of ",TANK," not pumping is",
"that the correct suction valve in the machinery",
"space/pump room was","not open."])).

```

```

write_solution(["unable to pump","clean valve"]):-
problem_tank(TANK), write_screen(["The cause of",
TANK,"not","pumping is debris/foriegn object/rag",
"blocking the suction","valve from functioning."])).

```

write_solution(["unable to pump","eductor"]):-
problem_tank(TANK), write_screen(["The cause
of",TANK,"not",
"pumping is one of the following:"
"A. eductor overboard discharge not open",
"B. eductor firemain supply not open",
"C. eductor suction valve not open",
"D. firemain isolated from firemain supply valve"])).

write_solution(["unable to pump","valves closed"]):-
problem_tank(TANK), write_screen(["The cause of ",TANK,
"not pumping is another suction valve or bilge well",
"valve in the","machinery space/pumproom is open",
"causing" a loss of vacuum to",TANK]).

write_solution(["unable to pump","bilge"]):-
problem_tank(TANK), write_screen(["The cause of",
TANK,"not pumping is a bulkhead stop to another",
"machinery space/", "pumproom is open."]).

write_solution(["unable to pump","submersible"]):-
problem_tank(TANK), write_screen(["The cause of",TANK,
"not pumping is the sea flooding valve",
"is stuck open or has debris",
"nder the seat or there is a large opening to",
"the sea."]).

write_solution(["unable to pump","cracks"]):-
problem_tank(TANK), write_screen(["The cause",
"of",TANK,"not pumping is cracks in (1)bulkhead",
"structure generally near ","welds to tranverse",
"bulkheads or (2)around piping ","penetrations into",
"adjacent tanks or (3)to the sea."]).

write_solution(["unable to pump","sounding tube"]):-
problem_tank(TANK), write_screen(["The cause of", TANK,
"not pumping is that it is reflooding through a break",
"in the sounding tube outside the tank or a break in",
"the air escape","allowing water to flow from an",
"exterior source into the void."]).

write_solution(["unable to pump","string leaks"]):-

```

problem_tank(TANK), write_screen(["The cause of",
TANK,"not pumping is it is refilling from a leak",
"in a void suction line","to another void in the",
"string."])).

```

```

/*****
*
*      WATER IN A FUEL OIL SERVICE TANK
*
*****/

```

```

write_solution(["water in tank","outboard void"]):-
otbd_void(QUES_VAR),
write_screen(["The outboard D.C. void", QUES_VAR,
"indicates oil present, pump down, open and gas free",
"problem Fuel Oil Service tank and check the 2 1/2",
"inch copper-nickle void suction line for leakage",
"particularly at silver braze fittings."])).

```

```

/*****
*
*      Water in the Fuel Oil Service/ Fuel Oil Storage Tank
*
*****/

```

```

write_solution(["water in fuel","void suction"]) :-
problem_tank(TANK),
otbd_void(OUTBOARD),!,
write_screen(["The void suction line to the",
"outboard void",OUTBOARD,
"is ruptured in ",TANK,".", "Empty, clean, gas free,",
"and repair the",
"break in the 2.5 inch","void suction line in",TANK])).

```

```

write_solution(["water in fuel","leaks"]) :-
inbd_void(INBOARD), write_screen(["A fuel oil ",
"service/ transfer/ stripping pipe is leaking in",
INBOARD,". Do not",
"flood ",INBOARD,". Empty, clean, gas free,and",
"repair the break in the failed pipe in",INBOARD ])].

```

```

write_solution(["water in fuel","cracks"]) :-

```

```

problem_tank(TANK), inbd_void(INBOARD),
write_screen(["There are cracks in ",INBOARD,
"allowing water to ", "leak into ",TANK,
" Do not flood ",INBOARD,". Empty, clean, gas ",
"free ",INBOARD, "and",TANK,
"and repair bulkhead cracks." ])).

```

```

write_solution(["water in fuel","x-fer"]) :-
write_screen(["The emergency connection from ",
"the fuel oil service pump suction piping directly to
the fuel oil","transfer system was open or leaking",
"through. Close and lock the ",
"valve or repair it. " ])).

```

```

write_solution(["water in fuel","stripp blockage"]) :-
problem_tank(TANK),
write_screen(["The stripping valve to ",TANK,
"is being held open by ", "foriegn matter or is",
"damaged, allowing water to backflow through ",
"the stripping line into the service tank."])).

```

```

write_solution(["water in fuel","stripp open"]) :-
problem_tank(TANK),
write_screen(["The stripping valve to ",TANK,
"is inoperative, the "," disc is off the stem, or",
"the disc is jammed in the closed position ",
"preventing",TANK," from being stripped."])).

```

```

write_solution(["water in fuel","skin of ship"]) :-
problem_tank(TANK),
above_tank(ABOVE),
write_screen(["The source of water in ",TANK,
"is a leak from the ","fourth deck compartment",
"above, ",ABOVE,". The tank top, tank top ",
"flange coaming, deck, gasket, stuffing tube,or a",
"piping penetration is allowing water to enter ",
TANK ])).

```

```

write_solution(["water in fuel","heavy seas"]) :-
problem_tank(TANK),
write_screen(["Water from high seas is backing through",
"the service tank overflow piping through a stuck or",
"leaking check valve, back ","into the service",tank,"

```


TANK).

```
/******  
*  
*   WATER IN THE FUEL OIL STORAGE TANK  
*  
*****/
```

```
write_solution(["water in fuel","fot leaks"]):-  
  inbd_void(INBOARD), write_screen(["A fuel oil ",  
    "transfer/ ballast/ stripping pipe is leaking in",  
    INBOARD,". Do not flood ",INBOARD,  
    ". Empty, clean, gas free, and repair the",  
    "break in the failed pipe in",INBOARD ]).
```

```
/******  
*  
*   VOID PUMPS BUT REFILLS WITH WATER  
*  
*****/
```

```
write_solution(["pumps but refills","industrial  
repair"]):- write_screen(["The sea flooding valve is",  
  "leaking through and will not reseal. ",  
  "The valve must be cofferdamed and repaired by an",  
  "industrial activity."]).
```

```
write_solution(["pumps but refills","cycle sea valve"]):-  
  write_screen(["Debris under the seat of the sea valve",  
    "or other obstruction",  
    "was released, allowing the valve to be resealed when",  
    "cycled."]).
```

```
write_solution(["pumps but refills","void cracks"]):-  
  problem_tank(TANK),  
  write_screen(["Cracks in bulkheads are allowing",  
    TANK,"to refill","from adjacent flooded voids.",  
    "Pumping adjacent voids will",  
    "remove source of water. Industrial repairs",  
    "required."]).
```

```
write_solution(["pumps but refills","debris"]):-
```

```
problem_tank(TANK),
write_screen(["Debris under the seat of the",
"void suction valve","allowed water to",
"back from the main drain system through the suction",
"valve into the void",TANK]).
```

```
write_solution(["pumps but refills","open ovbd"]):-
problem_tank(TANK),
write_screen(["The eductor overboard was closed",
"allowing fire main","pressure to leak back",
"through suction valve to void",TANK]).
```

```
write_solution(["pumps but refills","breaks"]):-
problem_tank(TANK),
write_screen(["The source of water refilling void",TANK,
"was a break in sounding tube or airescape piping",
"or a missing","sounding cap providing a path for",
"flooding from another source above."]).
```

```
write_solution(["pumps but refills","holes 4th deck"]):-
problem_tank(TANK),
above_tank(ABOVE),
write_screen(["The source of water refilling void",TANK,
"was a break in the deck, tank top cover,manhole",
"covering or piping penetrations in",ABOVE]).
```

```
write_solution(["pumps but refills","inboard void"]):-
problem_tank(TANK),
inbd_void(INBOARD),
write_screen(["The source of water refilling void",
TANK,"was a break in the void suction line in",
INBOARD]).
```

```
write_solution(["indeterminate data"]):-
write_screen(["The answers provided thus far are",
"not sufficient to","determine a cause, retrace",
"your answers through","the system again,and if",
"possible provide additional information."]).
```

```
*****
* PipesMain.prg is the main menu for the Pipes *
* database system *
*****
```

```
SET TALK OFF
CLEAR
SET STATUS OFF
SET BELL OFF
SET COLOR TO +G,BG/B,R,B
SelOpt = .T.
DO WHILE SelOpt
  Selopt = .F.
  SET FORMAT TO pipesmain
  Option = " "
  READ
  CLOSE FORMAT
  DO CASE
    CASE Option = " "
      EXIT
    CASE Option = "A"
      AddSel = .T.
      DO WHILE Addsel
        AddSel = .F.
        Add = " "
        SET FORMAT TO addscr
        READ
        CLOSE FORMAT
        DO CASE
          CASE Add = " "
            AddSel = .F.
          CASE Add = "A"
            do contedit
            Addsel = .T.
          CASE Add = "B"
            do compedit
          CASE Add = "C"
            do pipesedit
          CASE Add = "D"
            do adjedit
          OTHERWISE
            AddSel = .T.
        ENDCASE
      SelOpt = .T.
```

```

ENDDO
CASE Option = "Q"
    do pipequery
    SelOpt = .T.
CASE Option = "P"
    do pipeprint
    SelOpt = .T.
CASE Option = "B"
    CLEAR
    RUN pipesbk.bat
    Selopt = .T.
CASE Option = "R"
    CLEAR
    RUN pipesrs.bat
    SelOpt = .T.
CASE Option = "E"
    RUN SET BGIDIR=c:\dbase\bgi
    RUN Pipes
    SelOpt = .T.
OTHERWISE
    SelOpt = .T.
ENDCASE
ENDDO
SET STATUS ON RETURN

```

```

*****
* PipeQuery is the main query menu for the Pipes *
* database system *
*****
SET COLOR TO +G,BG/B,R,B
ON ESCAPE RETURN
Choice = .T.
DO WHILE Choice
    Choice = .F.
    SET FORMAT TO pipequery
    Option = " "
    READ
    CLOSE FORMAT
    DO CASE
        CASE Option = " "
            CHOICE = .F.
        CASE Option = "C"

```

```

do access
CHOICE = .T.
CASE Option = "P"
  ChkSel = .T.
  DO WHILE Chksel
    ChkSel = .F.
    PipeOpt = " "
    SET FORMAT TO pipesys
    READ
    CLOSE FORMAT
    DO CASE
      CASE PipeOpt = " "
        ChkSel = .F.
      CASE PipeOpt = "P"
        do passing
        Chksel = .T.
      CASE PipeOpt = "C"
        do pipecont
        Chksel = .T.
      CASE PipeOpt = "S"
        do specmat
        Chksel = .T.
      CASE PipeOpt = "M"
        do material
        Chksel = .T.
    OTHERWISE
      ChkSel = .T.
    ENDCASE
  Choice = .T.
  ENDDO
CASE Option = "A"
  do adjtank
  CHOICE = .T.
CASE Option = "L"
  do Lpaint
  CHOICE = .T.
CASE Option = "S"
  do strings
  CHOICE = .T.
CASE Option = "T"
  do TankType
  CHOICE = .T.
CASE Option = "I"

```

```

do inboard
CHOICE = .T.
CASE Option = "Q"
do custqry
CHOICE = .T.
OTHERWISE
Choice = .T.
ENDCASE
ENDDO
RETURN

```

```

*****
*   CompEdit.prg is used to add new records to the   *
*   database or modify records that already exists   *
*****

```

```

* UPPER(COMP_NUM) is the KEY field for the COMPARTMENT
* database

```

```

USE compartment INDEX compno
SET COLOR TO +G,BG/B,R,B
ON ESCAPE EXIT
Adding = .T.
DO WHILE Adding

```

```

CLEAR
CLOSE FORMAT
@ 3, 20 SAY "COMPARTMENT DATABASE UPDATE"
@ 22, 7 SAY "Press ESC to EXIT"
CompNo = SPACE(14)
@ 10,5 SAY "Enter the Compartment Number " GET CompNo;
FUNCTION "!"
READ

```

```

* Create a Search Variable
Search = UPPER(CompNo)

```

```

* RETURN if no input

```

```

IF Search = " "
Adding = .F.
LOOP
ENDIF

```

```

* Check database for compartment number

```

```

SEEK Search
SET FORMAT TO compscr  && open format file

* Edit if found

IF FOUND()
  READ
ENDIF

* ADD if not found

IF .NOT. FOUND()
  APPEND BLANK
  REPLACE Comp_Num WITH UPPER(CompNo)
  READ
ENDIF

ENDDO (while adding)
REINDEX
ERASE COMPARTM.TXT
COPY TO COMPARTM.FIELDS STRING,COMP_NUM TYPE DELIMITED
WITH
BLANK
CLOSE ALL
RETURN

*****
* AdjEdit.prg is used to add new records to the *
* database or modify records that already exists *
*****
* UPPER(COMP_NUM) is the KEY field for the COMPARTMENT
* database
USE adjacent INDEX adjcomp
SET COLOR TO +G,BG/B,R,B
ON ESCAPE EXIT
Adding = .T.
DO WHILE Adding
  CLEAR
  CLOSE FORMAT
  @ 3, 20 SAY "ADJACENT TANK DATABASE UPDATE"
  @ 22, 7 SAY "Press ESC to EXIT"
  CompNo = SPACE(14)
  @ 10,5 SAY "Enter the Compartment Number " GET CompNo;

```

```

FUNCTION "!"
READ

* Create a Search Variable
  Search = UPPER(CompNo)

* RETURN if no input

  IF Search = " "
    ADDING = .F.
    LOOP
  ENDIF

* Check database for compartment number

  SEEK Search
  SET FORMAT TO adjscr  && open format file

* Edit if found

  IF FOUND()
    READ
  ENDIF

* ADD if not found

  IF .NOT. FOUND()
    APPEND BLANK
    REPLACE Compt_Num WITH UPPER(CompNo)
    READ
    IF READKEY() <= 36
      DELETE
      PACK
    ENDIF
  ENDIF

ENDDO (while adding)
REINDEX
ERASE ADJACENT.TXT
COPY TO ADJACENT TYPE DELIMITED WITH BLANK
CLOSE ALL
RETURN

```



```

*****
* PipesEdit.prg is used to add new records to the
* database or modify records that already exists
*****
USE pipesyst INDEX pipesys,pipeiname
Adding = .T.
DO WHILE Adding
  CLEAR
  SET COLOR TO BG/B,N/G,R,R
  CLOSE FORMAT
  pipesys = SPACE(20)
  System = SPACE(35)
  SET FORMAT TO pipestart
  READ
  CLOSE FORMAT

* Create a Search Variable
  IF pipesys # " "
    Search = UPPER(pipesys)
    SET ORDER TO 1
  ELSE
    IF System # " "
      Search = UPPER(system)
      SET ORDER TO 2
    ELSE
      Search = " "
    ENDIF
  ENDIF

* RETURN if no input

  IF Search = " "
    Adding = .F.
    LOOP
  ENDIF

* Check database for compartment number
Looking = .T.

  SEEK Search
  SET FORMAT TO pipescr && open format file
  RK_QUIT = 12

```

* Edit if found

```
IF FOUND()
  SET COLOR TO +G,BG/B,R,B
  READ
  IF READKEY() = RK_QUIT
    Looking = .F.
  ENDIF
  DO WHILE Looking .AND. .NOT. EOF()
    SKIP
    IF SYSNUM = Search .or. Pipe_Sys = Search
      READ
      IF READKEY() = RK_QUIT
        Looking = .F.
      ENDIF
    ELSE
      Looking = .F.
    ENDIF
  ENDDO(while looking)
ENDIF
```

* ADD if not found

```
IF .NOT. FOUND() .AND. READKEY() # RK_QUIT
  APPEND BLANK
  IF pipesys # " "
    REPLACE SYSNUM WITH UPPER(pipesys)
  ENDIF
  IF System # " "
    REPLACE Pipe_Sys WITH UPPER(System)
  ENDIF
  READ
  IF Pipe_Sys = " " .OR. SYSNUM = " "
    DELETE
    PACK
  ENDIF
ENDIF
ENDDO (while adding)
REINDEX
CLOSE ALL
RETURN
```

```

*****
* ContEdit.prg is used to add new records to the *
* database or modify records that already exists *
*****
USE contains INDEX contcomp,contsys
Adding = .T.
DO WHILE Adding
  CLEAR
  SET COLOR TO BG/B,N/G,R,R
  CLOSE FORMAT
  CompNo = SPACE(14)
  System = SPACE(18)
  SET FORMAT TO contstart
  READ
  CLOSE FORMAT

* Create a Search Variable
IF CompNo # " "
  Search = UPPER(CompNo)
  SET ORDER TO 1
ELSE
  IF System # " "
    Search = UPPER(System)
    SET ORDER TO 2
  ELSE
    Search = " "
  ENDIF
ENDIF

* RETURN if no input

IF Search = " "
  Adding = .F.
  LOOP
ENDIF

* Check database for compartment number
Looking = .T.

SEEK Search
SET FORMAT TO contscr  && open format file
RK_QUIT = 12

```

* Edit if found

```
IF FOUND()
  SET COLOR TO +G,BG/B,R,B
  READ
  IF READKEY() = RK_QUIT
    Looking = .F.
  ENDIF
  DO WHILE Looking .AND. .NOT. EOF()
    SKIP
    IF COMPT_NUM = Search
      READ
      IF READKEY() = RK_QUIT
        Looking = .F.
      ENDIF
    ELSE
      Looking = .F.
    ENDIF
  ENDDO(while looking)
ENDIF
```

* ADD if not found

```
IF .NOT. FOUND() .AND. READKEY() # RK_QUIT
  APPEND BLANK
  IF CompNo # " "
    REPLACE Compt_Num WITH UPPER(CompNo)
  ENDIF
  IF System # " "
    REPLACE SysNum WITH UPPER(System)
  ENDIF
  READ
  IF SysNum = " " .OR. Compt_Num = " "
    DELETE
    PACK
  ENDIF
ENDIF
```

```
ENDDO (while adding)
REINDEX
ERASE CONTAINS.TXT
```

COPY TO CONTAINS TYPE DELIMITED WITH BLANK
CLOSE ALL
RETURN

```
*****
* Access.prg is used to query the adjacent database *
* for the above tank to get access to an eightdeck tank *
*****
USE adjacent INDEX adjcomp
SET COLOR TO +G,BG/B,R,B
ON ESCAPE EXIT
compno = SPACE(14)
SET FORMAT TO access
CLEAR
READ
CLOSE FORMAT
* Create a Search Variable
Search = UPPER(CompNo)

* RETURN if no input

IF Search = " "
    CLOSE ALL
    RETURN
ENDIF

* Check database for compartment number

SEEK Search
SET FORMAT TO adjprt
* Show if found

IF FOUND()
    READ
ELSE
    @ 10,16 CLEAR to 13,60
    @ 11,25 SAY "Not an eightdeck compartment"
    wait ""
ENDIF
CLOSE ALL
RETURN
```

```
*****
* TankType.prg is used to list compartments *
* by type *
*****
```

```
USE compartm
ON ESCAPE RETURN
Listing = .T.
DO WHILE Listing
  CLEAR
  SET COLOR TO BG/B,N/G,R,R
  CLOSE FORMAT
  Tank = SPACE(10)
  SET FORMAT TO tanktype
  READ
  CLOSE FORMAT
```

```
* Create a Search Variable
DO CASE
  CASE Tank = " "
    Listing = .F.
    LOOP
  CASE Tank = "S"
    Search = "FOS"
    ComName = " Fuel Oil Service"
  CASE Tank = "T"
    Search = "FOT"
    ComName = " Fuel Oil Storage"
  CASE Tank = "V"
    Search = "VOID"
    ComName = " Void"
  CASE Tank = "C"
    Search = "CONT"
    ComName = " Contaminated"
  CASE Tank = "J"
    Search = "JP-5"
    ComName = " JP-5"
  OTHERWISE
    @12, 21 SAY "Invalid selection"
    @13, 21 SAY "Press any key to continue"
    wait " "
    LOOP
ENDCASE
```

* Check database for pipe system number

```
Looking = .T.  
LOCATE FOR UPPER(USAGE) = Search  
ANS = " "  
SET FORMAT TO tankscr && open format file  
RK_QUIT = 12  
RK_PRINT = 2
```

* SHOW if found

```
IF FOUND()  
  STORE RECNO() TO saverec  
  STORE "USAGE" TO Sparam  
  STORE ComName+" Compartments " TO Title  
  STORE "comprt" TO prtfile  
  SET COLOR TO +W/B,BG/R,G,B  
  READ  
  IF READKEY() = RK_QUIT  
    Looking = .F.  
  ENDIF  
  IF READKEY() = RK_print  
    do fprint  
    Looking = .F.  
  ENDIF  
  
  DO WHILE Looking .AND. .NOT. EOF()  
    CONTINUE  
    IF FOUND()  
      READ  
      IF READKEY() = RK_QUIT  
        Looking = .F.  
      ENDIF  
      IF READKEY() = RK_PRINT  
        do fprint  
        Looking = .F.  
      ENDIF  
    ELSE  
      Looking = .F.  
    ENDIF  
  ENDDO(while looking)  
ENDIF  
ENDDO
```

```

SET COLOR TO +G,BG/B,R,B
CLOSE ALL
RETURN

```

```

*****
*   Passing.prg is used to find pipesystems passing           *
*   through a speified compartment                           *
*****

```

```

USE contains INDEX contcomp
ON ESCAPE RETURN
NoExit = .T.
DO WHILE NoExit
  CLEAR
  SET COLOR TO BG/B,N/G,R,R
  CLOSE FORMAT
  CompNo = SPACE(14)
  SET FORMAT TO PasStart
  READ
  CLOSE FORMAT

```

```

* Create a Search Variable
IF CompNo # " "
  Search = UPPER(CompNo)
ELSE
  Search = " "
ENDIF

```

```

* RETURN if no input

```

```

IF Search = " "
  NoExit = .F.
  LOOP
ENDIF

```

```

* Check database for compartment number
Looking = .T.

```

```

SEEK Search
ANS = " "
SET FORMAT TO passscr  && open format file
RK_QUIT = 12
RK_Print = 2

```


* SHOW if found

IF FOUND()

STORE RECNO() TO saverec

STORE "Compt_Num" TO Sparam

STORE "Pipes Passing Through Compartment "+Compt_Num;

TO Title

STORE "sysprt" TO prtfile

SET COLOR TO +W/B,BG/R,G,B

READ

IF READKEY() = RK_QUIT

Looking = .F.

ENDIF

IF READKEY() = RK_print

do fprint

Looking = .F.

ENDIF

DO WHILE Looking .AND. .NOT. EOF()

SKIP

IF COMPT_NUM = Search

READ

IF READKEY() = RK_QUIT

Looking = .F.

ENDIF

IF READKEY() = RK_print

do fprint

Looking = .F.

ENDIF

ELSE

Looking = .F.

ENDIF

ENDDO(while looking)

ELSE

@ 10,16 CLEAR to 11,60

@ 9,25 SAY "Not a valid compartment"

@ 11,17 SAY "Press "+CHR(17)+CHR(196)+CHR(217)+" to
continue"

wait ""

ENDIF

ENDDO

SET COLOR TO +G,BG/B,R,B

CLOSE ALL

RETURN

```
*****
* Material.prg is used to find the pipe systems      *
* made of a specific material                        *
*****
```

```
USE pipesyst INDEX pipemat
SET COLOR TO +G,BG/B,R,B
ON ESCAPE RETURN
NoExit = .T.
DO WHILE NoExit
  CLEAR
  SET COLOR TO BG/B,N/G,R,R
  CLOSE FORMAT
  pipemat = SPACE(10)
  SET FORMAT TO Pipemat
  READ
  CLOSE FORMAT
```

```
* Create a Search Variable
IF pipemat # " "
  Search = UPPER(pipemat)
ELSE
  Search = " "
ENDIF
```

```
* RETURN if no input
IF Search = " "
  NoExit = .F.
  LOOP
ENDIF
```

```
* Check database for pipe system number
Looking = .T.
```

```
SEEK Search
ANS = " "
SET FORMAT TO lspecmat  && open format file
RK_QUIT = 12
RK_PRINT = 2
```

```
* SHOW if found
```

```

IF FOUND()
  STORE RECNO() TO saverec
  STORE "Material" TO Sparam
  STORE "List of pipe systems that are made of "+pipemat;
    TO Title
  STORE "sysprt" TO prtfile
  SET COLOR TO +W/B,BG/R,G,B
  READ
  IF READKEY() = RK_QUIT
    Looking = .F.
  ENDIF
  IF READKEY() = RK_print
    do fprint
    Looking = .F.
  ENDIF

  DO WHILE Looking .AND. .NOT. EOF()
    SKIP
    IF Upper(Material) = Search
      READ
      IF READKEY() = RK_QUIT
        Looking = .F.
      ENDIF
      IF READKEY() = RK_PRINT
        do fprint
        Looking = .F.
      ENDIF
    ENDIF
  ENDDO(while looking)
ELSE
  @ 12,16 CLEAR to 11,58
  @ 9,25 SAY "No pipes of that material"
  @ 11,17 SAY "Press "+CHR(17)+CHR(196)+CHR(217)+" to
continue"
  wait ""
ENDIF
ENDDO
SET COLOR TO +G,BG/B,R,B
CLOSE ALL
RETURN

```

```

*****
* Adjtank.prg is used to query the adjacent database *
* for the tanks surrounding the specified tank *
*****
USE adjacent INDEX adjcomp
SET COLOR TO +G,BG/B,R,B
ON ESCAPE EXIT
AdjCont = .T.
DO WHILE AdjCont
    compno = SPACE(14)
    SET FORMAT TO passtart
    CLEAR
    READ
    CLOSE FORMAT
* Create a Search Variable
    Search = UPPER(CompNo)

* RETURN if no input

    IF Search = " "
        AdjCont = .F.
        LOOP
    ENDIF

* Check database for compartment number

    SEEK Search
    RK_QUIT = 12
    RK_PRINT = 2

    ANS = " "
    SET FORMAT TO adjtank
* Show if found

    IF FOUND()
        READ
        IF READKEY() = RK_QUIT
            AdjCont = .F.
        ENDIF
        IF READKEY() = RK_PRINT
            STORE RECNO() TO saverec
            STORE "Compt_Num" TO Sparam
            STORE "Compartments adjacent to "+Compt_Num;

```

```

    TO Title
    STORE "adjprt" TO prtfile
    do fprint
    ENDIF
ELSE
    @ 10,16 CLEAR to 11,60
    @ 9,25 SAY "Not a valid compartment"
    @ 11,17 SAY "Press "+CHR(17)+CHR(196)+CHR(217)+" to
continue"
    wait ""
    ENDIF
ENDDO
CLOSE ALL
RETURN

```

```

*****
* InbdTank.prg is used to find inboard compartments *
*****

```

```

USE compartm INDEX compno
ON ESCAPE RETURN
NoExit = .T.
DO WHILE NoExit
    CLEAR
    SET COLOR TO BG/B,N/G,R,R
    CLOSE FORMAT
    Indb = " "
    SET FORMAT TO inbdtank
    READ
    CLOSE FORMAT

```

```

* Create a Search Variable
DO CASE
    CASE inbd = " "
        NoExit = .F.
        LOOP
    CASE inbd = "T"
        do findinbd
    CASE Tank = "V"
        do voidinbd
    OTHERWISE
        @12, 12 SAY "Invalid selection"
        @13, 12 SAY "Press any key to continue"
        wait " "

```

```

        LOOP
    ENDCASE
ENDDO
SET COLOR TO +G,BG/B,R,B
CLOSE ALL
RETURN

```

```

*****
*   Findinbd.prg is used to find the inboard   *
*   tank of a specified compartment           *
*****

```

```

ON ESCAPE RETURN
NoExit = .T.
DO WHILE NoExit
    CLEAR
    SET COLOR TO BG/B,N/G,R,R
    CLOSE FORMAT
    CompNo = SPACE(14)
    SET FORMAT TO PasStart
    READ
    CLOSE FORMAT

```

```

* Create a Search Variable
IF CompNo # " "
    Search = UPPER(CompNo)
ELSE
    Search = " "
ENDIF

```

```

* RETURN if no input

```

```

    IF Search = " "
        NoExit = .F.
    LOOP
ENDIF

```

```

* Check database for compartment number
Looking = .T.

```

```

    SEEK Search
    ANS = " "
    SET FORMAT TO findinbd  && open format file
    RK_QUIT = 12

```

```

RK_Print = 2

* SHOW if found

IF FOUND()
  Strno = STRING
  Strlen = LEN(Strno)
  chkstr= SUBSTR(Strno,strlen,1)
  DO CASE
    CASE UPPER(chkstr) = "S"
      side = "PORT"
    CASE UPPER(chkstr) = "P"
      side = "STBD"
    OTHERWISE
      @ 21,12 SAY "ERROR"
  ENDCASE
  Use Adjacent INDEX adjcomp
  SEEK Search
  IF FOUND()
    SET COLOR TO +W/B,BG/R,G,B
    READ
    IF READKEY() = RK_QUIT
      Looking = .F.
    ENDIF
  ELSE
    @ 10,16 CLEAR to 11,60
    @ 9,25 SAY "Not a valid compartment"
    @ 11,17 SAY "Press "+CHR(17)+CHR(196)+CHR(217)+" to
continue"
    wait ""
  ENDIF
ENDDO
SET COLOR TO +G,BG/B,R,B
CLOSE ALL
RETURN

*****
* SpecMat.prg is used to find the material a *
* specific pipe system is made from *
*****

USE pipesyst INDEX pipesys
ON ESCAPE RETURN
Looking = .T.

```

```

DO WHILE Looking
  SET COLOR TO +G,BG/B,R,B
  SET FORMAT TO PasSys
  CLEAR
  System = SPACE(18)
  READ
  CLOSE FORMAT

```

```

* Create a Search Variable
  Search = UPPER(System)

```

```

* RETURN if no input

```

```

  IF Search = " "
    Looking = .F.
  LOOP
ENDIF

```

```

* Check database for pipe system

```

```

  SEEK Search
  RK_QUIT = 12

```

```

* Show if found

```

```

  SET FORMAT TO specmat && open format file
  IF FOUND()
    ANS = " "
    SET COLOR TO +W/B,BG/R,G,B
    READ
    IF READKEY() = RK_QUIT
      Looking = .F.
    ENDIF
  ELSE
    @ 10,18 CLEAR to 11,60
    @ 10,27 SAY "Not a valid pipe system"
    wait " "
  ENDIF
ENDDO
SET COLOR TO +G,BG/B,R,B
CLOSE ALL
RETURN

```



```

*****
* PipeCont.prg is used to find compartment *
* through which a speified pipe system passes *
*****
USE contains INDEX contsys
ON ESCAPE RETURN
NoExit = .T.
DO WHILE NoExit
  CLEAR
  SET COLOR TO BG/B,N/G,R,R
  CLOSE FORMAT
  System = SPACE(18)
  ANS = " "
  SET FORMAT TO PasSys
  READ
  CLOSE FORMAT

* Create a Search Variable
IF System # " "
  Search = UPPER(System)
ELSE
  Search = " "
ENDIF

* RETURN if no input

IF Search = " "
  NoExit = .F.
  LOOP
ENDIF

* Check database for pipe system number
Looking = .T.

SEEK Search
ANS = " "
SET FORMAT TO pcontscr && open format file
RK_QUIT = 12
RK_PRINT = 2

* SHOW if found

IF FOUND()

```

```

STORE RECNO() TO saverec
STORE "SYSNUM" TO Sparam
STORE "Compartments Containing "+SYSNUM TO Title
STORE "contprt" TO prtfile
SET COLOR TO +W/B,BG/R,G,B
READ
IF READKEY() = RK_QUIT
    Looking = .F.
ENDIF
IF READKEY() = RK_print
    do fprint
    Looking = .F.
ENDIF

DO WHILE Looking .AND. .NOT. EOF()
    SKIP
    IF SysNum = Search
        READ
        IF READKEY() = RK_QUIT
            Looking = .F.
        ENDIF
        IF READKEY() = RK_PRINT
            do fprint
            Looking = .F.
        ENDIF
    ELSE
        Looking = .F.
    ENDIF
    ENDDO(while looking)
ELSE
    @ 10,16 CLEAR to 11,60
    @ 9,25 SAY "Not a valid pipe system"
    @ 11,17 SAY "Press "+CHR(17)+CHR(196)+CHR(217)+" to
continue"
    wait ""
ENDIF
ENDDO
SET COLOR TO +G,BG/B,R,B
CLOSE ALL
RETURN

```

```

*****
* Strings.prg is used to find the compartments
* in a speified string
*****

USE compartment
ON ESCAPE RETURN
NoExit = .T.
DO WHILE NoExit
  CLEAR
  SET COLOR TO BG/B,N/G,R,R
  CLOSE FORMAT
  StrNo = SPACE(10)
  SET FORMAT TO String
  READ
  CLOSE FORMAT

* Create a Search Variable
  IF StrNo # " "
    Search = UPPER(StrNo)
  ELSE
    Search = " "
  ENDIF

* RETURN if no input

  IF Search = " "
    NoExit = .F.
    LOOP
  ENDIF

* Check database for string number
Looking = .T.

  LOCATE FOR STRING = Search
  ANS = " "
  SET FORMAT TO strscr  && open format file
  RK_QUIT = 12
  RK_Print = 2

* SHOW if found

  IF FOUND()
    STORE RECNO() TO saverec

```

```

STORE "STRING" TO Sparam
STORE "Compartments in STRING "+STRING;
TO Title
STORE "comprt" TO prtfile
SET COLOR TO +W/B,BG/R,G,B
READ
IF READKEY() = RK_QUIT
    Looking = .F.
ENDIF
IF READKEY() = RK_print
    do fprint
    Looking = .F.
ENDIF

DO WHILE Looking .AND. .NOT. EOF()
    CONTINUE
    IF FOUND()
        READ
        IF READKEY() = RK_QUIT
            Looking = .F.
        ENDIF
        IF READKEY() = RK_print
            do fprint
            Looking = .F.
        ENDIF
    ELSE
        Looking = .F.
    ENDIF
    ENDDO(while looking)
ELSE
    @ 10,16 CLEAR to 11,60
    @ 9,29 SAY "Not a valid string"
    @ 11,17 SAY "Press "+CHR(17)+CHR(196)+CHR(217)+" to
continue"
    wait ""
ENDIF
ENDDO
SET COLOR TO +G,BG/B,R,B
CLOSE ALL
RETURN

```

```

*****
* Stradj.prg is used to find the compartments *
* surrounding a specific string *
*****

USE compartment
ON ESCAPE RETURN
NoExit = .T.
DO WHILE NoExit
  CLEAR
  SET COLOR TO BG/B,N/G,R,R
  CLOSE FORMAT
  StrNo = SPACE(10)
  SET FORMAT TO String
  READ
  CLOSE FORMAT

  * Create a Search Variable
  IF StrNo # " "
    Search = UPPER(StrNo)
  ELSE
    Search = " "
  ENDIF

  * RETURN if no input

  IF Search = " "
    NoExit = .F.
    LOOP
  ENDIF

  * Check database for string number
  Looking = .T.

  LOCATE FOR STRING = Search
  ANS = " "
  SET FORMAT TO stradj && open format file
  RK_QUIT = 12
  RK_Print = 2

  * SHOW if found

  IF FOUND()
    SRSTR = UPPER(COMP_NUM)

```

```

USE ADJACENT INDEX adjcomp
SEEK SRSTR
IF FOUND()
    tankport = UPPER(PORT)
    tankstbd = UPPER(STBD)
CLOSE DATABASES
strsearch = tankport
USE COMPARTMENT INDEX compno
SEEK strsearch
IF FOUND()
    PortStr = STRING
ELSE
    PortStr = " "
ENDIF
strsearch = tankstbd
GO TOP
SEEK StrSearch
IF FOUND()
    StbdStr = STRING
ELSE
    StbdStr = " "
ENDIF
IF PortStr # " " .AND. StbdStr # " "
    Search = PortStr .OR. StbdStr
ELSE
    IF PortStr # " "
        Search = PortStr
    ELSE
        IF StbdStr # " "
            Search = StbdStr
        ELSE
            Search = " "
        ENDIF
    ENDIF
ENDIF
IF SEARCH # " "
    GO TOP
    LOCATE FOR STRING = Search
    IF FOUND()
        STORE RECNO() TO saverec
        STORE "COMPT_NUM" TO Sparam
        STORE "Compartments Surrounding STRING "+StrNo;
        TO Title
    
```

```

STORE "strprt" TO prtfile
SET COLOR TO +W/B,BG/R,G,B
READ
IF READKEY() = RK_QUIT
    Looking = .F.
ENDIF
IF READKEY() = RK_print
    do fprint
    Looking = .F.
ENDIF
DO WHILE Looking .AND. .NOT. EOF()
    CONTINUE
    IF FOUND()
        READ
        IF READKEY() = RK_QUIT
            Looking = .F.
        ENDIF
        IF READKEY() = RK_print
            do fprint
            Looking = .F.
        ENDIF
    ELSE
        Looking = .F.
    ENDIF
ENDDO(while looking)
ENDIF
ELSE
    @ 10,16 CLEAR TO 11,60
    @ 9,25 SAY "Invalid data in Search"
    @ 11,17 SAY "Press "+CHR(17)+CHR(196)+CHR(217)+" ;
        to continue"
ENDIF
ENDIF
ELSE
    @ 10,16 CLEAR to 11,60
    @ 9,29 SAY "Not a valid string"
    @ 11,17 SAY "Press "+CHR(17)+CHR(196)+CHR(217)+" to
continue"
    wait ""
ENDIF
ENDDO
SET COLOR TO +G,BG/B,R,B
CLOSE ALL

```

```

RETURN
*****
* Fprint.prg is used to print queries *
*****
SET CONSOLE OFF
SET MARGIN TO 4
Linect = 1
pagect = 1
pageln = 55
GOTO Saverec
SET PRINT ON
CLEAR
@ 23,10 SAY "Press any key to stop printing..."
ON KEY DO Interrupt
* Print header
  ? SPACE((80-LEN(TITLE))* .5)+Title
  ?
  ?
  Linect = 3

*Print Query Info

DO WHILE .NOT. EOF()
  IF &Sparam = Search
    do &prtfile
  ELSE
    SET PRINT OFF
    EJECT
    SET CONSOLE ON
    RETURN
  ENDIF

* See if a page break is needed

  IF Linect >= Pageln
    EJECT
    Pagect = Pagect + 1
    ? SPACE((80-LEN(TITLE))* .5)+Title
    ?
    ?
    Linect = 3
  ENDIF
  SKIP

```



```
ENDDO
SET CONSOLE ON
ON KEY
SET PRINT OFF
EJECT
RETURN
```

```
*****
* Pipeprint.prg allows user to build a report form *
* and send it to the screen or printer *
*****
```

```
SET SAFETY OFF
CLEAR
Dir *.DBF
DbFile = SPACE(8)
@ 22,2 SAY "Enter Database Filename: " GET DbFile
READ
USE &DbFile
CLEAR
DIR *.frm
Rfile = SPACE(8)
Printer = "N"
@ 22,2 SAY "Enter a new filename for the report form"
@ 23,2 SAY "or reuse an existing form from above: "
@ 23,41 GET Rfile
READ
```

```
* Build REPORT FORM
```

```
IF Rfile # " "
    MODIFY REPORT &Rfile
```

```
* PRINT
```

```
CLEAR
STORE " " TO Printer, PMacro
@ 15,5 SAY "Send data to printer? (Y/N) " GET Printer
PICT "!"
READ
IF Printer = "Y"
    PMacro = "TO PRINT"
    WAIT "Prepare printer, then press any key to continue..."
```

```

ENDIF
REPORT FORM &Rfile &PMacro
ENDIF
IF Printer = "Y"
EJECT
ENDIF
SET FILTER TO
CLOSE ALL
WAIT "Press any key to return"
RETURN
*****
* Passprt.prg is Fields to be printed for query pipes *
* passing through a compartment *
*****
?SPACE(5)+SYSNUM
Linect = Linect + 1

*****
* Strprt.prg is Fields to be printed for query *
* compartments in specific string *
*****
?SPACE(5)+COMP_NUM
Linect = Linect + 1

*****
* Adjprt.prg is Fields to be printed for query *
* compartments adjacent to specified compartment *
*****
?SPACE(5)+FWD
?SPACE(5)+AFT
?SPACE(5)+STBD
?SPACE(5)+PORT
?
Linect = Linect + 5

*****
* AddrQry.prg lets user build a custom query form or use *
* an existing form. Also allows user to direct data to *
* printer *
*****
SET SAFETY OFF
CLEAR
Dir *.DBF

```

```

DbFile = SPACE(8)
@ 22,2 SAY "Enter Database Filename: " GET DbFile
READ
USE &DbFile
CLEAR
DIR *.QRY
Qfile = SPACE(8)
@ 22,2 SAY "Enter a new filename for the query form"
@ 23,2 SAY "or reuse an existing form from above: "
@ 23,41 GET Qfile
READ
IF Qfile # " "
    MODIFY QUERY &Qfile
    GO TOP
    IF EOF()
        CLEAR
        ? "Warning... no records match search criterion!"
        ?
        WAIT
    ENDIF(EOF)
ENDIF(Qfile)
CLEAR
DIR *.frm
Rfile = SPACE(8)
@ 22,2 SAY "Enter a new filename for the report form"
@ 23,2 SAY "or reuse an existing form from above: "
@ 23,41 GET Rfile
READ

* Build REPORT FORM

IF Rfile # " "
    MODIFY REPORT &Rfile

* PRINT

CLEAR
STORE " " TO Printer, PMacro
@ 15,5 SAY "Send data to printer? (Y/N) " GET Printer
PICT "!"
READ
IF Printer = "Y"
    PMacro = "TO PRINT"

```

```

    WAIT "Prepare printer, then press any key to continue..."
ENDIF
    REPORT FORM &Rfile &PMacro
ENDIF
IF Printer = "Y"
    EJECT
ENDIF
SET FILTER TO
CLOSE ALL
Wait "Press any key to continue"
RETURN

```

```

*****
* CustQry.prg lets user build a custom query form or use          *
* an existing form. Also allows user to build a                  *
* report form and send it to the screen or printer              *
*****

```

```

SET SAFETY OFF
CLEAR
Dir *.DBF
DbFile = SPACE(8)
Printer = "N"
@ 22,2 SAY "Enter Database Filename: " GET DbFile
READ
USE &DbFile
CLEAR
MODIFY QUERY pipes
GO TOP
IF EOF()
    CLEAR
    ? "Warning... no records match search criterion!"
    ?
    WAIT
ENDIF(EOF)
CLEAR
DIR *.frm
Rfile = SPACE(8)
@ 22,2 SAY "Enter a new filename for the report form"
@ 23,2 SAY "or reuse an existing form from above: "
@ 23,41 GET Rfile
READ

```

```

* Build REPORT FORM

```

```

IF Rfile # " "
  MODIFY REPORT &Rfile

* PRINT

CLEAR
STORE " " TO Printer, PMacro
@ 15,5 SAY "Send data to printer? (Y/N) " GET Printer
PICT "!"
READ
IF Printer = "Y"
  PMacro = "TO PRINT"
  WAIT "Prepare printer, then press any key to continue..."
ENDIF
REPORT FORM &Rfile &PMacro
ENDIF
IF Printer = "Y"
  EJECT
ENDIF
SET FILTER TO
ERASE Pipes.qry
CLOSE ALL
WAIT "Press any key to return"
RETURN

```

```

*****
* PipesBk is the floppy disk backup for the Pipes *
* database and expert system *
*****

```

```

CLEAR
RUN pipesbk.dat
RETURN

```

```

*****
* PipesRs is the floppy disk restore for the Pipes *
* database and expert system *
*****

```

```

CLEAR
@ 10,12 SAY "Place disk containing files in Drive A and
Close door"
?
WAIT "      Press any Key to continue"

```

RUN pipesrs.dat
RETURN

-

LIST OF REFERENCES

1. *Carrier Life Enhancing Repairs (CLER) Program Engineered Maintenance Plan (Tank and Void Zoning, Tank Top, Piping, and Section)*, Naval Sea Systems Command Detachment PERA (CV), Bremerton, Washington, October 1987.
2. Rowe, N. C., *Introduction to Artificial Intelligence through Prolog*, Prentice-Hall, 1987.
3. Borland International, *TURBO PROLOG, Version 2.0, Reference Guide*, 1988.
4. Ashton-Tate, *d-BASE III PLUS, Version 3.0, Reference Manual*, 1986.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center.....2
 Cameron Station
 Alexandria, Virginia 22304-6145

2. Library, Code 0142.....2
 Naval Postgraduate School
 Monterey, California 93943-5002

3. Curriculum Officer.....1
 Code 37
 Computer Technology
 Naval Postgraduate School
 Monterey, California 93943-5000

4. CDR I. B. Clayton.....2
 COMNAVAIRPAC Code 73
 NAS North Island, California 92135

5. LT P. R. Boozer.....2
 128 Brody Road
 Chapin, South Carolina 29036

6. Associate Professor C. T. Wu1
 Code 52Wu
 Department of Computer Science
 Naval Postgraduate School
 Monterey, California 93943-5000