SYSTEM RELIABILITY PROGRAMS FOR VAX COMPUTER SYSTEM: INTERACTIVE SUM OF DISJOINT PRODUCTS (INSDP) & INTERACTIVE SIMPLIFIED TOPOLOGICAL RELIABILITY ANALYSIS PROGRAM (INSTRAP)

BY

MAUNG M. LAY
Bachelor of Science
Iowa State University
Ames, Iowa
1980

Master of Science
Oklahoma State University
Stillwater, Oklahoma
1982

Submitted to the Graduate Faculty of the
Department of Management
College of Business Administration
Oklahoma State University
in partial fulfillment of
the requirements for the Degree of
MASTER OF BUSINESS ADMINISTRATION
May, 1986

Name: Maung M. Lay (aka Clifford Shoung)

Institution: Oklahoma State University

Location: Stillwater, Oklahoma

Title of Study: SYSTEM RELIABILITY PROGRAMS FOR VAX COMPUTER SYSTEM: INTERACTIVE SUM OF DISJOINT PRODUCTS (INSDP) & INTERACTIVE SIMPLIFIED TOPOLOGICAL RELIABILITY ANALYSIS PROGRAM (INSTRAP)

Pages in Study: 89

Candidate for Degree of Master of Business Administration

Major Field: Business Administration

Scope and Method of Study: This study converted two batch-oriented PL/I system reliability programs- Sum of Disjoint Products (SDP) and Simplified Topological Reliability Analysis Program (STRAP), into interactive menu-driven programs on the VAX computer system. The interactive versions of the programs are INSDP and INSTRAP, respectively. The underlying principles of SDP and topological reliability are discussed briefly. The emphasis is to show how INSDP and INSTRAP can be used. Examples are provided to enhance the learning process of these programs.

Findings and Conclusions: The INSDP and INSTRAP were used by several graduate students for class assignments here at OSU. The programs' ease of use and the interactive features were found to be satisfactory. Eventually, these programs will be implemented on the microcomputers, and this study has provided the foundation for such purpose.

ADVISOR'S APPROVAL ______________________________

SYSTEM RELIABILITY PROGRAMS FOR VAX COMPUTER SYSTEM:
INTERACTIVE SUM OF DISJOINT PRODUCTS (INSDP)
&
INTERACTIVE SIMPLIFIED TOPOLOGICAL RELIABILITY
ANALYSIS PROGRAM (INSTRAP)

Report Approved:

____
Advisor

____
Director of Graduate Studies

____
Head, Department of Management

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to Dr. Mitchell Locks for his intellectual guidance and assistance throughout this study. I also like to thank Dr. Lee Manzer, the immediate past MBA director, for his assistance in many areas during my stay at the Business College.

Finally, I wish to give my deepest gratitude to my parents, sisters, brothers, and dear friends at Oklahoma State University for their constant support over the years.

TABLE OF CONTENTS

1.0 INTRODUCTION

The purpose of this paper is to convert two PL/I batch-oriented system reliability programs- Sum of Disjoint Products (SDP) and Simplified Topological Reliability Analysis Program (STRAP), into user-friendly menu-driven programs on VAX computer system. The interactive versions of these programs are called INSDP and INSTRAP, respectively.

There is no attempt to modify the original SDP and STRAP programs. However, a shell-sort routine is incorporated into the INSDP. Besides this, the interactive routines are built around the original programs.

In writing these interactive reliability programs, the author has attempted to achieve the following objectives:

(1) the program should be self-contained, that is - when using the program, the user must be able to use it with minimal reference to outside source.

(2) data can be entered either by keyboard or by system file.

(3) appropriate messages are displayed when user has made the errors.

(4) output of the programs are available both in hard copy version and screen output version.

In the opinion of the author, these objectives have been achieved in both INSDP and INSTRAP programs. However, only the repeated use of these programs by the users can prove likewise.

The author also made the assumption that the users of INSDP and INSTRAP are familiar with the VAX computer system. Hence, no specific instructions about the VAX computer system are given here.

The organization of this paper is as follows. In Chapter 2, the theoritical aspect of the sum of disjoint product is discussed briefly. The original SDP computer program and its composition are described next. Finally, the INSDP and its features are presented.

In Chapter 3, as in the case of SDP, the fundamentals of topological reliability are discussed. Then the original STRAP computer program is described briefly. Lastly, the INSTRAP and its attributes are presented.

In Chapter 4, the procedure in accessing the INSDP and INSTRAP via telecommunication device is described briefly. This chapter is included so the users that are not on OSU campus may access to these reliability programs.

Appendices A and C are the program listings of INSDP and INSTRAP, respectively. In addition, to facilitate the user in familiarizing with INSDP and INSTRAP, Appendices B and D are provided to illustrate the inputing procedures of these programs. Appendix E provides the additional informations on accessing the OSU Computer Center via communication network.

2.0 **SUM OF DISJOINT PRODUCTS (SDP)**

Abraham [1] algorithm (AA) is probably the best known disjoint method for evaluating the system reliability. In this algorithm, given the minimal paths or the minimal cuts, the system reliability function is built up by decomposing the Boolean polynomial into disjoint terms one path at a time. The reliability of the system is then the summation of probabilities of all these terms.

Prior to AA, Fretta and Montanari [2] illustrated how Boolean algebra could be used to find the sum of products for each of the simple paths between the pair of nodes. The system reliability is obtained by changing this into a sum of disjoint (mutually exclusive) products. Aggarwal et. al. [3] also used similar idea to find the disjoint sum. However, they used non-Boolean algebra to accomplish the task.

2.1 ABRAHAM ALGORITHM

For this algorithm, the system reliability network must be broken down into minimal paths or minimal cuts. The success and failure of the components of the minimal path may be represented by 1 and 0, respectively. A coherent system contains either success or failure components, while a noncoherent system may consist of both states simultaneously. The original AA only deals with coherent system.

The basic principle behind the AA is that if two or more events have no components in common, the probability that at least one of them will occur is the sum of the probability of the separate sets. The task is to make all the minimal paths disjoint from one another and add them together afterward.

To make the minimal paths disjoint from one another, AA uses two procedures to carry out the task. Locks [4] termed them as 'outer loop' and 'inner loop'. These two loops mainly invert and reinvert the values of the components from previous steps so that the paths are disjoint.

2.1.1 The outer loop

The major function of the procedure is to assure that the very first minimal paths, say path A, is disjoint from all other original paths (B,C,D,...,Z) of the given system network. If path A is not disjoint fom the second path B, the immediate terms B1, B2,..., may be created by altering the values of the components. Zeroes and ones may be placed in the appropriate positions so these immediate terms are disjoint from path A. The disjoint terms B1,B2,..., are then placed in a 'disjoint bank' that we will call Group 1.

Next original path C is checked to see whether it is disjoint from path A. Similar steps are taken to make paths C and A disjoint from one another. Immediate terms of path C may be created. However, these immediate terms will not be stored in Group 1. Because Group 1 only keeps the terms that are completely disjoint from each other. For path C, being disjoint from A does not guarantee that it is disjoint from other terms within the

Group 1. Hence, inner loop procedure is called to carry out such task.

2.1.2 The inner loop

The inner loop's function is to guarantee that at the given outer loop after path B, say C, the immediate terms are disjoint from the terms in Group 1. Again, the values of the components of path C, or its immediate terms are altered so they all are disjoint from Group 1. When these immediate terms or path C is disjoint from all the members of Group 1, they are then become the members themselves. It is important to note that only those terms that are not in Group 1 will be altered. Once the terms are stored in the Group 1, the values of the components will not be changed or altered.

Both outer loop and inner loop will proceed until all the original paths have been compared against the path A, and all the immediate terms are disjoint from one another. It is interesting to note that those original paths that could not be disjoint from path A are dropped from the algorithm for further consideration.

2.2 SDP COMPUTER PROGRAM

Chao [5] wrote a batch oriented PL/I SDP computer program to be run on IBM mainframe. Her SDP program is based on the earlier works of J.B. Keats [6]. Chao's SDP program provides the users with the following results: (1) echo print of input data (2) equivalent set of disjoint terms (3) the system reliability and (4) the importance of each component.

The SDP computer program is based on the following principle and it is described in relative details here. In a minimal state, a component is 1-valued for success and 0-valued for failure. If a component is free, it is denoted as a dash (-), which means it can be either 1-valued or 0-valued.

Minimal terms A1 and A2 are said to be disjoint if any component in A1 is 1 while the corresponding component in A2 is 0. For example,

A1= 1 - - 1
A2= 1 - 1 0

If A1 and A2 are not disjoint, it is possible to make them disjoint by searching for the corresponding positions where A1 is 1 (or 0) and A2 is -, but not the reverse condition. If only one 1 - (or 0 -) is found, then the - in A2 is replaced with a 0 (or 1). If there are two 1 - (or 0 -) combinations, then A2 is converted into two terms with 0 - and 1 0 (or 1 - and 0 1, respectively) in the two positions where - reside. For example,

A1= 1 - - - 1
A2= - 1 - 1 -

A2 is converted into

Y1= 0 1 - 1 -
Y2= 1 1 - 1 0

If there are more than two 1 - (or 0 -) combinations, say K, then K terms will be created at the outer loop. These positions are assigned with L(1),L(2),L(3),..., and L(K). The first disjoint term is obtained when replacing - in L(2) with 0 (or 1). At the same time, the component in L(1) position of this second term is set to 1 (Or 0) so the second term is disjoint from the first term. This procedure guarantees that the newly generated term will be disjoint with one another and with the predecessors. For example,

```
A1= - 0 - - 1 0 1 1 0 1
A2= 1 0 1 1 - 0 - - 0 -
```

A2 is generated into K(4) terms,

```
Y1= 1 0 1 1 0 1 - - 1 -
Y2= 1 0 1 1 1 1 0 - 1 -
Y3= 1 0 1 1 1 1 1 0 1 -
Y4= 1 0 1 1 1 1 1 1 1 0
```

2.2.1 Program composition

The program contains an external procedure CMPR, an internal block CMPR1 and another external procedure NEW. Procedure CMPR is used to compare the current minimal state to the preceeding disjoint products and see whether they are disjoint or not. If not, procedure CMPR1 is used to search for the 1 - (or 0 -) positions. When the positions are formed, procedure NEW is used to generate new terms which are, disjoint with one another and the preceeding terms. The program continues until all the potential disjoint products are obtained. An external procedure RELY is used at the end of the program to calculate the system reliability by adding the probability of success of all the disjoint products. In addition, another external procedure IMP is used to provide the order of importance for the components. The importance of a component is the partial derivative of the system probability with respect to the probability of the component [4], [5].

2.3 INTERACTIVE SDP (INSDP)

The INSDP is designed to run on VAX computer system. It is an user-friendly menu-driven program. The INSDP contains various error-checking routines, hence the program will not †bomb' easily. The computer listing of INSDP with documentation is shown in Appendix A.

2.3.1 To run INSDP

Once the INSDP is properly set up in VAX computer system (i.e. type in the whole program listed in Appendix A and stored under the file name-- INSDP), the user may type as follows,

$ **RUN INSDP**

Of course, it is assumed here that the program has already been compiled and linked correctly before the program execution.

2.3.2 Interactive data entry mode

The user is allowed to enter the input data either through keyboard or from the VAX data file. If the interactive data entry mode is selected, user must enter the following items: (1) the number of minimal states -N (2) the number of components in the system -M (3) the set of probability of success for each component -P and (4) the set of minimal paths or minimal cuts -S. The program also allows the user to change any input before the execution. Since the program is user-friendly, it tries to catch any errors during data entry. For example, error message is shown if the probability value entered is less than zero or greater than one.

2.3.3 File entry mode

Selecting the file entry mode brings the program to execution unless the file is not created properly or the data format is not correct. The data file must be in the following format:

N,M,P,...,S,...

For example, when N=3 and M=5, the data file may look like

```
3,5,.9,.85,.9,.9,.78,'-1-11',
'--111','----1'
```

Commas are used to separate the values and between the lines. Since the program only recognizes the input file with the name -- **INFLSDP.DAT,** it is important to fulfill such requirement.

2.3.4 Output options

If the program has executed successfully, INSDP would request the user to select the output options. User may either select output through screen (monitor) or hard copy option. When the screen output option is selected, INSDP again requests the following options and they are listed:

1. Echo print of input data
2. Equivalent set of disjoint terms
3. System reliability
4. The importance of each component
5. All of the above options
6. Return to options in changing inputs
7. Return to output options
8. Exit to main menu

Option 1 allows the user to check the input data as it is seen by the program. Input error may be detected and changes can be made by selecting Option 6 before another run. The second option shows the equivalent set of disjoint terms and the following option provides the system reliability as calculated by INSDP. Option 4 shows the sorted importance of each component. The fifth option activates Option 1 through 4.

As mentioned earlier, Option 6 may be used to correct or change any input value and Option 7 brings the user back to Output Options. This implies that user may view the output from the screen first, and if the output is satisfactory, he may request a final hard copy. The last option takes the user back to the Main Menu.

The hard copy option does not allow the user to choose various output choices as discussed earlier. All these options are included in the hard copy. The output file is stored under the file name --**OUTFSDP.DAT.** The typical output file command for VAX system is as follows:

$ **PRINT OUTFSDP.DAT/Q**=DESTINATION OF THE PRINTER

User must exit from the INSDP to obtain the hard copy since the PRINT FILE statement is only valid under the VAX operating system.

To assist the users in utilizing the INSDP program, the entire procedure of data inputing and how the output is obtained are illustrated through an example in Appendix B.

2.3.5 Limitation of INSDP

One major limitation of INSDP is its array sizes are all predetermined. Which means for a good size data input, these arrays must be changed accordingly. The responsible arrays are listed as follows:

D,S,Y,Z,T,P,DIF,ARRAY

3.0 **TOPOLOGICAL RELIABILITY**

Satyanarayana and Prabhaker (S&P) [7] introduced topological reliability (TR) in 1978. Over the years, Locks [4], Fischer [8], and Bolaki [9] have further refined the original work of S&P. The following discussion on TR is based on the works of the latter three authors in addition to the original authors.

The major function of TR is locating the p-acyclic subgraphs from the given system reliability graph. The p-acyclic graphs are found by systematic way of stripping away edges of the system graph using the rules formulated by S&P.

3.1 THE ESSENTIAL CONCEPTS OF TR

P-graph is a subgraph of the system graph G_0 , and it has all edges lie on a path from source vertex to the terminal vertex. An acyclic graph has no cycles and cycle graph contains one or more cycles. Using these definitions, a p-acyclic graph is a p-graph without any cycles and a p-cyclic graph is a p-graph with at least one cycle.

The depth-first-search technique is used to find the p-acyclic graphs from the system graph. Which means the system graph must be depicted as a tree. The convention is every node of the tree represents a subgraph and every internode (edge) indicates the removal of either an edge or series of edges to form a subgraph at the next node. Family relationships of the tree must be defined so searching routine can be prioritized. Exhibit 3.1 illustrates the relationship of the family tree of G_i.

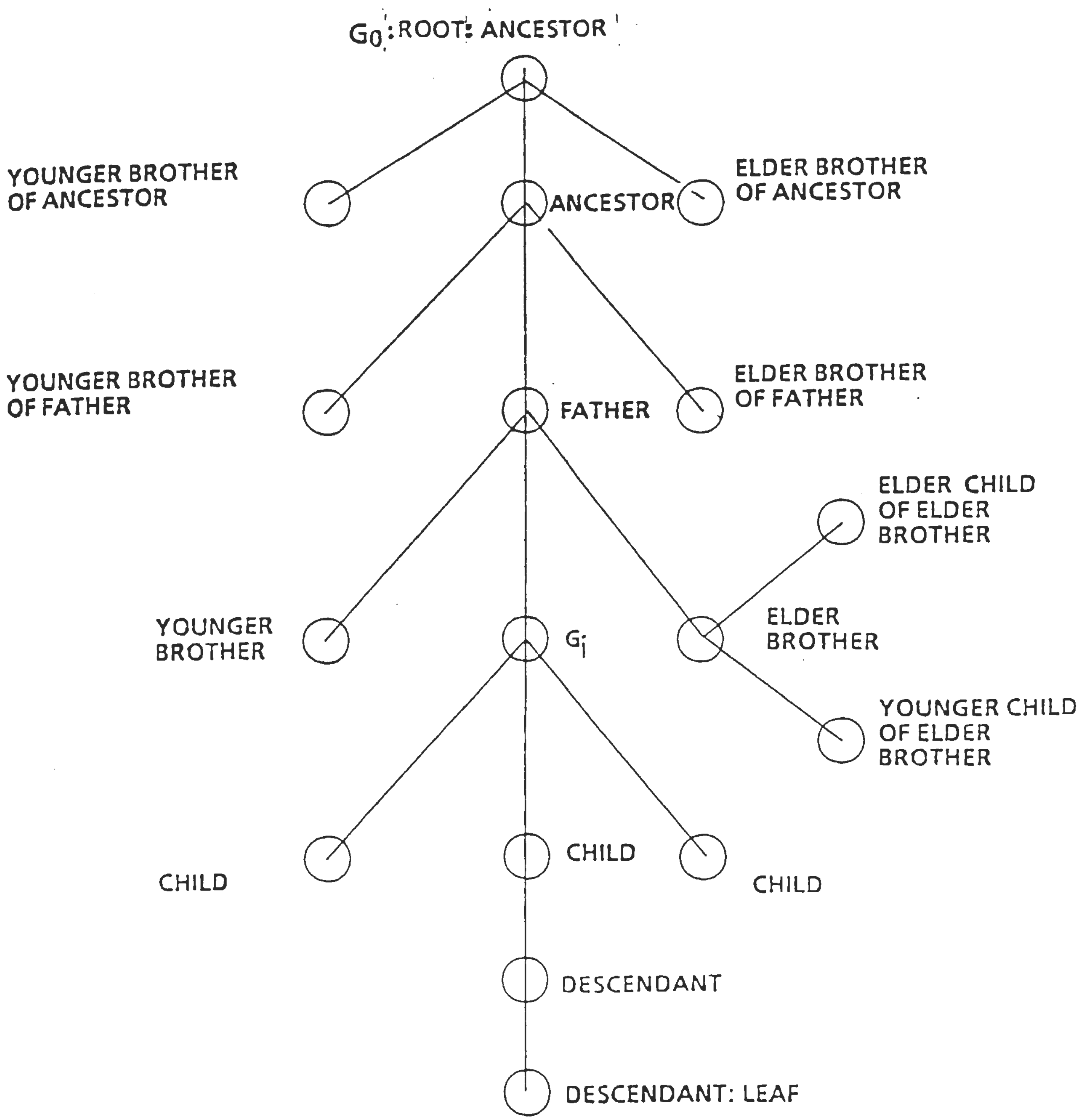

Exhibit 3.1 The Relationship of the Family Tree of Gi
Source: Locks [4]

Fischer [8] pointed out that the depth-first search tree is constructed by visiting the eldest child of the ancestor and then proceeding to the next eldest child. The eldest child is then processed and all of its children. The process will go on until a leaf is encountered. At the given leaf, the procedure backtracks until a node is encountered which has a child that has not been visited. The procedure then visits the eldest child which has not been visited.

3.1.1 The rules of the search

The feature that gives TR formula a high efficient rating is the use of weight restriction (WR). Weight restriction guarantees no duplications of effort during the search. The sets of edges removed from G_0 by G_i is the weight of G_i. The WR demands that G_i may not strip any edge that is in the weight of an elder brother, or an elder brother of the father or of an ancestor.

Four rules are used in TR and they all incorporated the WR. The rules are executed in sequence and they are described as follows.

<u>Rule One:</u> When the subgraph is cyclic, rule one is called to remove the edges, with the exception of the edges that are in the weight of an elder brother or of an elder brother of the father or of any ancestor. When these edges are stripped, the children are created, the rule is one edge per child. Rule one will continue to apply to the child if the graph is found to be cyclic.

Rule Two: In here, the subgraph is acyclic, but it is not p-acyclic. Rule two deletes all the hanging or loose edges that do not lie on a path from start vertex to terminal vertex, subject to the WR.

Rule Three: Rule three is utilized if the subgraph is p-acyclic but does not have a p-acyclic father. This rule searches for all the neutral sequences that can be removed, subject to WR. Neutral sequence is a consecutive string of vertices and edges in a p-acyclic subgraph with no internal vertices connecting to other parts of the subgraph. Each neutral sequence removed gives birth to a child of the given subgraph G_i. This rule guarantees that the following descendents will all be p-acyclic.

Rule Four: Rule four is called when the given p-acyclic subgraph also has a p-acyclic father. Rule four states that all the children of the given p-acyclic subgraph has a 1:1 identical to the younger brothers of the father. The rationale behind this is that all the neutral sequences of the given p-acyclic subgraph were previously identified when the children of the father were generated. Again, this rule is also subject to WR.

3.1.2 The system reliability formula

S&P [7] developed a reliability equation which can be read directly from the search tree. This equation finds the product of the probabilities of all the components of the system graph G_0, and divides the deleted edges from this product.

3.2 STRAP COMPUTER PROGRAM

Fischer [8] wrote the batch-oriented STRAP (Simplified Topological Reliability Analysis Program) in PL/I language to be used on IBM mainframe computer. Strictly speaking, Fischer's program is not based entirely on S&P's original work. STRAP never builds the search tree. Hence, the system reliability is generated differently. However, Fischer claimed that such an approach is more efficient than TR since the complete search tree is not stored. For a large problem, enormous amount of memory space is saved with Fischer's approach.

STRAP provides the users with the following features (1) echo print of the input data (2) input data presented in matrix form (3) linkset (4) importance calculation and (5) trace option. This program also has the capability in handling both the directed and undirected network.

3.2.1 Program composition

STRAP consists mainly of four internal procedures and they are Rule One, Rule Two, Rule Three, and Rule Four. STRAP also uses recursive feature of PL/I language extensively. Due to the nature of recursive feature, the program acts like the search tree without having to build and store the tree. In fact, recursive feature of PL/I works like a stack.

After the input data (edges) is stored under the array GRAPH, Rule One is called . Under this procedure, another subroutine - SEARCH is activated. SEARCH is used to find the cycles in the system graph. The logic behind this subroutine is decribed in relative details by Fischer [8] in his paper.

During the search, SEARCH stores the information on vertices it has been visited in UNA (Unavailable) and in VERTX (the vertices in the order in which they have been visited). Such informations allow the program to recognize a cycle in the system graph. When the cycle is found, REMOVE is called to delete the first edge of the cycle. Rule One is called again recursively until no cycles are found. Rule Two is then called.

Rule Two deletes all the unnecessary edges, such as the hanging edges. This procedure then calls subroutine P-GRAPH to see if the remaining graph is a p-graph. Rule Three is called if it is so.

Rule Three finds all the children of the given graph by deleting one sequence at a time. Subroutine P-GRAPH is called until all sequences have been deleted. The children are stored in the array CHILDREN.

Rule Four deletes the lowest numbered child from CHILDREN. The rule calls itself until a non p-graph is encountered and it returns.

3.3 INTERACTIVE STRAP (INSTRAP)

Similar to INSDP, INSTRAP is also an user-friendly menu-driven program designed to run on the VAX computer system. The complete listing of INSTRAP with documentation is shown in Appendix C. In INSTRAP, the presentation of input data in matrix form is omitted. This feature does not work well for a large problem since the matrix will not fit on the monitor properly.

3.3.1 To run INSTRAP

To run INSTRAP, the program must be loaded up in the VAX system correctly. The program should be compiled and linked accordingly. The user may type in

S **RUN INSTRAP**

3.3.2 Interactive data entry mode

The user may either enter the data from the keyboard or from the VAX file. If the former mode is selected, the user must enter the following items: (1) the number of vertices -N (2) the number of edges -M (3) the number of start vertex -S and (4) the number of terminal vertex -T. Specifically, the user must also enter the information of the edge. For example, 'edge-from' --B, 'edge-to'--E, 'direction of the edge'--D, and 'the probability of the edge' --R. Since the program prompts the user to input the appropriate data, no detailed discussion is needed here. The program also allows the user to change any input before the program execution. Similar to INSDP, error checking routines are incorporated so mistakes are minimized or omitted.

3.3.3 File entry mode

User must have the data file created prior to running the INSTRAP program. The data file should be in the following format:

```
N,M,S,T,
B,E,D,R,
    .
    .
    .
B,E,D,R
```

For example, the system graph in Exhibit 3.2 will have, N=4, M=4, S=1, T=4, and the data file may look like --

```
4,4,1,4,
1,2,1,.9
1,3,1,.9
2,4,1,.9
3,4,1,.9
```

Commas are used to separate the values and between the lines. It is important to use the file name-- **INFSTRAP.DAT** since the program will only recognize this name. Error will occur if other names are used instead.

3.3.4 Output options

Before the program is executed, the user is asked to select the output options for the program. The options are as follows:

1. Echo print, linkset, and reliability
2. Option 1, and importance calculation
3. Trace routine
4. All of the above

The first option gives the user the echo print of the input data, and the linkset of the system graph. In addition, the system reliability is also given. The second option includes

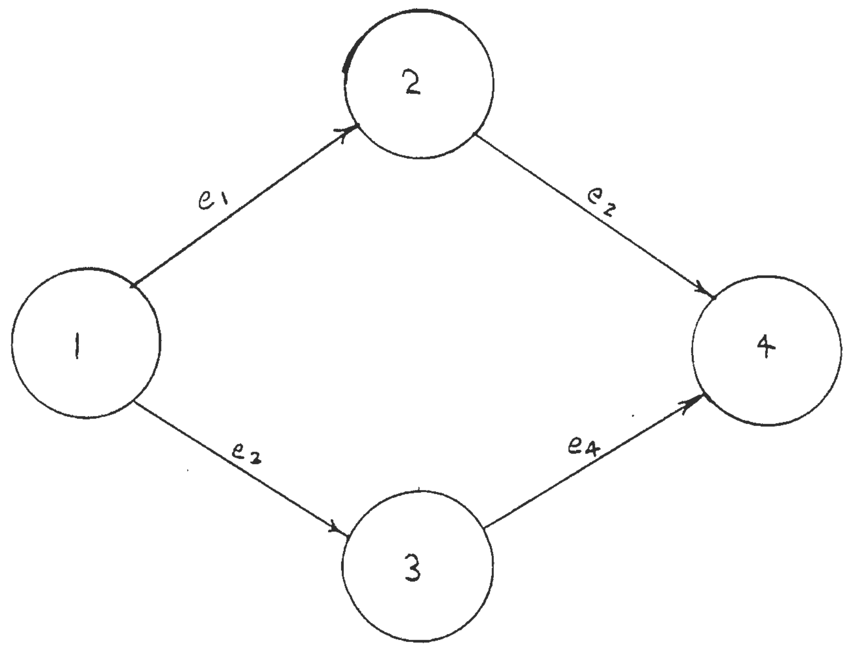

Exhibit 3.2 The Sample System Graph

option #1 and also gives the importance calculation of the edges. The importance of the edge is calculated by finding the linkset and taking the partial derivative of each term with respect to the desired edge. The third option is used to trace the order of edges that are deleted. It is used for diagnosis purpose. Of course, the last option encompasses all the previous options.

In the next step, INSTRAP asks the user to select the output mode. Three options are available and they are (1) screen output (2) hard copy and (3) both.

The user may get the hard copy of the output only after he or she has exited from the INSTRAP program. The output file is stored under the file name-- **OFSTRAP.DAT**. The typical output file command for VAX is --

$ **PRINT OFSTRAP.DAT/Q=** DESTINATION OF THE PRINTER

To assist the users in utilizing INSTRAP program, the entire procedure of data entry and output step are depicted in Appendix D.

3.3.5 Limitation of INSTRAP

INSTRAP does not have a limited array size for its variables as it is with INSDP. Therefore, there is virtually no limitation for this program. However, it may only limit by the memory space available on the VAX system if the input data is astronomically large.

4.0 ACCESSING INSDP AND INSTRAP VIA ASYNCHRONOUS COMMUNICATION NETWORK

The OSU Computer Center allows the asynchronous terminals to access to all the interactive computing systems it offers. The Center network is also accessible via the nationwide **Telenet** public data network. Saving in long distance calls may be realized by using the **Telenet.** More information on **Telenet** is provided in Appendix E.

The OSU network may be connected either by direct connection or by phone dial-up via modem device. For direct connection, the user may simply enter CNTL T (pressing CNTL or CTRL key and T simultaneously) and followed by pressing RETURN. To use the dial-up service, it depends on the user's modem speed (baud rate-bps), terminal type, and the location of use. Exhibit 4.1 shows the proper phone numbers for dial-up.

If the open line is available, the phone call to the OSU Computer Center will be answered and accompanied by a high pitched tone. Connect the telephone to the modem when the tone is heard. Then depress the RETURN key twice.

When the user's terminal is connected to the OSU communication network, the following messages may appear on the screen:

OKLAHOMA STATE UNIVERSITY COMPUTER CENTER NETWORK
XX.XXX
ENTER SYSTEM NAME IN CAPITAL LETTERS (system names)

XX.XXX is an unique identification number given to the current user by the network. In response to the ENTER SYSTEM NAME, user should type **VAX.** If the VAX port is available, the

SPEED	ON CAMPUS	OFF CAMPUS
300 bps (Bell compatible)	7792	405/624-7792
1200 bps (Vadic 3400 and Bell 212 compatible)	7600	405/624-7600

Exhibit 4.1 Phone numbers for dial-up service

user then has the access to the VAX 11/780 system. Additional information on the logon procedure and the network messages are shown in Appendix E.

4.1 TERMINAL SPECIFICATIONS

To have access to the OSU Computer Center asynchronous network, the terminals or computer ports must be configured with the following characteristics:

Character Code.............	ASCII
Character Bits............	7
Parity Bits...............	1
Parity....................	Even
Start Bits................	1
Stop Bits.................	1
Duplex Characteristics....	Full Duplex
Character Echoing.........	To be done by the local device
End of Line Character.....	Carriage Return
Flow Control Technique....	XON (Resume) / XOFF (Suspend)

The OSU Computer Center stated that 'the XON flow control character (hex 11) which can be generated from an ASCII terminal by a CNTL Q. The XOFF control character is an ASCII DC3 control character (hex 13) which can be generated from an ASCII terminal by a CNTL S.'

5.0 CONCLUSION

Both INSDP and INSTRAP programs are much easier to use than their previous versions - SDP and STRAP1, respectively. The programs' ease of use should allow more users to utilize them in research, classroom assignments, etc. Hence, the power and the usefulness of INSDP and INSTRAP may be realized. Hopefully, more bugs in these programs will be discovered with the repeated use. This may be the best way to improve and further enhance these reliability programs.

Eventually, these programs will be implemented on the microcomputers, and the author believed that this study has provided the foundation for such purpose. Because the interactive features of INSDP and INSTRAP are readily transferable to the micro-compiler with few changes that are pertinent to the given compiler.

BIBLIOGRAPHY

[1] Abraham, J.A.,"An Improved Method for Network Reliability", IEEE Trans. Reliability, Vol. R-28, April 1979, pp. 58-61.

[2] Fretta, L. and U.C. Montanari, "A Boolean Algebra Method for Computing the Terminal Reliability in a Communication Network", IEEE Trans. Circuit Theory, Vol. CT-20, May 1973, pp. 203-211.

[3] Aggarwal, K.K., K.B. Misra and I.S. Gupta, "A Fast Algorithm for Reliability Evaluation", IEEE Trans. Reliability, Vol. R-24, April 1975, pp. 83-85.

[4] Locks, Mitchell O., "Some Recent Developments in System Reliability", IEEE Trans. Reliability, Vol. R-34, December 1985, pp.425-436.

[5] Chao, Chih-Pu, "Recursive Disjoint Products: The Reliability Evaluation for both Coherent and Noncoherent Systems", MBA Research Report, Department of Administrative Sciences, Oklahoma State University, June 1983.

[6] Keats, J.B., the Professor of Industrial Engineering at OSU.

[7] Satyanarayana A. and A. Prabhaker, "New Topological Formula and Rapid Algorithm for Reliability Analysis of Complex Networks", IEEE Trans. Reliability, Vol. R-27, June 1978, pp. 82-100.

[8] Fischer, David V., "Computerized System Reliability: Simplified Topological Reliability Analysis Program (STRAP1): Minimum Path Method (MPM)", MBA Research Report, Department of Administrative Sciences, Oklahoma State University, May 1984.

[9] Bolaki, Chandru, "STRAP: Structured Topological Reliability Analysis Program", MBA Research Report, Department of Administrative Sciences, Oklahoma State University, July 1983.

APPENDIX A

INSDP PROGRAM LISTING

```
      INSDP: PROCEDURE OPTIONS(MAIN);
  1   /*****************************************************************************/
  1   /*  MAUNG M. LAY                   INTERACTIVE SUM OF DISJOINT PRODUCTS      */
  1   /*  AUGUST 1985                           (VAX COMPUTER SYSTEM)              */
  1   /*                                                                           */
  1   /*  UPGRADED FROM CHIH-PU CHAO'S SDP PROGRAM                                 */
  1   /*                                                                           */
  1   /*  GIVEN A SET OF MINIMAL STATES, S, FOR A COHERENT OR A NONCOHERENT        */
  1   /*  SYSTEM, THIS PROGRAM DEVELOPS A BOOLEAN FUNCTION CONSISTING OF A         */
  1   /*  SET OF DISJOINT TERMS, D, AND EXAMINES THE IMPORTANCE OF EACH            */
  1   /*  COMPONENT. THE PROGRAM IS, BASICALLY, REVISED FROM DR. KEATS'            */
  1   /*  PROGRAM WHICH USES THE ALGORITHM DEVELOPED BY J.A. ABRAHAM (IEEE         */
  1   /*  TRANSACTION ON RELIABILITY, VOL. R-28, NO. 1, APRIL 1979,PP 58-61)       */
  1   /*                                                                           */
  1   /*****************************************************************************/
  1   /*  VARIABLES----                                                            */
  1   /*                                                                           */
  1   /*  D - THE SET OF DISJOINT TERMS                                            */
  1   /*  S - THE SET OF MINIMAL PATHS OR MINIMAL CUTS                             */
  1   /*  Y - THE SET OF TERMS MADE DISJOINT FROM D                                */
  1   /*  Z - THE SET OF TERMS MADE DISJOINT FROM Y                                */
  1   /*  P - THE SET OF PROBABILITY OF SUCCESS, ASSIGNED TO EACH COMPONENT        */
  1   /*      IN THE SYSTEM                                                        */
  1   /*  N - THE NUMBER OF MINIMAL STATES                                         */
  1   /*  M - THE NUMBER OF COMPONENTS IN THE SYSTEM                               */
  1   /*  K - THE NUMBER OF POSITIONS IDENTIFIED FOR THE POTENTIAL CHANGES         */
  1   /*      WHEN A TERM IN S IS COMPARED WITH A TERM IN D                        */
  1   /*  L - ARRAY OF COMPONENTS WHERE A TERM IN D HAS A 1 OR 0 VALUE AND         */
  1   /*      THE TERM BEING COMPARED HAS A '-' (FREE COMPONENT)                   */
  1   /*  NBR   - THE UPDATED NUMBER OF DISJOINT TERMS IN D                        */
  1   /*  REL   - THE SYSTEM RELIABILITY                                           */
  1   /*  DIF   - THE IMPORTANCE OF EACH COMPONENT                                 */
  1   /*  FLAG  - '0' IF TWO TERMS COMPARED ARE ALREADY DISJOINT,'1' IF NOT        */
  1   /*                                                                           */
  1   /*  L1,L2,L3,L4 - INTEGER VALUES USED IN COMPARISON PURPOSE                  */
  1   /*  COMD - CHARACTER VARIABLE USED TO ACCEPT INPUT FROM TERMINAL             */
  1   /*  ANY - CHARACTER VARIABLE STRING USED TO ACCEPT ANY INPUT FROM             */
  1   /*        TERMINAL                                                           */
  1   /*  ARRAY - STRUCTURED ARRAY WITH IND AND DIF. THIS ARRAY IS USED TO         */
  1   /*          STORE INFORMATION ON DIFFERENTIATED COMPONENTS. THIS ARRAY       */
  1   /*          IS CREATED FOR SHELLSORT ROUTINE IN PROCEDURE IMP.               */
  1   /*                                                                           */
  1   /*  INPUT DATA FILE NAME IS  'INFLSDP.DAT'                                   */
  1   /*  OUTPUT DATA FILE NAME IS 'OUTFSDP.DAT'                                   */
  1   /*                                                                           */
  1   /*****************************************************************************/
  1   /*****************************************************************************/
  1   /*                        DOCUMENTATION OF THE PROGRAM                       */
  1   /*                  (ONLY THE MAIN PROCEDURES ARE LISTED)                    */
  1   /*                                                                           */
  1   /*     MAIN PROCEDURE:                                                       */
  1   /*       INPUT: N,M,P,S                                                      */
  1   /*       OUTPUT: D                                                           */
  1   /*                                                                           */
  1   /*     1. D(1)=S(1),NBR=1                                                    */
```

```
1 /*   2. DO STEPS 3-13 FOR ALL J, 2 <=J<=N                                    */
1 /*   3. COMPARE D(1) TO S(J) BY USING PROCEDURE CMPR. POSITIONS WHERE        */
1 /*      D(1) IS 1 (OR 0 IF THERE IS NO 1) AND S(J) IS '-' ARE STORED         */
1 /*      IN ARRAY L.                                                          */
1 /*   4. IF THE COMPARISON IN STEP RESULTS IN THE IDENTIFICATION OF ONE       */
1 /*      OR MORE POSITIONS, SAY K, THEN USE PROCEDURE NEW TO CREATE NEW      */
1 /*      TERMS DISJOINT WITH D(1) AND ONE ANOTHER. THEN K NEW TERMS ARE      */
1 /*      STORED IN ARRAY Y.                                                   */
1 /*   5. IF THE NUMBER OF TERMS IN SET D IS 1, THEN PUT THE NEWLY            */
1 /*      CREATED TERMS IN SET D AND GO BACK TO STEP 2 BY SETTING J=3.        */
1 /*   6. DO STEPS 7-12 FOR ALL II, 2<=II<=NBR.                                */
1 /*   7. K2=K, JJ=0.                                                          */
1 /*   8. DO STEPS 9 - 11 FOR ALL KK, 1<=KK<=K2.                               */
1 /*   9. COMPARE D(II) TO Y(KK) BY USING CMPR.                                */
1 /*  10. IF THE COMPARISON IN STEP 9 RESULTS IN THE IDENTIFICATION OF        */
1 /*      ONE OR MORE POSITIONS, THEN CALL NEW TO CREATE K1 NEW TERMS         */
1 /*      FROM Y(KK) WHICH ARE PUT IN ARRAY Z. THEN PUT THESE TERMS IN        */
1 /*      THE NEXT AVAILABLE POSITION OF ARRAY T.                              */
1 /*  11. IF THE COMPARISON IN STEP 9 INDICATES THAT D(II) AND Y(KK)          */
1 /*      WERE ALREADY DISJOINT, THEN DO JJ=JJ + 1, T(JJ)=Y(KK).              */
1 /*  12. IF JJ > 0, THEN PUT EACH ELEMENT OF T IN ARRAY Y. K=JJ.              */
1 /*  13. IF K > 0, THEN PUT EACH ELEMENT OF Y IN D SET. EACH TIME AN         */
1 /*      ELEMENT IS PLACED, INCREMENT NBR BY 1.                               */
1 /*                                                                           */
1 /*  PROCEDURE CMPR                                                           */
1 /*    INPUT: THE CURRENT TWO PATHS TO BE COMPARED                            */
1 /*    OUTPUT: K, ARRAY L                                                     */
1 /*                                                                           */
1 /*  1. K=0.                                                                  */
1 /*  2. TO CHECK IF THERE IS ANY '1 0' COMBINATION EXISTS. IF ANY OF IT      */
1 /*     EXISTS, THEN RETURN.                                                  */
1 /*  3. TO CHECK IF THERE IS ANY '0 1' COMBINATION EXISTS. IF ANY OF IT      */
1 /*     EXISTS, THEN RETURN.                                                  */
1 /*  4. CALL THE INTERNAL BLOCK CMPR1 TO SEARCH FOR '1 -' COMBINATIONS.      */
1 /*  5. IF K = 0,I.E., NONE OF '1 -' IS FOUND, THEN CALL CMPR1 AGAIN         */
1 /*     TO LOOK FOR THE '0 -' COMBINATIONS.                                   */
1 /*  6. EACH TIME WHEN A POTENTIAL POSITION IS FOUND, INCREMENT K BY 1       */
1 /*     AND RECORD THE POSITIONS IN L.                                        */
1 /*                                                                           */
1 /*  PROCEDURE NEW                                                            */
1 /*    INPUT: THE TERM WITH SOME POSITIONS IDENTIFIED,K, ARRAY L              */
1 /*    OUTPUT: DISJOINT TERMS OR INTERMEDIATE(PARTIALLY DISJOINT) TERMS      */
1 /*                                                                           */
1 /*  1. DO STEPS 2 - 5 FOR ALL I, 1<=I<=K.                                    */
1 /*  2. CHANGE THE COMPONENTS IN THOSE POSITIONS TO BE 0( OR 1 ASSOCIA-      */
1 /*     TED WITH THE '0 -' COMBINATIONS).                                     */
1 /*  3. IF K = 1 THEN RETURN.                                                 */
1 /*  4. DO STEP 5 FOR ALL J, 1<=J<=I-1.                                       */
1 /*  5. CHANGE ALL THE POSITIONS BEFORE THE CURRENT ONE TO 1 (OR 0          */
1 /*     ASSOCIATED WITH THE '0 -' COMBINATIONS).                              */
1 /*                                                                           */
1 /*  PROCEDURE RELY                                                           */
1 /*    INPUT: THE DISJOINT SET D, THE COMPONENT PROBABILITIES OF SUC-        */
1 /*           CESS P,NBR, THE NUMBER OF COMPONENTS M                          */
1 /*    OUTPUT: THE SYSTEM RELIABILITY REL                                     */
1 /*                                                                           */
1 /*  1. REL = 0.                                                              */
```

```
/*     2. DO STEPS 3-4 FOR ALL I, 1<=I<=NBR.
/*     3. PROD=1
/*     4. DO STEPS 5-6 FOR ALL J, 1<=J<=M.
/*     5. MULTIPLY PROD BY P(J) WHEN A '1' IS FOUND, AND BY (1-P(J)) WHEN
/*        A '0' IS ENCOUNTERED.
/*     6. REL=REL + PROD.
/*
/*     PROCEDURE IMP
/*        INPUT: THE DISJOINT SET D, THE COMPONENT PROBABILITIES OF SUC-
/*               CESS P,NBR, THEN THE NUMBER OF COMPONENTS M.
/*       OUTPUT: SORTED (SHELLSORT) IMPORTANCE OF EACH COMPONENT IN NON-
/*               INCREASING ORDER.
/*
/*     1. OBTAIN THE PROBABILITY OF SUCCESS, AS IT IS DONE IN RELY, FOR
/*        EACH DISJOINT TERM.
/*     2. DO STEP 3 FOR ALL II, 1<=II<=M.
/*     3. DIFFERENTIATE EACH PROBABILITY POLYNOMIAL BY P(II) AND ADD THE
/*        RESULTING DIFFERENTIATED VALUES TOGETHER.
/*     4. SHELLSORT ROUTINE IS CALLED TO SORT THE SYSTEM COMPONENTS.
/*
/**********************************************************************

DCL (L1,L2,L3,L4) FIXED BINARY;
DCL COMD CHAR(1) VAR,
    ANY CHAR(80) VAR;
DCL (D(150),S(100),Y(100),Z(100),T(100)) CHAR(30) VAR;
DCL (N,M,NBR,K,L(90),FLAG,K2,JJ,K1,I,I1,II,KK,J,IND) FIXED BINARY;
DCL (REL,P(100),DIF(100)) FLOAT;
DCL 1 ARRAY(100),
      2 IND FIXED,
      2 DIF FLOAT;
DCL INFLSDP FILE INPUT;
DCL OUTFILE PRINT FILE;


/*                                                              */
PUT PAGE;
PUT SKIP EDIT ('INTERACTIVE SDP PROGRAM')(X(23),A);
PUT SKIP EDIT ('***********************')(X(23),A);
MAIN_MENU:
   PUT SKIP(3) EDIT ('*** MAIN MENU ***')(X(26),A);
   PUT SKIP(2) EDIT ('1. TO ENTER THE DATA INTERACTIVELY')(X(17),A);
   PUT SKIP EDIT ('2. TO ENTER THE DATA THROUGH FILE')(X(17),A);
   PUT SKIP EDIT ('3. EXIT') (X(17),A);
   PUT SKIP(3) EDIT ('ENTER 1,2 OR 3: ')(X(7),A);
   GET LIST(COMD);
/*                                                              */
   IF COMD='1' THEN GO TO INTERACT_DATA;
   IF COMD='2' THEN GO TO LOADFILE_DATA;
   IF COMD='3' THEN STOP;
/*                                                              */
ERROR1:PUT SKIP(3) EDIT('***OUT OF RANGE, REENTER***')(X(10),A);
   CALL ADVANCE;
   GO TO MAIN_MENU;
/*                                                              */
INTERACT_DATA:PUT PAGE;
   PUT SKIP(4);
```

```
    PUT SKIP(2) EDIT ('INTERACTIVE DATA ENTRY MODE')(X(20),A);
    PUT SKIP EDIT ('YOU ARE TO ENTER THE FOLLOWING ITEMS:')(X(17),A);
    PUT SKIP(2) EDIT ('THE NUMBER OF MINIMAL STATES--N')(X(14),A);
    PUT SKIP EDIT ('THE NUMBER OF COMPONENTS IN THE SYSTEM--M')(X(14),A);
    PUT SKIP EDIT ('THE SET OF PROBABILITY OF SUCCESS FOR EACH COMPONENT--P')
                  (X(14),A);
    PUT SKIP EDIT ('THE SET OF MINIMAL PATHS OR MINIMAL CUTS--S')(X(14),A);
    CALL ADVANCE;
 /*                                                                  */
RECEIVE_DATA: PUT PAGE;
    PUT SKIP(2) EDIT('YOU MAY ENTER DATA NOW')(X(20),A);
    PUT SKIP EDIT('IF YOU MADE A MISTAKE(S), REMEMBER WHICH ONE')(X(14),A);
    PUT SKIP EDIT('SINCE YOU CAN CHANGE IT LATER')(X(20),A);

    PUT SKIP(3) EDIT('N= ')(X(7),A);
    GET LIST(N);
    PUT SKIP(2) EDIT('M= ')(X(7),A);
    GET LIST(M);

    ENTER_P:DO I =1 TO M;
      AGAIN1: PUT SKIP(2) EDIT ('P','(',I,')',' = ')(X(7),A,A,F(2),A,A);
      GET LIST(P(I));
      IF (P(I) < 0) | (P(I) > 1) THEN GO TO ERROR12;
      END;

    ENTER_S: PUT PAGE;
    PUT SKIP EDIT ('MAKE SURE YOU INCLUDE THE SINGLE QUOTES BEFORE AND AFTER')
                  (X(10),A);
    PUT SKIP EDIT ('THE STRING OF MINIMAL TERMS, SUCH AS ''1-1-111''')(X(10),A);
    DO J = 1 TO N;
      AGAIN2: PUT SKIP(2) EDIT('S','(',J,')',' = ')(A,A,F(2),A,A);
      GET LIST(S(J));
      IF S(J) ='' THEN GO TO ERROR12A;
      L3=LENGTH(S(J));
      IF L3 ^= M THEN GO TO ERROR13;
      END;

    CHANGE_INPUT: PUT PAGE;
      PUT SKIP(3) EDIT('OPTIONS IN CHANGING INPUT VALUES')(X(21),A);
      PUT SKIP(3) EDIT('1. N VALUE')(X(16),A);
      PUT SKIP EDIT ('2. M VALUE')(X(16),A);
      PUT SKIP EDIT ('3. P VALUE(S)')(X(16),A);
      PUT SKIP EDIT ('4. S VALUE(S)')(X(16),A);
      PUT SKIP EDIT ('5. NO CHANGES REQUIRED,PROGRAM EXECUTION REQUESTED')
                    (X(16),A);
      PUT SKIP EDIT ('6. EXIT TO MAIN MENU')(X(16),A);
      PUT SKIP(3) EDIT ('ENTER 1,2,3,4,5 OR 6  ')(X(10),A);

      GET LIST(COMD);
      IF COMD='1' THEN GO TO NVALUE;
      IF COMD='2' THEN GO TO MVALUE;
      IF COMD='3' THEN GO TO BACKTO_P;
      IF COMD='4' THEN GO TO BACKTO_S;
      IF COMD='5' THEN GO TO SDP;
      IF COMD='6' THEN GO TO MAIN_MENU;

    ERROR8: PUT SKIP(3) EDIT('***OUT OF RANGE, REENTER***')(X(15),A);
```

```
1          CALL ADVANCE;
1          GO TO CHANGE_INPUT;
1
1       NVALUE: PUT PAGE;
1          PUT SKIP(3) EDIT('N = ')(X(7),A);
1          GET LIST(N);
1          GO TO CHANGE_INPUT;
1
1       MVALUE: PUT PAGE;
1          PUT SKIP(3) EDIT('M = ')(X(7),A);
1          GET LIST(M);
1          GO TO CHANGE_INPUT;
1  BACKTO_P: PUT PAGE;
1          PUT SKIP(2) EDIT('OPTIONS IN REENTERING P VALUES')(X(20),A);
1          PUT SKIP(3) EDIT('1. REENTER ALL VALUES')(X(15),A);
1          PUT SKIP EDIT('2. SELECTIVELY')(X(15),A);
1          PUT SKIP(3) EDIT('ENTER 1 OR 2:  ')(X(10),A);
1          GET LIST(COMD);
1          IF COMD='1' THEN GO TO ENTER_P;
1          IF COMD='2' THEN GO TO SELECT_P;
1
1       ERROR9: PUT SKIP(3) EDIT('***INCORRECT INPUT, REENTER***')(X(10),A);
1          CALL ADVANCE;
1          GO TO BACKTO_P;
1
1       SELECT_P: PUT PAGE;
1          PUT SKIP(3) EDIT('WHICH P VALUE IS TO BE ALTERED?')(X(20),A);
1          PUT SKIP EDIT('ENTER 1,2, ETC. ')(X(22),A);
1          PUT SKIP(3) EDIT('ENTER ONE NUMBER NOW    ')(X(10),A);
1          GET LIST(L3);
1          IF (L3 < 1) | (L3 > M) THEN GO TO ERROR10;
1             AGAIN3:PUT SKIP(3) EDIT('P','(',L3,')',' = ')(A,A,F(2),A,A);
1             GET LIST (P(L3));
1             IF (P(L3) < 0) | (P(L3) > 1) THEN GO TO ERROR14;
1
1        CHANGE_P: PUT SKIP(3) EDIT('MORE CHANGES? ENTER Y/N  ')(X(20),A);
1          GET LIST(COMD);
1          IF COMD='Y' THEN GO TO SELECT_P;
1          IF COMD='N' THEN GO TO CHANGE_INPUT;
1
1       ERROR11:PUT SKIP(3) EDIT ('***INCORRECT INPUT, REENTER***')(X(10),A);
1          CALL ADVANCE;
1          GO TO CHANGE_P;
1
1       ERROR10: PUT SKIP(3) EDIT('***OUT OF RANGE, REENTER***')(X(10),A);
1          CALL ADVANCE;
1          GO TO SELECT_P;
1
1       BACKTO_S:PUT PAGE;
1          PUT SKIP(2) EDIT('OPTIONS IN REENTERING S VALUES')(X(20),A);
1          PUT SKIP(3) EDIT('1. REENTER ALL VALUES')(X(15),A);
1          PUT SKIP EDIT('2. SELECTIVELY')(X(15),A);
1          PUT SKIP(3) EDIT('ENTER 1 OR 2:  ')(X(10),A);
1          GET LIST(COMD);
1          IF COMD='1' THEN GO TO ENTER_S;
1          IF COMD='2' THEN GO TO SELECT_S;
1
```

```
    ERROR15:PUT SKIP(3) EDIT('***INCORRECT INPUT, REENTER***')(X(10),A);
       CALL ADVANCE;
       GO TO BACKTO_S;

    SELECT_S:PUT PAGE;
       PUT SKIP(3) EDIT('WHICH S VALUE IS TO BE ALTERED?')(X(20),A);
       PUT SKIP EDIT('ENTER 1,2, ETC. ')(X(22),A);
       PUT SKIP(3) EDIT('ENTER ONE NUMBER NOW  ')(X(10),A);
       GET LIST(L3);
       IF (L3 < 1) ! (L3 > N) THEN GO TO ERROR16;
          AGAIN4:PUT SKIP(3) EDIT('S','(',L3,')',' = ')(A,A,F(2),A,A);
          GET LIST (S(L3));
          L4=LENGTH(S(L3));
          IF L4 ^= M THEN GO TO ERROR17;

    CHANGE_S: PUT SKIP(3) EDIT('MORE CHANGES? ENTER Y/N  ')(X(20),A);
       GET LIST(COMD);
       IF COMD='Y' THEN GO TO SELECT_S;
       IF COMD='N' THEN GO TO CHANGE_INPUT;

    ERROR18:PUT SKIP(3) EDIT('***INCORRECT INPUT, REENTER***')(X(10),A);
       CALL ADVANCE;
       GO TO CHANGE_S;


    ERROR12:PUT SKIP(3) EDIT('*VALUE IS LESS THAN 0 OR GREATER THAN 1,REENTER*')
                                (X(10),A);
       CALL ADVANCE;
       GO TO AGAIN1;

    ERROR12A: PUT SKIP(3) EDIT('**SINGLE QUOTE(S) MISSING, REENTER**')(X(10),A);
       CALL ADVANCE;
       GO TO AGAIN2;

    ERROR13: PUT SKIP(3) EDIT('***LENGTH NOT EQUAL TO M,REENTER***')(X(10),A);
       CALL ADVANCE;
       GO TO AGAIN2;

    ERROR14:PUT SKIP(3) EDIT('*VALUE IS LESS THAN 0 OR GREATER THAN 1,REENTER*')
                                (X(10),A);
       CALL ADVANCE;
       GO TO AGAIN3;

    ERROR16:PUT SKIP(3) EDIT('***OUT OF RANGE, REENTER***')(X(10),A);
       CALL ADVANCE;
       GO TO SELECT_S;

    ERROR17:PUT SKIP(3) EDIT('***LENGTH NOT EQUAL TO M,REENTER***')(X(10),A);
       CALL ADVANCE;
       GO TO AGAIN4;


   /*                                                                    */

 LOADFILE_DATA: PUT PAGE;
    PUT SKIP EDIT('YOUR DATA FILE SHOULD BE ALREADY CREATED')(X(20),A);
    PUT SKIP EDIT('IT MUST BE NAMED INFLSDP.DAT')(X(20),A);
```

```
1      PUT SKIP EDIT('THE DATA FILE SHOULD BE IN THE FOLLOWING FORMAT')
1                    (X(20),A);
1      PUT SKIP(2) EDIT('N,M,...,P,...,...,S,...')(X(20),A);
1      PUT SKIP EDIT('13,12,.9,.9,.9,.9,.9,.9,.86,.79,.9,')(X(20),A);
1      PUT SKIP EDIT('.9,.98,.9,''1-11---------'',''1-1111-----1'',')
1                    (X(20),A);
1      PUT SKIP EDIT('''-1---1---1--'',...,''---1111--1-1''')(X(20),A);
1      PUT SKIP(2) EDIT('NOTE: USE COMMA TO SEPARATE THE VALUES,AND BETWEEN LINES')
1                      (X(20),A);
1      PUT SKIP(2) EDIT('IF NOT, TYPE IN- E TO EXIT TO MAIN MENU')
1                      (X(20),A);
1      PUT SKIP EDIT('IF YES, TYPE IN- C TO CONTINUE  ')(X(20),A);
1  /*                                                                  */
1      GET LIST(COMD);
1      IF COMD='E' THEN GO TO MAIN_MENU;
1      IF COMD='C' THEN GO TO READFILE;
1  /*                                                                  */
1  ERROR2: PUT SKIP(3) EDIT('***INPUT ERROR DETECTED, REENTER***')(X(10),A);
1      CALL ADVANCE;
1      GO TO LOADFILE_DATA;
1  /*                                                                  */
1  READFILE:PUT PAGE;
1      PUT SKIP(3) EDIT('***INITIATE READING OF INFILSDP.DAT***')(X(20),A);

1      OPEN FILE(INFLSDP) INPUT;
1      GET FILE(INFLSDP) LIST (N,M);
1  /*                                                                  */
1      DO I= 1 TO M;
2        GET FILE(INFLSDP) LIST (P(I));
2        IF (P(I) < 0) | (P(I) > 1) THEN GO TO ERROR4;
2        END;
1  /*                                                                  */
1      DO J= 1 TO N;
2        GET FILE(INFLSDP) LIST (S(J));
2        L4=LENGTH(S(J));
2        IF L4 ^= M THEN GO TO ERROR5;
2        END;
1  /*                                                                  */
1      CLOSE FILE(INFLSDP);
1  /*                                                                  */
1      GO TO SDP;
1  /*                                                                  */
1      ERROR4: PUT PAGE;
1      PUT SKIP(3) EDIT('***P VALUE IS < 0 OR > 1,CHECK INPUT FILE***')
1                      (X(10),A);
1      CALL ADVANCE;
1      GO TO MAIN_MENU;
1  /*                                                                  */
1      ERROR5: PUT PAGE;
1      PUT SKIP(3) EDIT('***ERROR DETECTED WITH S VALUES, CHECK INPUT FILE***')
1                      (X(10),A);
1      CALL ADVANCE;
1      GO TO MAIN_MENU;
1  /*                                                                  */
1  SDP: PUT PAGE;
1      PUT SKIP(15) EDIT('**PROGRAM IS EXECUTING, PLEASE WAIT**')(X(19),A);
1      D(1)=S(1);
```

 29-DEC-19

```
1          NBR=1;
1     /*                                                    */
1          DO J=2 TO N;
2            CALL CMPR(D(1),S(J),K,L,FLAG,I1);
2
2            IF K > 0 THEN DO;
3              CALL NEW(S(J),Y,I1,K,L);
3              END;
2            IF K = 0 & FLAG = 0 THEN DO;
3              NBR = NBR + 1;
3              D(NBR) = S(J);
3              END;
2            IF K > 0 & NBR = 1 THEN DO;
3              NBR = K + 1;
3              DO I = 1 TO K;
4                D(I + 1)= Y(I);
4                END;
3     /*                                                    */
3              GO TO LOOP_OUT;
3              END;
2     /*                                                    */
2          DO II = 2 TO NBR;
3            K2 = K;
3            JJ = 0;
3          DO KK = 1 TO K2;
4            CALL CMPR(D(II),Y(KK),K1,L,FLAG,I1);
4            IF K1 > 0 THEN DO;
5              CALL NEW(Y(KK),Z,I1,K1,L);
5              DO I = 1 TO K1;
6                T(I +JJ) = Z(I);
6                END;
5              JJ = JJ + K1;
5              END;
4            IF K1 = 0 & FLAG = 0 THEN DO;
5              JJ = JJ + 1;
5              T(JJ) = Y(KK);
5              END;
4            END;
3     /*                                                    */
3            IF JJ > 0 THEN DO;
4              DO I = 1 TO JJ;
5                Y(I) = T(I);
5                END;
4              END;
3              K= JJ;
3             END;
2     /*                                                    */
2            IF K > 0 THEN DO;
3              DO I = 1 TO K;
4                NBR = NBR + 1;
4                D(NBR) = Y(I);
4                END;
3            END;
2     /*                                                    */
2     LOOP_OUT: END;
1
1
```

```
/*     ***** OUTPUT SECTION *****-    */

PUT PAGE;
PUT SKIP(3) EDIT ('**EXECUTION HAS COMPLETED**')(X(24),A);
PRINTOUT: PUT SKIP(4) EDIT ('OUTPUT OPTIONS')(X(29),A);
    PUT SKIP(2) EDIT ('1. OUTPUT THROUGH SCREEN') (X(24),A);
    PUT SKIP(2) EDIT ('2. HARD COPY') (X(24),A);
    PUT SKIP(3) EDIT ('ENTER 1 OR 2  ')(X(14),A);
    GET LIST(COMD);

    IF COMD='1' THEN GO TO SCREEN_OUTPUT;
    IF COMD='2' THEN GO TO PRINT_OUTPUT;
/*                                        */
ERROR6: PUT PAGE;
    PUT SKIP(15) EDIT('***OUT OF RANGE,REENTER THE VALUE***')(X(7),A);
    CALL ADVANCE;
    GO TO PRINTOUT;
/*                                        */
SCREEN_OUTPUT: PUT PAGE;
    PUT SKIP(4) EDIT ('SCREEN_OUTPUT OPTIONS') (X(26),A);
    PUT SKIP(2) EDIT ('1. ECHO PRINT OF INPUT DATA')(X(22),A);
    PUT SKIP EDIT ('2. EQUIVALENT SET OF DISJOINT TERMS')(X(22),A);
    PUT SKIP EDIT ('3. SYSTEM RELIABILITY')(X(22),A);
    PUT SKIP EDIT ('4. THE IMPORTANCE OF EACH COMPONENT')(X(22),A);
    PUT SKIP EDIT ('5. ALL OF THE ABOVE OPTIONS')(X(22),A);
    PUT SKIP EDIT ('6. RETURN TO OPTIONS IN CHANGING INPUTS')(X(22),A);
    PUT SKIP EDIT ('INTERACTIVELY')(X(25),A);
    PUT SKIP EDIT ('7. RETURN TO OUTPUT OPTIONS')(X(22),A);
    PUT SKIP EDIT ('8. EXIT TO MAIN MENU')(X(22),A);
    PUT SKIP(3) EDIT('ENTER 1,2,3,4,5,6, 7 OR 8  ') (X(17),A);
    GET LIST(COMD);
    IF COMD='1' THEN GO TO ECHO_DATA;
    IF COMD='2' THEN GO TO DISJOINT;
    IF COMD='3' THEN GO TO RELIA;
    IF COMD='4' THEN GO TO IMPOR;
    IF COMD='5' THEN GO TO ECHO_DATA;
    IF COMD='6' THEN GO TO CHANGE_INPUT;
    IF COMD='7' THEN GO TO PRINTOUT;
    IF COMD='8' THEN GO TO MAIN_MENU;
 /*                                                           */
ERROR7: PUT PAGE;
    PUT SKIP(15) EDIT ('***OUT OF RANGE, REENTER THE VALUE***')(X(18),A);
    CALL ADVANCE;
    GO TO SCREEN_OUTPUT;
/*                                        */
ECHO_DATA: PUT PAGE;
    PUT SKIP EDIT ('ECHO PRINT OF INPUT DATA')(X(20),A);
    PUT SKIP(3) EDIT ('N VALUE IS  ')(X(25),A);
    PUT EDIT (N)(A);
    PUT SKIP(3) EDIT ('M VALUE IS  ')(X(25),A);
    PUT EDIT (M)(A);
    CALL ADVANCE;
    PUT PAGE;
    PUT SKIP EDIT ('ORIGINAL MINIMAL TERMS')(X(20),A);
    PUT SKIP;
    DO J = 1 TO N BY 18;
       DO I = J TO J+17;
```

```
3          IF I <= N THEN PUT SKIP EDIT(S(I))  (X(3),A);
3        END;
2        CALL ADVANCE;
2      END;
1
1      PUT PAGE;
1      PUT SKIP EDIT ('ECHO PRINT OF PROBABILITY VALUES')(X(20),A);
1      PUT SKIP;
1      DO J = 1 TO M BY 18;
2        DO I = J TO J+17;
3          IF I <= M THEN PUT SKIP EDIT(P(I))  (X(3),F(5,3));
3        END;
2        CALL ADVANCE;
2      END;
1
1      IF COMD='5' THEN GO TO DISJOINT;
1      GO TO SCREEN_OUTPUT;
1   /*                                               */
1   DISJOINT: PUT PAGE;
1      PUT SKIP EDIT('EQUIVALENT OF DISJOINT TERMS')(X(20),A);
1      PUT SKIP;
1      DO J = 1 TO NBR BY 18;
2        DO I = J TO J+17;
3          IF I <= NBR THEN PUT SKIP EDIT(D(I))  (X(3),A);
3        END;
2        CALL ADVANCE;
2      END;
1
1      IF COMD='5' THEN GO TO RELIA;
1      GO TO SCREEN_OUTPUT;
1   /*                                               */
1   RELIA: CALL RELY(D,P,M,REL,NBR);
1      PUT PAGE;
1      PUT SKIP(2);
1      PUT SKIP EDIT('SYSTEM RELIABILITY')(X(25),A);
1      PUT EDIT(REL)  (F(10,5));
1      CALL ADVANCE;
1      IF COMD='5' THEN GO TO IMPOR;
1      GO TO SCREEN_OUTPUT;
1   /*                                               */
1   IMPOR: PUT PAGE;
1      CALL IMP(D,P,DIF,M,NBR,IND,ARRAY);
1      PUT SKIP EDIT('THE IMPORTANCE OF EACH COMPONENT IN SORTED ORDER')(X(15),A);
1      PUT SKIP(2) EDIT('COMPONENT NO.','IMPORTANCE CALCULATION')(A,X(12),A);
1      DO J =1 TO M BY 18;
2        DO I = J TO J+17;
3          IF I <= M THEN PUT SKIP EDIT(ARRAY(I).IND)  (X(5),F(3));
3          IF I <= M THEN PUT EDIT(ARRAY(I).DIF)(X(18),F(10,6));
3        END;
2        CALL ADVANCE;
2      END;
1
1      PUT SKIP(3) EDIT ('THE MOST IMPORTANT COMPONENT IS:')(X(15),A);
1      PUT EDIT(IND)(F(3));
1      CALL ADVANCE;
1      GO TO SCREEN_OUTPUT;
1   /*                                               */
```

```
1 ADVANCE: PROC;
2    PUT SKIP(3) EDIT('PRESS RETURN TO CONTINUE   ')(X(10),A);
2    GET LIST(ANY);
2    RETURN;
2    END ADVANCE;
1 /*                                       */
1 PRINT_OUTPUT: OPEN FILE(OUTFILE) TITLE ('OUTFSDP.DAT');
1   PUT FILE(OUTFILE) SKIP EDIT('ECHO PRINT OF INPUT DATA')(X(20),A);
1   PUT FILE(OUTFILE) SKIP(3) EDIT('N VALUE IS    ')(X(25),A);
1   PUT FILE(OUTFILE) EDIT (N)(A);
1   PUT FILE(OUTFILE) SKIP(3) EDIT('M VALUE IS    ')(X(25),A);
1   PUT FILE(OUTFILE) EDIT (M)(A);
1   PUT FILE(OUTFILE) PAGE;
1   PUT FILE(OUTFILE) SKIP(2) EDIT('ORIGINAL MINIMAL TERMS')(X(20),A);
1   PUT FILE(OUTFILE) SKIP(3);
1   DO J = 1 TO N;
2     PUT FILE(OUTFILE) SKIP EDIT(S(J))(X(1),A);
2     END;
1   PUT FILE(OUTFILE) PAGE;
1   PUT FILE(OUTFILE) SKIP EDIT('ECHO PRINT OF PROBABILITY VALUES')(X(20),A);
1   DO J = 1 TO M;
2     PUT FILE(OUTFILE) SKIP EDIT(P(J))(X(5),F(5,3));
2     END;
1
1   PUT FILE(OUTFILE) PAGE;
1   PUT FILE(OUTFILE) SKIP EDIT('EQUIVALENT OF DISJOINT TERMS')(X(20),A);
1   PUT FILE(OUTFILE) SKIP(2);
1   DO J = 1 TO NBR;
2     PUT FILE(OUTFILE) SKIP EDIT(D(J))(X(1),A);
2     END;
1
1   CALL RELY(D,P,M,REL,NBR);
1   PUT FILE(OUTFILE) PAGE;
1   PUT FILE(OUTFILE) SKIP(2) EDIT('SYSTEM RELIABILITY')(X(25),A);
1   PUT FILE(OUTFILE) EDIT (REL)(F(10,5));
1
1   CALL IMP(D,P,DIF,M,NBR,IND,ARRAY);
1   PUT FILE(OUTFILE) PAGE;
1   PUT FILE(OUTFILE) SKIP(2) EDIT('THE SORTED IMPORTANCE OF EACH COMPONENT')
1                              (X(15),A);
1   PUT FILE(OUTFILE) SKIP(2) EDIT('COMPONENT NO.','IMPORTANCE CALCULATION')
1                              (A,X(12),A);
1   DO J = 1 TO M;
2     PUT FILE(OUTFILE) SKIP EDIT(ARRAY(J).IND)(X(5),F(3));
2     PUT FILE(OUTFILE) EDIT(ARRAY(J).DIF) (X(18),F(10,6));
2     END;
1   PUT FILE(OUTFILE) SKIP(3) EDIT('THE MOST IMPORTANT COMPONENT IS:')
1                              (X(15),A);
1   PUT FILE(OUTFILE) EDIT(IND)(F(3));
1   CLOSE FILE(OUTFILE);
1
1   PUT PAGE;
1   PUT SKIP(3) EDIT('YOUR HARD COPY IS READY. YOU WILL RECEIVE AN ECHO')
1                  (X(16),A);
1   PUT SKIP EDIT   ('PRINT OF INPUT DATA, EQUIVALENT OF DISJOINT TERMS,')
1                  (X(16),A);
1   PUT SKIP EDIT   ('SYSTEM RELIABILITY, THE IMPORTANCE OF EACH COMPO- ')
```

```
1                         (X(16),A);
1     PUT SKIP EDIT       ('NENT, AND THE MOST IMPORTANT COMPONENT OF SYSTEM.')
1                         (X(16),A);
1     PUT SKIP(2) EDIT('OUTPUT FILE NAME IS OUTFSDP.DAT')(X(16),A);
1     CALL ADVANCE;
1
1     PUT PAGE;
1     PUT SKIP(3) EDIT('IF YOU WISH TO RECEIVE THE HARD COPY, YOU SHOULD')
1                         (X(16),A);
1     PUT SKIP EDIT       ('EXIT AT THE MAIN MENU. THEN USE THE APPROPRIATE ')
1                         (X(16),A);
1     PUT SKIP EDIT       ('PRINT COMMAND, SUCH AS $ PRINT OUTFSDP.DAT/Q=BUS')
1                         (X(16),A);
1     CALL ADVANCE;
1     GO TO MAIN_MENU;
1
1
1  /*                                               */
1
1  END INSDP;


   /*************************************************************************/
   /* PROCEDURE CMPR                                                        */
   /*                                                                       */
   /* VARIABLES ---                                                         */
   /*    S - THE TERM IN DISJOINT SET                                       */
   /*    T - THE TERM BEING COMPARED WITH S                                 */
   /*                                                                       */
   /* DESCRIPTION ---                                                       */
   /*                                                                       */
   /* THIS PART OF THE PROGRAM IS TRYING TO CHECK IF THE TWO TERMS          */
   /* COMPARED ARE ALREADY DISJOINT. IF THEY ARE, THEN RETURN TO            */
   /* MAIN PROGRAM. OTHERWISE, FIND THE POTENTIAL POSITIONS FOR             */
   /* MAKING THEM DISJOINT.                                                 */
   /*                                                                       */
   /*************************************************************************/

   CMPR: PROC(S,T,K,L,FLAG,I1);
1  DCL  (S,T) CHAR(*) VAR;
1  DCL  (K,L(*),FLAG,I1,J1,I,II,J) FIXED BINARY;
1  DCL  IN FIXED BINARY;
1  DCL  (A,B,U,V,R,E,F,U1,V1) CHAR(30) VAR;
1  DCL  ANY CHAR(80) VAR;
1
1  K=0; IN=0; I1=0; FLAG=1;
1  A= '1'; B= '0'; U=S; V=T; E=S; F=T; U1=S; V1=T;
1
1  I= INDEX(U,A);      /* LOOK FOR '1' IN THE GIVEN MINIMAL CUT */
1
1  IF I = 0 THEN DO;     /* NO '1' IS FOUND                      */
2    J = INDEX(U,B);      /* LOOK FOR '0' IN THE GIVEN MINIMAL CUT */
2    IF J = 0 THEN RETURN;    /* NEITHER '1' NOR '0' IS FOUND    */
2    END;
1
1  /* THIS DO WHILE LOOP SEARCHES 1-)0 COMBINATION                 */
1  /*                                                              */
```

```
DO WHILE ('1'B);
  I=INDEX(U,A);
  IF I = 0 THEN GO TO OUT2;
  IF SUBSTR(V,I,1) = B THEN DO; /* 1->0 COMBINATION EXISTS    */
    FLAG=0;
    RETURN;    /* THE TWO PATHS ARE DISJOINT                   */
    END;

  IF I + 1 > LENGTH(U) THEN GO TO OUT2; /* IF 1->0 COMBINATION DOES NOT */
                                        /* EXIST FOR THE GIVEN COMPONENT*/
                                        /* OF THE PATHS, MOVE TO NEXT    */
                                        /* COMPONENT. IF NEXT COMPONENT */
                                        /* IS EMPTY, GOTO OUT2           */
    U=SUBSTR(U,I + 1);
    V=SUBSTR(V,I + 1);

      END;
OUT2:;

/* THIS DO WHILE SEARCHES FOR 0->1 COMBINATION                  */
/*                                                              */

DO WHILE('1'B);
  J1= INDEX(U1,B);
  IF J1 =0 THEN GO TO OUT3;  /* IF 0->1 COMBINATION EXISTS,THEN DO  */
  IF SUBSTR(V1,J1,1)= A THEN DO;
  FLAG=0;
  RETURN;                    /* TWO PATHS ARE DISJOINT             */
  END;

  IF J1 + 1 > LENGTH(U1) THEN GO TO OUT3; /* IF 0->1 COMBINATION DOES NOT */
                                          /* FOR THE GIVEN COMPONENT OF   */
                                          /* THE PATHS, MOVE TO NEXT COM- */
                                          /* PONENT. IF NEXT POSITION IS  */
                                          /* EMPTY, GOTO OUT3.            */
    U1= SUBSTR(U1,J1 + 1);
    V1= SUBSTR(V1,J1 + 1);

  END;
OUT3:;

/* WHEN EXECUTION REACHES HERE, THAT MEANS NO '1->0' OR '0->1' COMBINATION */
/* IS FOUND BETWEEN THE PATHS. CALL CMPR1 TO SEARCH FOR '1->-' COMBINATION */
/* CONSEQUENTLY, '0->-' COMBINATION IS ALSO SEARCHED.                      */
/*                                                                         */

I1=1;
CALL CMPR1;
IF K = 0 THEN DO; /* NO '1->-' COMBINATION IS FOUND  */
  E=S;
  F=T;
  I1=0;
  IN=0;
/*          SEARCH FOR '0->-' COMBINATION          */
  CALL CMPR1;
  END;
```

```
     1
     1  /* CMPR1 SEARCHES FOR '1->-' OR '0->-' COMBINATION DEPENDING ON I1 VALUE *
     1  /*                                                                       *
     1
     1  CMPR1: PROC;
     2  IF I1=1 THEN R ='1';
     2          ELSE R ='0';
     2  DO WHILE ('1'B);
     3    II = INDEX(E,R);
     3    IF II=0 THEN GO TO OUT4;
     3    IN = IN + II;
     3    IF SUBSTR(F,II,1)='-' THEN DO; /* POSITION OF THE FREE COMPONENT IS    *
     4                                   /* REMEMBERED. */
     4      K=K+1;
     4      L(K)= IN;
     4
     4      END;
     3    IF II + 1 > LENGTH(E) THEN RETURN;  /*CHECK WHETHER THE END OF PATH    *
     3                                        /*IS REACHED. IF NOT, MOVE TO NEXT *
     3                                        /*POSITION OF THE PATH.            *
     3      E= SUBSTR(E, II + 1);
     3      F= SUBSTR(F, II + 1);
     3
     3      END;
     2  OUT4:;
     2
     2  RETURN;
     2  END CMPR1;
     1  /*                              */
     1  RETURN;
     1  END CMPR;

        /***********************************************************************/
        /*                                                                     */
        /* PROCEDURE NEW                                                       */
        /*                                                                     */
        /* VARIABLES ---                                                       */
        /*    X - THE TERM FROM WHICH NEW TERMS WILL BE GENERATED              */
        /*    Y - SET OF NEW DISJOINT OR INTERMEDIATE (PARTIALLY DISJOINT)     */
        /*        TERMS GENERATED FROM X                                       */
        /*                                                                     */
        /* DESCRIPTION ---                                                     */
        /* THIS PROCEDURE GENERATES K NEW TERMS, Y(1),Y(2)....., Y(K)          */
        /* FROM TERM X. EACH NEW TERM IS DISJOINT FROM ONE ANOTHER AND         */
        /* FROM THE CURRENT DISJOINT TERM WHICH X IS BEING COMPARED WITH       */
        /*                                                                     */
        /***********************************************************************/

        NEW: PROC(X,Y,I1,K,L);
     1  DCL (X,Y(*)) CHAR(*) VAR;
     1  DCL (I1,K,L(*)) FIXED  BINARY;
     1  DCL (I,J) FIXED BINARY;
     1  DCL (A,B) CHAR(30) VAR;
     1
     1  IF I1 = 1 THEN DO;
     2    A='0';
     2    B='1';
```

```
      END;
      ELSE DO;
      A='1';
      B='0';
      END;

   DO I = 1 TO K;
      Y(I) = X;
      SUBSTR(Y(I),L(I),1)=A;
      IF K = 1 THEN RETURN;
      DO J = 1 TO I - 1;       /* THIS DO LOOP IS USED TO DISJOINT THE NEW TERMS   */
         SUBSTR(Y(I),L(J),1)=B; /* FROM EACH OTHER */
         END;
      END;

   RETURN;
   END NEW;

   /***********************************************************************************/
   /*                                                                                 */
   /* PROCEDURE RELY                                                                  */
   /*                                                                                 */
   /* THIS PROCEDURE CALCULATES THE SYSTEM RELIABILITY BY SUMMING UP THE              */
   /* PROBABILITY OF SUCCESS OF ALL THE DISJOINT TERMS.                               */
   /*                                                                                 */
   /***********************************************************************************/

   RELY: PROC(D,P,M,REL,NBR);
   DCL D(*) CHAR(*) VAR;
   DCL (P(*),REL) FLOAT;
   DCL (M,NBR,I,J) FIXED BINARY;
   DCL PROD FLOAT;

   REL=0.0;
   DO I = 1 TO NBR;
      PROD=1.0;
      DO J =1 TO M;
         IF SUBSTR(D(I),J,1)='1' THEN PROD = PROD * P(J);
         IF SUBSTR(D(I),J,1)='0' THEN PROD = PROD * (1 - P(J));
         END;
      REL = REL + PROD;
      END;

   RETURN;
   END RELY;

   /***********************************************************************************/
   /*                                                                                 */
   /* PROCEDURE  IMP                                                                  */
   /*                                                                                 */
   /* DESCRIPTION ---                                                                 */
   /* THE IMPORTANCE OF A COMPONENT IS THE PARTIAL DERIVATIVE OF THE SYSTEM           */
   /* PROBABILITY WITH RESPECT TO THE PROBABILITY OF THE COMPONENT. THE               */
   /* PROCEDURE BELOW PERFORMS THIS CALCULATION FOR EACH COMPONENT.                   */
   /*                                                                                 */
   /***********************************************************************************/
```

```
    IMP: PROC(D,P,DIF,M,NBR,IND,ARRAY);
 1  DCL D(*) CHAR(*) VAR;
 1  DCL (P(*),DIF(*),PROD(100),TEMP,TEMP1,MAX) FLOAT;
 1  DCL (M,NBR,IND,I,J,II,JJ) FIXED BINARY;
 1  DCL 1 ARRAY(*),
 1        2 IND FIXED,
 1        2 DIF FLOAT;
 1
 1  DO I = 1 TO NBR;
 2     TEMP=1.0;
 2     DO J = 1 TO M;
 3        IF SUBSTR(D(I),J,1)='1' THEN TEMP=TEMP *P(J);
 3        IF SUBSTR(D(I),J,1)='0' THEN TEMP=TEMP * (1 - P(J));
 3        END;
 2     PROD(I)=TEMP;
 2     END;
 1
 1  DO II = 1 TO M;
 2     TEMP=0.0;
 2     DO JJ = 1 TO NBR;
 3        TEMP1=PROD(JJ);
 3        IF SUBSTR(D(JJ),II,1) ='1'
 3           THEN TEMP1=TEMP1/P(II);
 3        IF SUBSTR(D(JJ),II,1) ='0'
 3           THEN TEMP1=-(TEMP1/(1 - P(II)));
 3        IF SUBSTR(D(JJ),II,1) ='-'
 3           THEN TEMP1=0.0;
 3        TEMP= TEMP + TEMP1;
 3        END;
 2     DIF(II)= TEMP;
 2     END;
 1
 1     DO I = 1 TO M;      /* DIF ARRAY IS TRANSFERRED TO 'ARRAY' ARRAY */
 2        ARRAY(I).IND=I;  /* 'ARRAY' ARRAY IS USED FOR SORTING PURPOSE */
 2        ARRAY(I).DIF=DIF(I);
 2     END;
 1
 1  CALL SHELL_SORT;
 1  CALL MAXI;
 1
 1  /***********************************************************************/
 1  /*   SHELL SORT IS USED HERE TO SORT THE ORDER OF IMPORTANCE OF EACH    */
 1  /* COMPONENT. IT IS SORTED IN NONDECREASING ORDER.                      */
 1  /***********************************************************************/
 1  SHELL_SORT: PROC;
 2     DCL (INC,CURRENT,PREVIOUS,J,K,Y) FIXED BINARY;
 2     DCL X FLOAT;
 2     DCL (INSERTED,TRUE,FALSE) BIT(1);
 2
 2     TRUE='1'B;    FALSE='0'B;
 2     IND=M;
 2
 2     DO WHILE(INC > 1);
 3        INC=INC/2;
 3        DO J=1 TO INC;
 4           K= J + INC;
 4           DO WHILE(K <= M);
```

```
5               INSERTED=FALSE;
5               X=ARRAY(K).DIF;
5               Y=ARRAY(K).IND;
5               CURRENT=K;
5               PREVIOUS=CURRENT - INC;
5                  DO WHILE(PREVIOUS >=J & ^ INSERTED);
6                     IF X > ARRAY(PREVIOUS).DIF THEN DO;
7                        ARRAY(CURRENT).DIF=ARRAY(PREVIOUS).DIF;
7                        ARRAY(CURRENT).IND=ARRAY(PREVIOUS).IND;
7                        CURRENT=PREVIOUS;
7                        PREVIOUS=PREVIOUS - INC;
7                     END;
6                     ELSE
6                        INSERTED=TRUE;
6                   END;
5                   ARRAY(CURRENT).DIF=X;
5                   ARRAY(CURRENT).IND=Y;
5                   K=K + INC;
5               END;
4           END;
3        END;
2
2   RETURN;
2   END SHELL_SORT;
1
1   MAXI: PROC;
2   MAX= -10.0;    /*INITIAL VALUE IS LESS THAN ALL POSSIBLE DATA VALUE */
2   DO I = 1 TO M;
3      IF DIF(I) > MAX THEN DO;
4         MAX=DIF(I);
4         IND=I;
4         END;
3      END;
2
2   RETURN;
2   END MAXI;
1
1   RETURN;
1   END IMP;
```

COMMAND LINE

PLI INSDP/LIS

APPENDIX B

An example is used here to assist the user in utilizing the INSDP program. The system reliability graph is shown in Exhibit B-1. The set of minimal paths or cuts for the system graph is listed in Exhibit B-2. There are a total of 13 minimal paths for the given graph and there are 12 components in the system. Hence, N=13 and M=12. The probability of success for each component is arbitarily set at .90. It is important to remember that while inputing the original minimal terms (the S values), the single quotes must be used since INSDP regards the minimal terms as the character strings. The system may clash if quote(s) are omitted.

The following section illustrates step-by-step how the INSDP program is utilized to find the disjoint terms for the system network and its system reliability.

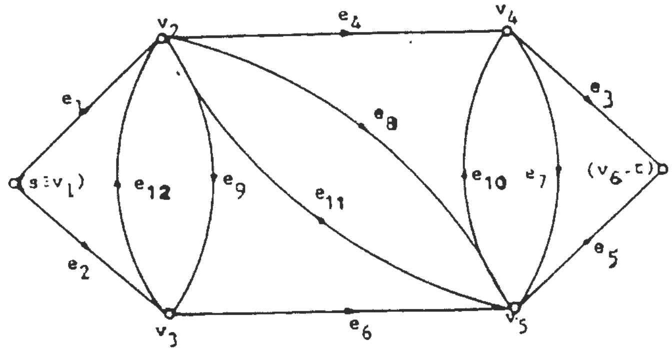

Exhibit B-1 System Reliability Graph

S	The original minimal terms											
1	1	-	1	1	-	-	-	-	-	-	-	-
2	1	-	-	-	1	-	-	1	-	-	-	-
3	-	1	-	-	1	1	-	-	-	-	-	-
4	1	-	-	1	1	-	1	-	-	-	-	-
5	1	-	1	-	-	-	-	1	-	1	-	-
6	1	-	-	-	1	1	-	-	1	-	-	-
7	-	1	1	-	-	1	-	-	-	1	-	-
8	-	1	-	-	1	-	-	1	-	-	-	1
9	-	1	1	1	-	-	-	-	-	-	-	1
10	1	-	1	-	-	1	-	-	1	1	-	-
11	-	1	1	1	-	1	-	-	-	-	1	-
12	-	1	1	-	-	-	-	1	-	1	-	1
13	-	1	-	1	1	-	-	-	-	1	-	1

Exhibit B-2 The Set of Minimal Paths or Cuts

```
OKLAHOMA STATE UNIVERSITY COMPUTER CENTER ASYNCHRONOUS COMMUNICATI
02.057
ENTER SYSTEM NAME IN CAPITAL LETTERS (IBM OR VAX)
VAX
COM

Username: U5162AA
Password:

               !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
               !!!        OKLAHOMA STATE UNIVERSITY       !!!
               !!!                 VMS 4.1                !!!
               !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

NOTE: one of the disk drives is having hardware problems.  This is
      dua2,  none of the normal user accounts are on it so users s
      not be affected.
    Last interactive login on Friday, 15-NOV-1985 14:36
%DCL-W-UNDFIL, file has not been opened by DCL - check logical nam

 DISK QUOTA INFORMATION:
  User [U5162AA] has 928 blocks used, 1072 available,
  of 2000 authorized and permitted overdraft of 1000 blocks on DUA

$ RUN INSDP




                       INTERACTIVE SDP PROGRAM
                       ***********************


                          *** MAIN MENU ***

                  1. TO ENTER THE DATA INTERACTIVELY
                  2. TO ENTER THE DATA THROUGH FILE
                  3. EXIT


       ENTER 1,2 OR 3: 1
```

INTERACTIVE DATA ENTRY MODE
YOU ARE TO ENTER THE FOLLOWING ITEMS:

THE NUMBER OF MINIMAL STATES--N
THE NUMBER OF COMPONENTS IN THE SYSTEM--M
THE SET OF PROBABILITY OF SUCCESS FOR EACH COMPONENT--P
THE SET OF MINIMAL PATHS OR MINIMAL CUTS--S

PRESS RETURN TO CONTINUE

YOU MAY ENTER DATA NOW
IF YOU MADE A MISTAKE(S), REMEMBER WHICH ONE
SINCE YOU CAN CHANGE IT LATER

```
N= 13

M= 12

P( 1) = .9

P( 2) = .9

P( 3) = .9

P( 4) = .9

P( 5) = .9

P( 6) = .9

P( 7) = .9

P( 8) = .9

P( 9) = .9

P(10) = .9

P(11) = .9

P(12) = .9
```

```
S( 1) = 11-11------'

          ***LENGTH NOT EQUAL TO M,REENTER***

          PRESS RETURN TO CONTINUE

S( 1) = '1-11---------'

S( 2) = '1---1--1----'

S( 3) = '-1--11------'

S( 4) = '1--11-1-----'

S( 5) = '1-1----1-1--'

S( 6) = '1---11--1---'

S( 7) = -11--1---1--'

          ***LENGTH NOT EQUAL TO M,REENTER***

          PRESS RETURN TO CONTINUE

S( 7) = '-11--1---1--'

S( 8) = '-1--1--1---1'

S( 9) = '-111-------1'

S(10) = 1-1--1--11--'

          ***LENGTH NOT EQUAL TO M,REENTER***

          PRESS RETURN TO CONTINUE

S(10) = '1-1-----11-'

          ***LENGTH NOT EQUAL TO M,REENTER***

          PRESS RETURN TO CONTINUE

S(10) = '1-1--1--11--'

S(11) = '-111-1----1-'

S(12) = '-11-----1-1-1'

S(13) = '-1-10-----1-1'
```

OPTIONS IN CHANGING INPUT VALUES

1. N VALUE
2. M VALUE
3. P VALUE(S)
4. S VALUE(S)
5. NO CHANGES REQUIRED,PROGRAM EXECUTION REQUESTED
6. EXIT TO MAIN MENU

ENTER 1,2,3,4,5 OR 6   4

OPTIONS IN REENTERING S VALUES

1. REENTER ALL VALUES
2. SELECTIVELY

ENTER 1 OR 2:   2

WHICH S VALUE IS TO BE ALTERED?
ENTER 1,2, ETC.

ENTER ONE NUMBER NOW   134

OUT OF RANGE, REENTER

```
PRESS RETURN TO CONTINUE




                    WHICH S VALUE IS TO BE ALTERED?
                      ENTER 1,2, ETC.

          ENTER ONE NUMBER NOW  13

S(13) = '-1-11----1-1'

                    MORE CHANGES? ENTER Y/N  N




                     OPTIONS IN CHANGING INPUT VALUES

                1. N VALUE
                2. M VALUE
                3. P VALUE(S)
                4. S VALUE(S)
                5. NO CHANGES REQUIRED,PROGRAM EXECUTION REQUESTED
                6. EXIT TO MAIN MENU

          ENTER 1,2,3,4,5 OR 6  5
```

```
**PROGRAM IS EXECUTING, PLEASE WAIT**




**EXECUTION HAS COMPLETED**

          OUTPUT OPTIONS

     1. OUTPUT THROUGH SCREEN

     2. HARD COPY

ENTER 1 OR 2   1




        SCREEN_OUTPUT OPTIONS

   1. ECHO PRINT OF INPUT DATA
   2. EQUIVALENT SET OF DISJOINT TERMS
   3. SYSTEM RELIABILITY
   4. THE IMPORTANCE OF EACH COMPONENT
   5. ALL OF THE ABOVE OPTIONS
   6. RETURN TO OPTIONS IN CHANGING INPUTS
      INTERACTIVELY
   7. RETURN TO OUTPUT OPTIONS
   8. EXIT TO MAIN MENU

ENTER 1,2,3,4,5,6, 7 OR 8   5
```

```
                    ECHO PRINT OF INPUT DATA

                         N VALUE IS                 13

                         M VALUE IS                 12

          PRESS RETURN TO CONTINUE


                    ORIGINAL MINIMAL TERMS

1-11---------
1---1--1-----
-1--11-------
1--11-1------
1-1----1-1---
1---11--1----
-11--1---1---
-1--1--1---1
-111--------1
1-1--1--11---
-111-1----1-
-11----1-1-1
-1-11----1-1

          PRESS RETURN TO CONTINUE


                    ECHO PRINT OF PROBABILITY VALUES

0.900
0.900
0.900
0.900
0.900
0.900
0.900
0.900
0.900
0.900
0.900
0.900

          PRESS RETURN TO CONTINUE
```

EQUIVALENT OF DISJOINT TERMS

```
1-11--------
1-0-1--1----
1-101--1----
01--11------
110-11-0----
111011-0----
10011-10----
11011010----
1-100--1-1--
100011-01---
100111001---
101011-01---
011-01---1--
111001-0-1--
01--10-1---1
011100-----1
011101---0-1
011110-0---1
```

PRESS RETURN TO CONTINUE

```
101001-011--
011101---010
011000-1-1-1
010110-0-1-1
11011000-1-1
```

PRESS RETURN TO CONTINUE

SYSTEM RELIABILITY 0.97724

PRESS RETURN TO CONTINUE

THE IMPORANCE OF EACH COMPONENT IN SORTED ORDER

COMPONENT NO.	IMPORTANCE CALCULATION
5	0.107639
1	0.107639
2	0.098737
3	0.098147
4	0.019276
6	0.018686
8	0.010747
10	0.008997
12	0.008931
9	0.000860
7	0.000263
11	0.000066

PRESS RETURN TO CONTINUE

THE MOST IMPORTANT COMPONENT IS: 5

PRESS RETURN TO CONTINUE

```
           SCREEN_OUTPUT OPTIONS

       1. ECHO PRINT OF INPUT DATA
       2. EQUIVALENT SET OF DISJOINT TERMS
       3. SYSTEM RELIABILITY
       4. THE IMPORTANCE OF EACH COMPONENT
       5. ALL OF THE ABOVE OPTIONS
       6. RETURN TO OPTIONS IN CHANGING INPUTS
          INTERACTIVELY
       7. RETURN TO OUTPUT OPTIONS
       8. EXIT TO MAIN MENU

  ENTER 1,2,3,4,5,6, 7 OR 8  7




        **EXECUTION HAS COMPLETED**


            OUTPUT OPTIONS

        1. OUTPUT THROUGH SCREEN

        2. HARD COPY

 ENTER 1 OR 2  2




   YOUR HARD COPY IS READY. YOU WILL RECEIVE AN ECHO
   PRINT OF INPUT DATA, EQUIVALENT OF DISJOINT TERMS.
   SYSTEM RELIABILITY, THE IMPORTANCE OF EACH COMPO-
   NENT, AND THE MOST IMPORTANT COMPONENT OF SYSTEM.

   OUTPUT FILE NAME IS OUTFSDP.DAT

PRESS RETURN TO CONTINUE
```

```
          IF YOU WISH TO RECEIVE THE HARD COPY, YOU SHOULD
          EXIT AT THE MAIN MENU. THEN USE THE APPROPRIATE
          PRINT COMMAND, SUCH AS $ PRINT OUTFSDP.DAT/Q=BUS


    PRESS RETURN TO CONTINUE


                     *** MAIN MENU ***

           1. TO ENTER THE DATA INTERACTIVELY
           2. TO ENTER THE DATA THROUGH FILE
           3. EXIT


       ENTER 1,2 OR 3: 3
$ PRIN OUTFSDP.DAT/Q=UCC1
Job OUTFSDP (queue UCC1, entry 1092) started on UCC1
$ LO
  U5162AA       logged out at 15-NOV-1985 15:06:49.13
DISC
```

APPENDIX C

INSTRAP PROGRAM LISTING

```
    INSTRAP: PROCEDURE OPTIONS(MAIN);
1 /**********************************************************************/
1 /*     MAUNG M. LAY              INTERACTIVE SIMPLIFIED TOPOLOGICAL     */
1 /*                                   RELIABILITY ANALYSIS PROGRAM       */
1 /*     OCTOBER 1985                    (VAX COMPUTER SYSTEM)            */
1 /*                                                                    */
1 /*     UPGRADED FROM DAVID V. FISCHER'S(1984) STRAP1 PROGRAM          */
1 /*                                                                    */
1 /*         FISCHER'S STRAP1 IS BASED ON THE WORK OF SATYANARAYANA AND  */
1 /*     PRABHAKAR (S&R)(1978). HOWEVER, IT IS A SIMPLIFIED VERSION OF S&R */
1 /*     ORIGINAL WORK. INSTEAD OF BUILDING AN ACTUAL SEARCH TREE AS SUG- */
1 /*     GESTED BY S&R, PL/I FEATURE-- RECURSIVE ROUTINES ARE USED EXTEN- */
1 /*     SIVELY TO 'ACT' AS A TREE. THE FOLLOWING PARAGRAPHS DESCRIBED   */
1 /*     THE WORKING LOGIC BEHIND FISCHER'S ORIGINAL WORK.              */
1 /*                                                                    */
1 /*       RULE ONE : RULE ONE STARTS BY CALLING A SUBROUTINE WITHIN IT  */
1 /*     CALLED SEARCH. SEARCH TRAVERSES THE GRAPH MUCH THE SAME WAY THAT */
1 /*     THE MINIMAL PATHS WERE FOUND EXCEPT THAT WHEN SEARCH ENCOUNTERS  */
1 /*     A VERTEX WHICH HAS BEEN PREVIOUSLY VISITED, IT RECOGNIZES THIS AS */
1 /*     A CYCLE. DURING THE SEARCH PROCESS, SEARCH KEEPS TRACK OF THE VER-*/
1 /*     TICES THAT HAVE BEEN VISITED BOTH IS UNA (UNAVAILABLE) AND IN   */
1 /*     VERTX (THE VERTICES IN THE ORDER IN WHICH THEY HAVE BEEN VISITED).*/
1 /*     BY KNOWING THE ORDER IN WHICH THE VERTICES HAVE BEEN VISITED, WHEN*/
1 /*     SEARCH FINDS A CYCLE, VISITS A VERTEX WHICH HAS BEEN PREVIOUSLY  */
1 /*     VISITED, THE CYCLE CAN BE FOUND BY MEANS OF THE PREVIOUS NODES.  */
1 /*     THE ARRAY VERTX IS TRAVERSED FROM THE FIRST CELL UNTIL THE CURRENT*/
1 /*     VERTEX IS FOUND. ALL VERTICES BETWEEN THESE TWO VERTICES WILL BE */
1 /*     VERTICES IN THE CYCLE. THIS VERTEX AND THE VERTEX IN THE NEXT CELL*/
1 /*     WILL BE THE BEGINNING AND ENDING VERTICES OF THE FIRST EDGE OF THE*/
1 /*     CYCLE. THIS EDGE IS DELETED BY CALLING REMOVE AND RULE_ONE IS    */
1 /*     CALLED AGAIN RECURSIVELY. THE ACT OF DELETING THE EDGE OF THE CY- */
1 /*     CLE MAY DISTURB THE PREVIOUS SEARCH. BY CALLING RULE_ONE, THE    */
1 /*     SEARCH OF THE GRAPH IS INITIALIZED WITH THIS EDGE DELETED THUS   */
1 /*     STARTING THE SEARCH OVER WITH A NEW SUBGRAPH. PL/I CREATES       */
1 /*     'ENVIRONMENTS' IN WHICH THE PROGRAM OPERATES. BY CALLING RULE_ONE */
1 /*     RECURSIVELY, THE PROGRAM CREATES A 'SUB-ENVIRONMENT' IN WHICH TO  */
1 /*     PROCESS THE NEW SUBGRAPH. WHEN THE PROGRAM RETURNS, IT WILL HAVE  */
1 /*     THE SAME PARAMETERS WHICH EXISTED AT THE TIME OF THE CALL. THIS  */
1 /*     MEANS THAT THE CURRENT SEARCH WILL STILL BE IN PROGRESS. THE EDGE */
1 /*     WHICH HAS DELETED CAN BE RESTORED AND THE NEXT EDGE CAN BE DELETED*/
1 /*     WITH A NEW CALL TO RULE_ONE. BECAUSE OF THE NATURE OF A RECURSIVE */
1 /*     PROCEDURE, THE PROGRAM CAN ACT EXACTLY LIKE THE SEARCH TREE WITH- */
1 /*     OUT HAVING TO BUILD  AND STORE THE TREE. IN THIS PROGRAM, CALLING */
1 /*     A RULE IS ANALOGOUS TO GOING DOWN THE TREE AND A RETURN IS ANALO- */
1 /*     GOUS TO BACKTRACKING. REMEMBER THAT RECURSIVE ROUTINE ACTS AS A   */
1 /*     STACKING FUNCTION. WHEN RULE_ONE IS ABLE TO SEARCH THE ENTIRE TREE*/
1 /*     WITHOUT FINDING THE CYCLE, IT CALLS RULE_TWO.                    */
1 /*                                                                    */
1 /*       RULE TWO : SINCE RULE_TWO CAN ONLY BE REACHED BY FIRST GOING   */
1 /*     THROUGH RULE_ONE, THE SUBGRAPH MUST BE ACYCLIC. IF AN ACYCLIC    */
1 /*     GRAPH HAS ALL INTERNAL VERTICES WITH INDEGREE AND OUTDEGREE GREAT-*/
1 /*     ER THAN ZERO IT WILL BE P-ACYCLIC. THIS IS TRUE BECAUSE IN ORDER  */
1 /*     FOR EVERY VERTEX TO HAVE AN INDEGREE AND AN OUTDEGREE, IT MUST   */
1 /*     HAVE AN EDGE ON EITHER SIDE OF IT. SINCE AN EDGE MUST HAVE A VER- */
1 /*     TEX ON EITHER END, THAT VERTEX MUST ALSO HAVE AN EDGE ON EITHER  */
```

```
/*   SIDE OF IT. THE ONLY WAY THAT ONE VERTEX CAN HAVE AN INDEGREE AND */
/*   AN OUTDEGREE WITHOUT ENTERING THE TERMINAL OR LEAVING THE START,  */
/*   IS FOR A CYCLE TO BOTH ENTER AND LEAVE THAT VERTEX. THE GIVEN SUB-*/
/*   GRAPH MAY NOT BE ACYCLIC BECAUSE THE DELETION OF EDGES MAY CAUSE A*/
/*   PORTION OF THE GRAPH, WHICH MAY CONTAIN A CYCLE, TO BE UNREACHABLE*/
/*   BY SEARCH. DURING SEARCH, THE EDGES WHICH ARE TRAVERSED DURING THE*/
/*   SEARCH ARE PUT IN ARRAY CALLED USED. IF THERE ARE ANY EDGES IN THE*/
/*   GRAPH WHICH HAVE NOT BEEN SEARCHED, AND ARE NOT IN USED, THEY WILL*/
/*   BE DELETED. IF ANY OF THESE EDGES ARE IN THE WEIGHT, THE PROCEDURE*/
/*   RETURNS SINCE THE SUBGRAPH CANNOT BE MADE P-ACYCLIC. ALL OTHER    */
/*   EDGES MUST BE 'HANGING EDGES' AND WILL HAVE AN INDEGREE OR OUTDE- */
/*   GREE OF ONE OF THEIR BOUNDING VERTICES EQUAL TO ZERO IF ANY OF    */
/*   THESE IS IN THE WEIGHT, THE PROCEDURE RETURNS, RULE_TWO THEN CALLS*/
/*   P_GRAPH WHICH CHECKS THE SUBGRAPH TO SEE IF IT IS P-ACYCLIC. IF   */
/*   THE SUBGRAPH IS P-ACYCLIC, RULE_THREE IS CALLED.                  */
/*                                                                     */
/*     RULE THREE : RULE THREE SEARCHES THE ARRAY GRAPH FOR ENTRIES    */
/*   WHICH ARE GREATER THAN ZERO AND DELETES THESE ENTRIES ONE AT A    */
/*   TIME. WHEN THE EDGE IS DELETED, IT CALLS P_GRAPH TO CHECK FOR A   */
/*   P-ACYCLIC GRAPH. IF THE GRAPH IS P-ACYCLIC, THE EDGES, ALONG WITH */
/*   ALL EDGES WHICH ARE IN A SEQUENCE WITH IT, ARE STORED IN THE ARRAY*/
/*   CHILDREN. THE EDGE IS THEN RESTORED, ALONG WITH ALL OTHER EDGES   */
/*   IN THE SEQUENCE, AND THE NEXT ENTRY IS DELETED. THIS FINDS ALL    */
/*   NEUTRAL SEQUENCES OF THE GIVEN GRAPH. SINCE THE CURRENT GRAPH IS  */
/*   P-ACYCLIC, SUBROUTINE RELIABILITY IS CALLED. ONCE ALL NEUTRAL SE- */
/*   QUENCES ARE FOUND, RULE FOUR IS CALLED AND THE ARRAY CHILDREN IS  */
/*   PASSED TO IT.                                                     */
/*                                                                     */
/*     RULE FOUR : RULE_FOUR TAKES THE ARRAY CHILDREN AND SEARCHES THIS*/
/*   ARRAY FOR AN ENTRY GREATER THAN ZERO. THIS ENTRY WILL BE THE BE-  */
/*   GINNING VERTEX OF THE CHILD. RULE_FOUR THEN SEARCHES THIS COLUMN  */
/*   OF THE ARRAY GRAPH FOR THE ENTRY I. RULE_FOUR THEN DELETES THIS   */
/*   ENTRY, AND SETS CHILDREN(I) TO ZERO, AND CALLS ITSELF RECURSIVELY.*/
/*   WHEN RULE_FOUR RETURNS IT WILL CONTINUE THE SEARCH FROM THIS POINT*/
/*   IN THE ARRAY CHILDREN.                                            */
/*********************************************************************/

/*********************************************************************/
/*   D_FLAG: DATA FLAG, 1=ON 0=OFF                                     */
/*   L_FLAG: LINKSET FLAG, 1=ON 0=OFF                                  */
/*   T_FLAG: TRACE FLAG, 1=ON 0=OFF                                    */
/*   I_FLAG: IMPORTANCE CALCULATION FLAG, 1=ON 0=OFF                   */
/*   S_FLAG: SCREEN OUTPUT OPTION FLAG, 1=ON 0=OFF                     */
/*   H_FLAG: HARD COPY OUTPUT OPTION FLAG, 1=ON 0=OFF                  */
/*   R_FLAG: RELIABILITY CALCULATION FLAG, 1=ON 0=OFF                  */
/*   BOTH_OUT: INDICATES WHEN BOTH S_FLAG AND H_FLAG ARE ACTIVATED     */
/*   COMD: CHARACTER VARIABLE USED TO ACCEPT INPUT FROM TERMINAL       */
/*   N: NUMBER OF VERTICES                                             */
/*   M: NUMBER OF EDGES                                                */
/*   START: THE NUMBER OF START VERTEX                                 */
/*   TERMINAL: THE NUMBER OF TERMINAL VERTEX                           */
/*                                                                     */
/*********************************************************************/

DCL (N,M) FIXED BINARY;
DCL COMD CHAR(1);
DCL (D_FLAG,L_FLAG,T_FLAG,I_FLAG,S_FLAG,H_FLAG,R_FLAG,BOTH_OUT)FIXED BINARY;
```

```
1 DCL INFSTRAP FILE INPUT;
1 DCL OUTFILE PRINT FILE;
1
1 MAIN_MENU:D_FLAG=0;
1   L_FLAG=0;
1   T_FLAG=0;
1   I_FLAG=0;
1   S_FLAG=0;
1   H_FLAG=0;
1   R_FLAG=0;
1   BOTH_OUT=0;
1
1   PUT PAGE;
1   PUT SKIP EDIT('INTERACTIVE STRAP PROGRAM')(X(23),A);
1   PUT SKIP EDIT('*************************')(X(23),A);
1   PUT SKIP(3) EDIT('***MAIN MENU***')(X(26),A);
1   PUT SKIP(2) EDIT ('1. TO ENTER THE DATA INTERACTIVELY')(X(17),A);
1   PUT SKIP EDIT ('2. TO ENTER THE DATA THROUGH FILE')(X(17),A);
1   PUT SKIP EDIT ('3. EXIT')(X(17),A);
1   PUT SKIP(3) EDIT('ENTER 1,2 OR 3    ')(X(7),A);
1   GET LIST(COMD);
1
1   IF COMD='1' THEN GO TO INTERACT_DATA;
1   IF COMD='2' THEN GO TO LOADFILE_DATA;
1   IF COMD='3' THEN STOP;
1
1 ERROR1: PUT SKIP(3) EDIT('***OUT OF RANGE, REENTER***')(X(10),A);
1   CALL ADVANCE;
1   GO TO MAIN_MENU;
1
1 INTERACT_DATA: PUT PAGE;
1   PUT SKIP(2) EDIT('INTERACTIVE DATA ENTRY MODE')(X(20),A);
1   PUT SKIP EDIT('YOU ARE TO ENTER THE FOLLOWING DATA:')(X(17),A);
1   PUT SKIP(2) EDIT('THE NUMBER OF VERTICES - N')(X(14),A);
1   PUT SKIP EDIT('THE NUMBER OF EDGES - M')(X(14),A);
1   PUT SKIP EDIT('THE NUMBER OF START VERTEX - START')(X(14),A);
1   PUT SKIP EDIT('THE NUMBER OF TERMINAL VERTEX - TERMINAL')(X(14),A);
1   PUT SKIP(2) EDIT('ALSO REQUIRED TO INPUT ---')(X(14),A);
1   PUT SKIP EDIT('THE INFORMATION ABOUT THE EDGES, SUCH AS')(X(14),A);
1   PUT SKIP EDIT('THE BEGINNING NODE, THE ENDING NODE,')(X(14),A);
1   PUT SKIP EDIT('THE DESIRED STATUS 1=DIRECTED, 0=UNDIRECTED')(X(14),A);
1   PUT SKIP EDIT('THE RELIABILITY OF THE EDGE')(X(14),A);
1   CALL ADVANCE;
1
1 RECEIVE_DATA: PUT PAGE;
1   PUT SKIP(3) EDIT('YOU MAY ENTER DATA NOW')(X(20),A);
1   PUT SKIP EDIT('IF YOU MADE A MISTAKE(S), REMEMBER WHICH ONE')(X(14),A);
1   PUT SKIP EDIT('SINCE YOU COULD CHANGE IT LATER')(X(18),A);
1
1   PUT SKIP(3) EDIT('N= ')(X(7),A);
1   GET LIST(N);
1   PUT SKIP(3) EDIT('M= ')(X(7),A);
1   GET LIST(M);
1
1 CHANGE_INPUT1: PUT PAGE;
1   PUT SKIP(3) EDIT('REQUEST TO CHANGE')(X(28),A);
1   PUT SKIP(3) EDIT('1. N VALUE')(X(23),A);
```

```
1     PUT SKIP EDIT('2. M VALUE')(X(23),A);
1     PUT SKIP EDIT('3. NO CHANGES, CONTINUE')(X(23),A);
1     PUT SKIP(3) EDIT('ENTER 1,2 OR 3  ')(X(17),A);
1     GET LIST(COMD);
1
1     IF COMD='1' THEN GO TO NVALUE;
1     IF COMD='2' THEN GO TO MVALUE;
1     IF COMD='3' THEN GO TO CONT;
1
1  ERROR2:PUT SKIP(3) EDIT('***OUT OF RANGE, REENTER***')(X(15),A);
1     CALL ADVANCE;
1     GO TO CHANGE_INPUT1;
1
1  NVALUE: PUT PAGE;
1     PUT SKIP(3) EDIT('N= ',N)(X(7),A,F(3));
1     PUT SKIP(2) EDIT('N=    ')(X(7),A);
1     GET LIST(N);
1     GO TO CHANGE_INPUT1;
1
1  MVALUE: PUT PAGE;
1     PUT SKIP(3) EDIT('M= ',M)(X(7),A,F(3));
1     PUT SKIP(2) EDIT('M=    ')(X(7),A);
1     GET LIST(M);
1     GO TO CHANGE_INPUT1;
1
1  ADVANCE: PROC;
2     DCL ANY CHAR(80) VAR;
2     PUT SKIP(3) EDIT('PRESS RETURN TO CONTINUE')(X(10),A);
2     GET LIST(ANY);
2     RETURN;
2     END ADVANCE;
1
1  LOADFILE_DATA: PUT PAGE;
1     PUT SKIP EDIT('YOUR DATA FILE SHOULD BE ALREADY CREATED')(X(20),A);
1     PUT SKIP EDIT('IT MUST BE NAMED INFSTRAP.DAT')(X(20),A);
1     PUT SKIP EDIT('THE DATA FILE SHOULD BE IN THE FOLLOWING FORMAT')
1                 (X(20),A);
1     PUT SKIP(2) EDIT('N,M,S,T,')(X(20),A);
1     PUT SKIP EDIT    ('B,E,D,R,')(X(20),A);
1     PUT SKIP EDIT    ('    .     ')(X(20),A);
1     PUT SKIP EDIT    ('    .     ')(X(20),A);
1     PUT SKIP EDIT    ('B,E,D,R ')(X(20),A);
1     PUT SKIP(2) EDIT('NOTE:USE COMMA TO SEPARATE THE VALUES, AND BETWEEN LINES')
1                    (X(20),A);
1     PUT SKIP(2) EDIT('IF NOT, TYPE IN E TO EXIT TO MAIN MENU  ')(X(20),A);
1     PUT SKIP    EDIT('IF YES, TYPE IN C TO CONTINUE  ')(X(20),A);
1
1     GET LIST(COMD);
1     IF COMD='E' THEN GO TO MAIN_MENU;
1     IF COMD='C' THEN GO TO READFILE;
1
1  ERROR10: PUT SKIP(3) EDIT('***INPUT ERROR DETECTED,REENTER***')(X(15),A);
1     CALL ADVANCE;
1     GO TO LOADFILE_DATA;
1
1  READFILE: R_FLAG=1;
1     OPEN FILE(INFSTRAP) INPUT;
```

```
  1   GET FILE(INFSTRAP) LIST(N,M);
  1
! 1 /*                                                                  */
  1
  1 CONT: PUT PAGE;
  1 A: BEGIN;
  2   DCL (IN,OUT,GRAPH(0:N,0:N),Z,O,MO,SIGN,N1,I,J,K,D_NUM,START,
  2         TERMINAL,DATA(2*M,3),LINE,LLINE) FIXED BINARY;
  2   DCL (RELIA,GRAPH_REL,EDGE_REL(2*M),REL(2*M)) FLOAT;
  2   DCL COMD CHAR(1);
  2   GRAPH =0;
  2   D_NUM =0;
  2   RELIA =1.0;
  2   GRAPH_REL =0.0;
  2   MO = -1;
  2   Z  = 0;
  2   N1 = 1;
  2   O  = 1;
  2   LINE=0;
  2   LLINE=0;
  2
  2   IF R_FLAG=1 THEN DO;
  3     GET FILE(INFSTRAP) LIST(START,TERMINAL);
  3
  3     DO I =1 TO M;
  4     GET FILE(INFSTRAP) LIST(DATA(I,1),DATA(I,2),DATA(I,3),REL(I));
  4     END;
  3     CLOSE FILE(INFSTRAP);
  3     R_FLAG=0;
  3     GO TO CONT3;
  3   END;
  2
  2   PUT SKIP(3) EDIT('START= ')(X(7),A);
  2   GET LIST(START);
  2   PUT SKIP(3) EDIT('TERMINAL= ')(X(7),A);
  2   GET LIST(TERMINAL);
  2
  2 CHANGE_INPUT2: PUT PAGE;
  2   PUT SKIP(3) EDIT('REQUEST TO CHANGE')(X(28),A);
  2   PUT SKIP(3) EDIT('1. START VALUE')(X(23),A);
  2   PUT SKIP EDIT   ('2. TERMINAL VALUE')(X(23),A);
  2   PUT SKIP EDIT   ('3. NO CHANGES, CONTINUE')(X(23),A);
  2   PUT SKIP(3) EDIT('ENTER 1,2 OR 3    ')(X(17),A);
  2   GET LIST(COMD);
  2
  2   IF COMD='1' THEN GO TO SVALUE;
  2   IF COMD='2' THEN GO TO TVALUE;
  2   IF COMD='3' THEN GO TO CONT2;
  2
  2 ERROR3:PUT SKIP(3) EDIT('***OUT OF RANGE, REENTER***')(X(15),A);
  2   CALL ADVANCE1;
  2   GO TO CHANGE_INPUT2;
  2
  2 SVALUE: PUT PAGE;
  2   PUT SKIP(3) EDIT('START= ',START)(X(7),A,F(3));
  2   PUT SKIP(2) EDIT('START=    ')(X(7),A);
  2   GET LIST(START);
```

```
  GO TO CHANGE_INPUT2;

TVALUE: PUT PAGE;
  PUT SKIP(3) EDIT('TERMINAL= ',TERMINAL)(X(7),A,F(3));
  PUT SKIP(2) EDIT('TERMINAL=    ')(X(7),A);
  GET LIST(TERMINAL);
  GO TO CHANGE_INPUT2;

CONT2: CALL DATA_IN;

CONT3: PUT PAGE;
PUT SKIP(3) EDIT('OUTPUT SELECTIONS')(X(30),A);
  PUT SKIP(3) EDIT('1. ECHO PRINT,LINKSET, AND RELIABILITY')(X(18),A);
  PUT SKIP EDIT   ('2. OPTION 1. AND IMPORTANCE CALCULATION')(X(18),A);
  PUT SKIP EDIT   ('3. TRACE ROUTINE')(X(18),A);
  PUT SKIP EDIT   ('4. ALL OF THE ABOVE')(X(18),A);
  PUT SKIP(3) EDIT('ENTER 1,2,3, OR 4    ')(X(14),A);
  GET LIST(COMD);

  IF COMD='1' THEN GO TO OPT1;
  IF COMD='2' THEN GO TO OPT2;
  IF COMD='3' THEN GO TO OPT3;
  IF COMD='4' THEN GO TO OPT4;

ERROR8:PUT SKIP(3) EDIT('***OUT OF RANGE,REENTER***')(X(15),A);
  CALL ADVANCE1;
  GO TO CONT3;

OPT1: D_FLAG=1;
  L_FLAG=1;
  GO TO CONT4;

OPT2: I_FLAG=1; D_FLAG=1; L_FLAG=1;
  GO TO CONT4;

OPT3: T_FLAG=1; L_FLAG=1;
  GO TO CONT4;

OPT4: D_FLAG=1;
  L_FLAG=1;
  T_FLAG=1;
  I_FLAG=1;

CONT4: PUT PAGE;
  PUT SKIP(3) EDIT('OUTPUT MODE')(X(30),A);
  PUT SKIP(3) EDIT('1. SCREEN OUTPUT REQUESTED')(X(20),A);
  PUT SKIP EDIT   ('2. HARD COPY REQUESTED')(X(20),A);
  PUT SKIP EDIT   ('3. ALL OF THE ABOVE')(X(20),A);
  PUT SKIP(3) EDIT('ENTER 1,2, OR 3    ')(X(15),A);
  GET LIST(COMD);

  IF COMD='1' THEN GO TO OUTMODE1;
  IF COMD='2' THEN GO TO OUTMODE2;
  IF COMD='3' THEN GO TO OUTMODE3;

ERROR9:PUT SKIP(3) EDIT('***OUT OF RANGE,REENTER***')(X(15),A);
  CALL ADVANCE1;
```

```
  GO TO CONT4;

OUTMODE1: S_FLAG=1;
  GO TO OUT_1;

OUTMODE2: H_FLAG=1;
  GO TO OUT_1;

OUTMODE3:BOTH_OUT=1;

OUT_1:PUT PAGE;
  IF (S_FLAG=1 | BOTH_OUT=1) THEN DO;
   IF D_FLAG=1 THEN DO;
    PUT SKIP(3) EDIT('ECHO PRINT OF INPUT DATA')(X(30),A);
    PUT SKIP(3) EDIT('NO. OF VERTICES =  ',N)(X(15),A,F(3));
    PUT SKIP(2) EDIT('NO. OF EDGES    =  ',M)(X(15),A,F(3));
    PUT SKIP(3) EDIT('START VERTEX    =  ',START)(X(15),A,F(3));
    PUT SKIP(2) EDIT('TERMINAL VERTEX =  ',TERMINAL)(X(15),A,F(3));

  CALL ADVANCE1;

  PUT PAGE;

  CALL HEADER;

  DO J = 1 TO M BY 15;
     DO I =J TO J+14;
       IF I<=M THEN PUT SKIP EDIT(I,'.',DATA(I,1),DATA(I,2),DATA(I,3),REL(I))
                (X(5),F(3),A,X(8),F(4),X(5),F(4),X(6),F(4),X(12),F(5,3));
     END;
     CALL ADVANCE1;
     IF I<=M THEN CALL HEADER;
  END;
 END;
END;

IF (H_FLAG=1 | BOTH_OUT=1) THEN DO;
  IF D_FLAG=1 THEN DO;
     OPEN FILE(OUTFILE) TITLE('OFSTRAP.DAT');
     PUT FILE(OUTFILE) SKIP EDIT('******INSTRAP******')(X(35),A);
     PUT FILE(OUTFILE) SKIP(3) EDIT('ECHO PRINT OF INPUT DATA')(X(25),A);
     PUT FILE(OUTFILE) SKIP(3) EDIT('NO. OF VERTICES  =  ',N)(X(20),A,F(3));
     PUT FILE(OUTFILE) SKIP(2) EDIT('NO. OF EDGES     =  ',M)(X(20),A,F(3));
     PUT FILE(OUTFILE) SKIP(2) EDIT('START VERTEX     =  ',START)(X(20),A,F(3));
     PUT FILE(OUTFILE) SKIP(2) EDIT('TERMINAL VERTEX  =  ',TERMINAL)
                               (X(20),A,F(3));

     PUT FILE(OUTFILE) PAGE;
     PUT FILE(OUTFILE) SKIP EDIT('EDGE NO.','FROM','TO ','DIRECTION',
                       'RELIABILITY')(X(5),A,X(5),A,X(5),A,X(5),A,X(5),A);
     PUT FILE(OUTFILE) SKIP(2);
     DO I = 1 TO M;
        PUT FILE(OUTFILE) SKIP EDIT(I,'.',DATA(I,1),DATA(I,2),DATA(I,3),REL(I))
          (X(6),F(3),A,X(8),F(3),X(5),F(3),X(9),F(3),X(15),F(5,3));
     END;
  END;
END;
```

```
2
2  /***************************************************************************/
2  /*    THIS SECTION READS THE INPUT DATA INTO THE ARRAY 'GRAPH'. ALSO       */
2  /* IF UNDIRECTED EDGE IS FOUND, TRANSFORMATION TO DIRECTED EDGES IS        */
2  /* CARRIED OUT.                                                            */
2  /***************************************************************************/
2
2     K=0;
2     DO I = 1 TO M;
3        K=K + 1;
3        J=GRAPH(DATA(K,1),DATA(K,2));
3        IF (J ^= 0) THEN DO;
4           EDGE_REL(J) = EDGE_REL(J) + REL(K) - EDGE_REL(J) * REL(K);
4           K= K - 1;
4           M= M - 1;
4        END;
3
3        ELSE DO;
4           GRAPH(DATA(K,1),DATA(K,2)) = K;
4           EDGE_REL(K)= REL(K);
4        END;
3
3        IF (DATA(K,3) = 0) THEN
3           IF (GRAPH(DATA(K,2),DATA(K,1)) ^= 0) THEN DO;
4           J = GRAPH(DATA(K,2),DATA(K,1));
4           EDGE_REL(J)= EDGE_REL(J) + REL(K) - EDGE_REL(J) * REL(K);
4        END;
3
3        ELSE DO;
4           M= M + 1;
4           GRAPH(DATA(K,2),DATA(K,1)) = M;
4           EDGE_REL(M) = REL(K);
4           DATA(M,2)= DATA(K,1);
4           DATA(M,1)= DATA(K,2);
4           DATA(M,3)= DATA(K,3);
4        END;
3     END;
2
2  /***************************************************************************/
2
2     B: BEGIN;
3     DCL (WEIGHT(M),P_FLAG,UNA(N),BIT1,BIT0) BIT(1);
3     DCL D_SEQ(M + 1,2) BINARY FIXED;
3     DCL IMPORTANCE(M) FLOAT;
3
3     D_SEQ=0;
3     IMPORTANCE=0.0;
3     BIT1='1'B;
3     BIT0='0'B;
3     WEIGHT=BIT0;
3     DO I =1 TO M;
4        IF (EDGE_REL(I) = -1) THEN WEIGHT(I)=BIT1;
4        ELSE RELIA=RELIA * EDGE_REL(I);
4     END;
3
3     IF (S_FLAG=1|BOTH_OUT=1) THEN CALL HEADER1;
3     IF (H_FLAG=1|BOTH_OUT=1) THEN CALL HEADER2;
```

```
/************************************************************************/
/*    THE FOLLOWING ROUTINE SEARCHES THE GRAPH AND COUNTS THE NONZERO    */
/* ENTRIES. THE TOTAL FOR EACH ROW AND COLUMN IS THEN PUT IN THE ZERO    */
/* CELLS FOR EACH ROW AND COLUMN.                                        */
/************************************************************************/

   DO I = 1 TO N;
     DO J = 1 TO N;
       IF (GRAPH(I,J) > 0) THEN DO;
         GRAPH(I,0)=GRAPH(I,0) + 1;
         GRAPH(0,J)=GRAPH(0,J) + 1;
       END;
     END;
   END;

/************************************************************************/
/*   THE FOLLOWING DO LOOP CHECKS TO SEE IF THERE ARE ANY EDGES ENTERING*/
/* THE STARTING NODE, LEAVING THE TERMINAL NODE, OR, WHICH LEAVE AND    */
/* ENTER A SINGLE NODE. SUCH EDGES WILL BE IN THE START COLUMN, THE     */
/* TERMINAL ROW OR THE DIAGONAL.                                         */
/* THESE TYPES OF EDGES WILL CONTRIBUTE NOTHING TO THE RELIABILITY AND  */
/* MUST BE REMOVED ALONG WITH ANY SEQUENCES WHICH CONTAIN THEM. THIS    */
/* ROUTINE ALSO REMOVES ALL 'HANGING' EDGES.                            */
/************************************************************************/

   DO I = 1 TO N;
     IF (GRAPH(I,START) > 0) THEN CALL REMOVE(I,START,Z,BITO);
     IF (GRAPH(TERMINAL,I) > 0) THEN CALL REMOVE(TERMINAL,I,Z,BITO);
     IF (GRAPH(I,I) > 0) THEN CALL REMOVE(I,I,Z,BITO);
     IF (I ^= START) THEN IF (GRAPH(0,I)=0) THEN IF (GRAPH(I,0) > 0)
       THEN DO J = 1 TO N;
         IF (GRAPH(I,J) > 0) THEN CALL REMOVE(I,J,Z,BITO);
       END;
     IF (I ^= TERMINAL) THEN IF (GRAPH(I,0)=0) THEN IF (GRAPH(0,I)>0)
       THEN DO J =1 TO N;
         IF (GRAPH(J,I) > 0) THEN CALL REMOVE(J,I,Z,BITO);
       END;
    END;

   CALL RULE_ONE;

   IF(S_FLAG=1|BOTH_OUT=1) THEN DO;
   PUT SKIP EDIT('__________','GRAPH RELIABILITY',GRAPH_REL)
                (COL(60),A,COL(1),A,COL(60),F(8,6));
   CALL ADVANCE1;
   PUT PAGE;

     IF (I_FLAG=1) THEN DO;
      PUT SKIP EDIT('EDGE IMPORTANCE')(A);
      PUT SKIP(2);
      DO J = 1 TO M BY 15;
        DO I=J TO J+14;
        IF I<=M THEN PUT SKIP EDIT('IMPORTANCE(',I,') =',IMPORTANCE(I))
                        (A,F(2),A,F(9,6));
        END;
        CALL ADVANCE1;
        IF I<=M THEN PUT SKIP EDIT('EDGE IMPORTANCE')(A);
```

```
      END;
     END;
   END;

   IF(H_FLAG=1|BOTH_OUT=1) THEN DO;
     PUT FILE(OUTFILE) SKIP EDIT('__________','GRAPH RELIABILITY',GRAPH_REL)
                       (COL(60),A,COL(20),A,COL(60),F(8,6));
     PUT FILE(OUTFILE) PAGE;
     IF (I_FLAG=1) THEN DO;
       PUT FILE(OUTFILE) SKIP EDIT('EDGE IMPORTANCE')(A);
       PUT FILE(OUTFILE) SKIP(2);
       DO I =1 TO M;
         PUT FILE(OUTFILE) SKIP EDIT('IMPORTANCE(',I,') =',IMPORTANCE(I))
                                   (A,F(2),A,F(9,6));
       END;
     END;
    END;

   IF (H_FLAG=1|BOTH_OUT=1) THEN DO;
     CLOSE FILE(OUTFILE);
     PUT PAGE;
     PUT SKIP(3) EDIT('YOUR HARD COPY IS READY')(X(20),A);
     PUT SKIP(2) EDIT('OUTPUT FILE NAME IS OFSTRAP.DAT')(X(20),A);
     CALL ADVANCE1;
     PUT PAGE;
     PUT SKIP(3) EDIT('IF YOU WISH TO RECEIVE THE HARD COPY, YOU SHOULD')
                 (X(20),A);
     PUT SKIP EDIT('EXIT AT THE MAIN MENU, THEN USE THE APPROPRIATE')(X(20),A);
     PUT SKIP EDIT('PRINT COMMAND, SUCH AS $PRINT OFSTRAP.DAT/Q=QUS')(X(20),A);
     CALL ADVANCE1;
   END;

     GO TO MAIN_MENU;

/*************************************************************************/
/*  RULE ONE SEARCHES THE GRAPH FOR CYCLES AND DELETES THE EDGE OF THE */
/* CYCLE. RULE ONE IS THEN CALLED AGAIN RECURSIVELY UNTIL THE SEARCH IS*/
/* COMPLETED WITH NO CYCLES FOUND. RULE TWO IS THEN CALLED. RULE ONE    */
/* KEEPS TRACK OF THE EDGES VISITED DURING FINAL SEARCH IN THE ARRAY    */
/* 'USED'. WHEN RULE TWO RETURNS, RULE ONE RESTORES THE FIRST EDGE OF   */
/* THE MOST RECENT CYCLE AND DELETES THE NEXT EDGE AND CALLS RULE ONE   */
/* UNTIL ALL EDGES OF THE CYCLE HAVE BEEN DELETED. THE PROCEDURE RE-    */
/* TURNS TO THE PREVIOUS CYCLE UNTIL ALL CYCLES HAVE BEEN PROCESSED.    */
/*************************************************************************/

RULE_ONE: PROC RECURSIVE;
   DCL (VERTX(M+1),N1) FIXED BINARY;
   DCL (USED(M),ANCESTOR(M),CYCLE_FLAG) BIT(1);
     VERTX=0;
     USED='0'B;
     ANCESTOR='0'B;
     CYCLE_FLAG='0'B;
     UNA  =BIT0;
     USED =BIT1;
     VERTX(1)=START;
     N1 = 1;
```

```
4      CALL SEARCH(START);
4      IF (CYCLE_FLAG=BIT0) THEN CALL RULE_TWO;
4      RETURN;
4
4   SEARCH: PROC(START) RECURSIVE;
5      DCL (START,L,L1,N2) FIXED BINARY;
5      N1=N1 + 1;
5      IF (START = TERMINAL) THEN RETURN;
5      IF (UNA(START)) THEN DO;
6         CYCLE_FLAG=BIT1;
6         L = 1;
6         DO WHILE (VERTX(L) ^= START);
7            L = L +1;
7         END;
6
6         DO L1 = L TO N1-2;
7            IF (WEIGHT(ABS(GRAPH(VERTX(L1),VERTX(L1+1))))=BIT0) THEN DO;
8               CALL REMOVE(VERTX(L1),VERTX(L1+1),Z,BIT1);
8               ANCESTOR = WEIGHT;
8               CALL RULE_ONE;
8               WEIGHT = ANCESTOR;
8               CALL RESTORE(VERTX(L1),VERTX(L1+1));
8            END;
7         END;
6      RETURN;
6      END;
5
5      UNA(START) = BIT1;
5
5      DO N2 = 1 TO N;
6         IF (GRAPH(START,N2) > 0) THEN DO;
7         VERTX(N1)=N2;
7         USED(GRAPH(START,N2)) = BIT0;
7         CALL SEARCH(N2);
7         IF (CYCLE_FLAG) THEN RETURN;
7         UNA(N2)=BIT0;
7         VERTX(N1)=0;
7         N1 =N1 - 1;
7         END;
6      END;
5      N2=N;
5      RETURN;
5      END SEARCH;
4
4   /************************************************************************/
4   /*    RULE TWO DELETES ALL UNNECESSARY EDGES BY DELETING ALL EDGES WHICH*/
4   /* HAVE A BEGINNING VERTEX WITH IN-DEGREE, GRAPH(N,0), OF ZERO OR AN    */
4   /* ENDING VERTEX WITH OUT-DEGREE, GRAPH(0,N), OF ZERO. SUCH EDGES ARE   */
4   /* KNOWN AS 'HANGING EDGES'. THE RULE ALSO DELETES ALL EDGES WHICH WERE*/
4   /* NOT VISITED BY THE FINAL SEARCH OF RULE ONE. THE RULE THEN CALLS     */
4   /* P GRAPH TO SEE IF THE REMAINING GRAPH IS A P GRAPH. IF SO, THE RULE */
4   /* CALLS RULE THREE, OTHERWISE IT RETURNS.                              */
4   /************************************************************************/
4
4   RULE_TWO: PROC;
5   DCL (I,J) FIXED BINARY;
5     DO I = 1 TO N;
```

```
6        IF (I ^= START) THEN IF (GRAPH(0,I)=0) THEN IF (GRAPH(I,0))0) THEN
6          DO J = 1 TO N;
7            IF (GRAPH(I,J) ) 0) THEN CALL REMOVE(I,J,Z,BITO);
7          END;
6        IF (I ^= TERMINAL) THEN IF(GRAPH(I,0)=0) THEN IF (GRAPH(0,I))0) THEN
6          DO J = 1 TO N;
7            IF (GRAPH(J,I) ) 0) THEN CALL REMOVE(J,I,Z,BITO);
7          END;
6      END;
5
5      DO I = 1 TO N;
6        DO J =1 TO N;
7          IF (GRAPH(I,J) ) 0) THEN IF (USED(GRAPH(I,J))) THEN
7            CALL REMOVE(I,J,Z,BITO);
7        END;
6      END;
5
5      CALL P_GRAPH;
5      IF (P_FLAG) THEN CALL RULE_THREE;
5      RETURN;
5    END RULE_TWO;
4    END RULE_ONE;
3
3  | /*************************************************************************/
3  | /*   RULE THREE FINDS ALL CHILDREN OF THE GIVEN GRAPH BY DELETING ONE   */
3  | /* SEQUENCE AT A TIME AND CALLING P_GRAPH UNTIL ALL SEQUENCES HAVE BEEN*/
3  | /* DELETED. THESE CHILDREN ARE STORED IN 'CHILDREN' AND RULE FOUR IS    */
3  | /* CALLED.                                                               */
3  | /*************************************************************************/
3
3    RULE_THREE: PROC;
4      DCL (CHILDREN(M),TEMP,I,J,K) FIXED BINARY;
4      DCL ANCESTOR(M) BIT(1);
4
4      CHILDREN=0;
4      DO I = 1 TO N;
5        DO J = 1 TO N;
6          TEMP=GRAPH(I,J);
6           IF (TEMP ) 0) THEN IF (WEIGHT(TEMP)=BITO)
6             THEN IF (CHILDREN(TEMP)=0) THEN DO;
7             CALL REMOVE(I,J,Z,BITO);
7             CALL P_GRAPH;
7             IF (P_FLAG) THEN DO;
8               CHILDREN(TEMP)=I;
8               K=0;
8               DO WHILE(D_SEQ(D_NUM-K,1) ^= I);
9                 CHILDREN(-GRAPH(D_SEQ(D_NUM-K,1),D_SEQ(D_NUM-K,2)))=-1;
9                 K=K+1;
9               END;
8             END;
7             CALL RESTORE(I,J);
7           END;
6         END;
5       END;
4
4       CALL RELIABILITY;
4       ANCESTOR=WEIGHT;
```

```
4      CALL RULE_FOUR(CHILDREN);
4      WEIGHT=ANCESTOR;
4      RETURN;
4   END RULE_THREE;
3
3   /*********************************************************************/
3   /*    RULE FOUR DELETES THE LOWEST NUMBERED CHILD FROM CHILDREN, DEL-  */
3   /* ETES THAT CHILD FROM 'CHILDREN' AND RECALLS ITSELF AGAIN. THE RULE  */
3   /* CONTINUES UNTIL A NON-P_GRAPH IS ENCOUNTERED AT WHICH TIME IT RE-   */
3   /* TURNS. IT RESTORES THE FIRST CHILD AND DELETES THE SECOND CHILD THEN*/
3   /* CALLS ITSELF AGAIN. IT CONTINUES UNTIL 'CHILDREN' IS EMPTY.        */
3   /*********************************************************************/
3
3   RULE_FOUR: PROC(FATHERS) RECURSIVE;
4     DCL (I,J,K,L,ELDER_BRO,CHILDREN(M)) FIXED BINARY;
4     DCL H FIXED BINARY;
4     DCL ANCESTOR(M) BIT(1);
4     DCL FATHERS(*) FIXED BINARY;
4
4     I=1;
4     J=1;
4     K=1;
4     L=1;
4     ELDER_BRO=1;
4     CHILDREN=1;
4
4     DO K =1 TO M;
5       CHILDREN(K)= FATHERS(K);
5     END;
4
4     L=D_NUM + 1;
4     DO WHILE(I<M);
5       DO WHILE(CHILDREN(I) <=0);
6         I=I + 1;
6         IF (I > M) THEN RETURN;
6       END;
5       K = 1;
5       DO WHILE(ABS(GRAPH(CHILDREN(I),K)) ^= I);
6         K = K + 1;
6       END;
5       ELDER_BRO=CHILDREN(I);
5       J=D_NUM + 1;
5       CALL REMOVE(ELDER_BRO,K,Z,BIT1);
5       DO H =J TO D_NUM;
6         CHILDREN(-GRAPH(D_SEQ(H,1),D_SEQ(H,2)))=0;
6       END;
5
5       CALL P_GRAPH;
5
5       IF (P_FLAG) THEN DO;
6         CALL RELIABILITY;
6         ANCESTOR=WEIGHT;
6         CALL RULE_FOUR(CHILDREN);
6         WEIGHT=ANCESTOR;
6       END;
5       ELSE DO;
6         L=0;
```

```
         DO WHILE(GRAPH(D_SEQ(D_NUM-L,1),D_SEQ(D_NUM-L,2)) ^=
                  GRAPH(ELDER_BRO,K));
           WEIGHT(-GRAPH(D_SEQ(D_NUM,1),D_SEQ(D_NUM,2)))=BIT0;
           L=L + 1;
         END;
         WEIGHT(-GRAPH(ELDER_BRO,K))=BIT0;
      END;
      CALL RESTORE(ELDER_BRO,K);
    END;
    RETURN;
  END RULE_FOUR;

/*************************************************************************/
/*    REMOVE IS THE SUBROUTINE WHICH REMOVES A REQUESTED EDGE AND ALL   */
/* EDGES WHICH MAY BE MEMBERS OF A SEQUENCE WHICH INCLUDES THE REQUEST-*/
/* ED EDGE. THE FOUR PARAMETERS PASSED TO THE SUBROUTINE ARE THE TWO    */
/* VERTICES OF THE REQUESTED EDGE, THE DIRECTION, AND THE WEIGHT_FLAG.  */
/* THE DIRECTION IS USED SINCE THE PROCEDURE IS RECURSIVE. WHEN THE     */
/* PROCEDURE IS INITIALLY CALLED, THE DIRECTION IS ZERO. THE PROCEDURE  */
/* WILL THEN CHECK TO SEE IF THE BACKWARD DIRECTION HAS AN EDGE WHICH   */
/* WILL BE A PART OF THE SEQUENCE. IF AN EDGE EXISTS IN THE BACKWARD    */
/* DIRECTION, THE PROCEDURE CALLS ITSELF WITH A DIRECTION OF -1. THE    */
/* PROCEDURE NEXT CHECKS IN THE FORWARD DIRECTION (1). THE PROCEDURE    */
/* THEN REMOVES THE REQUESTED EDGE. THE WEIGHT_FLAG IS USED TO DETER-   */
/* MINE WHETHER OR NOT TO INCLUDE THE REMOVED EDGE IN THE WEIGHT. A     */
/* VALUE OF 1 WILL WEIGHT THE EDGE.                                     */
/*************************************************************************/

REMOVE: PROC(OUT,IN,DIR,WEIGHT_FLAG) RECURSIVE;
  DCL (OUT,IN,DIR,I,J) FIXED BINARY;
  DCL WEIGHT_FLAG BIT(*);

  IF (S_FLAG=1!BOTH_OUT=1) THEN DO;
    IF (T_FLAG=1) THEN PUT SKIP EDIT('REMOVE   ') (A);
    IF (T_FLAG=1) THEN PUT EDIT (GRAPH(OUT,IN))(F(3));
  END;

  IF (H_FLAG=1!BOTH_OUT=1) THEN DO;
    IF (T_FLAG=1) THEN PUT FILE(OUTFILE) SKIP EDIT('REMOVE   ')(A);
    IF (T_FLAG=1) THEN PUT FILE(OUTFILE) EDIT(GRAPH(OUT,IN))(F(3));
  END;

    IF (GRAPH(OUT,IN) <= 0) THEN RETURN;
    J= GRAPH(OUT,IN);
    IF (DIR = 0) THEN IF (WEIGHT(J)=BIT0) THEN DO;
      D_NUM=D_NUM + 1;
      D_SEQ(D_NUM,1)= OUT;
      D_SEQ(D_NUM,2)= IN;
      IF (WEIGHT_FLAG) THEN WEIGHT(J)=BIT1;
      GRAPH(OUT,IN)=-J;
      GRAPH(OUT,0) =GRAPH(OUT,0) - 1;
      GRAPH(0,IN)  =GRAPH(0,IN) - 1;
    END;

    IF (DIR < 1) THEN IF (GRAPH(OUT,0)=0) THEN IF(GRAPH(0,OUT)=1) THEN
      DO I = 1 TO N;
        IF (GRAPH(I,OUT) > 0) THEN IF(WEIGHT(GRAPH(I,OUT))=BIT0) THEN DO;
```

```
6              GRAPH(I,OUT)=-GRAPH(I,OUT);
6              D_NUM=D_NUM + 1;
6              D_SEQ(D_NUM,1)=I;
6              D_SEQ(D_NUM,2)=OUT;
6              IF (WEIGHT_FLAG) THEN WEIGHT(-GRAPH(I,OUT))=BIT1;
6              GRAPH(0,OUT)=0;
6              GRAPH(I,0)=GRAPH(I,0) - 1;
6              CALL REMOVE(I,OUT,MO,WEIGHT_FLAG);
6              I = N + 1;
6            END;
5          END;
4
4    IF (DIR > -1) THEN IF (GRAPH(0,IN)=0) THEN IF (GRAPH(IN,0)=1) THEN
4       DO I = 1 TO N;
5          IF (GRAPH(IN,I) > 0)) THEN IF (WEIGHT(GRAPH(IN,I))=BIT0) THEN DO;
6            GRAPH(IN,I)=-GRAPH(IN,I);
6            D_NUM=D_NUM + 1;
6            D_SEQ(D_NUM,1)=IN;
6            D_SEQ(D_NUM,2)=I;
6            IF (WEIGHT_FLAG) THEN WEIGHT(-GRAPH(IN,I))=BIT1;
6            GRAPH(IN,0)=0;
6            GRAPH(0,I) =GRAPH(0,I) -1;
6            CALL REMOVE(IN,I,0,WEIGHT_FLAG);
6            I=N + 1;
6          END;
5       END;
4    RETURN;
4    END REMOVE;
3
3  /*************************************************************************/
3  /*    RESTORE IS A SUBROUTINE WHICH USES THE ARRAYS D_SEQ AND GRAPH TO */
3  /* RESTORE THE REQUESTED EDGES TO THE GRAPH.  THE INFORMATION TO RESTORE*/
3  /* IS THE FROM AND TO VERTICES OF THE EDGE REQUESTED. RESTORE WILL THEN*/
3  /* RESTORE THE REQUESTED EDGE AND ALL EDGES REMOVE SINCE THE DELETION  */
3  /* OF THE REQUESTED EDGE. IN THIS WAY, RESTORE WILL NOT ONLY RESTORE   */
3  /* A SINGLE EDGE BUT SEQUENCES WHICH MAY HAVE BEEN REMOVED INCLUDING   */
3  /* THAT EDGE.                                                          */
3  /*************************************************************************/
3
3  RESTORE: PROC(OUT,IN) RECURSIVE;
4    DCL (OUT,IN) FIXED;
4
4    IF (S_FLAG=1|BOTH_OUT=1) THEN DO;
5       IF (T_FLAG=1) THEN PUT SKIP EDIT ('RESTORE   ')(A);
5       IF (T_FLAG=1) THEN PUT EDIT(GRAPH(D_SEQ(D_NUM,1),D_SEQ(D_NUM,2)))(F(3));
5    END;
4
4    IF (H_FLAG=1|BOTH_OUT=1) THEN DO;
5       IF (T_FLAG=1) THEN PUT FILE(OUTFILE) SKIP EDIT('RESTORE   ')(A);
5       IF (T_FLAG=1) THEN PUT FILE(OUTFILE) EDIT(GRAPH(D_SEQ(D_NUM,1),
5                                          D_SEQ(D_NUM,2)))(F(3));
5    END;
4
4       GRAPH(D_SEQ(D_NUM,1),D_SEQ(D_NUM,2))=
4                                   -GRAPH(D_SEQ(D_NUM,1),D_SEQ(D_NUM,2));
4       GRAPH(D_SEQ(D_NUM,1),0)=GRAPH(D_SEQ(D_NUM,1),0) + 1;
4       GRAPH(0,D_SEQ(D_NUM,2))=GRAPH(0,D_SEQ(D_NUM,2)) + 1;
```

```
4       D_NUM=D_NUM - 1;
4       IF (GRAPH(D_SEQ(D_NUM + 1,1),D_SEQ(D_NUM+1,2)) ^= GRAPH(OUT,IN))
4         THEN CALL RESTORE(OUT,IN);
4       RETURN;
4       END RESTORE;
3
3   /*************************************************************************/
3   /*     P_GRAPH CHECKS TO SEE IF THE CURRENT GRAPH IS A P_GRAPH. IF THE   */
3   /* GRAPH IS NON-CYCLIC, THEN THE GRAPH WILL BE P_ACYCLIC. THE PROCEDURE*/
3   /* CHECKS BY SEEING IF ALL INTERNAL VERTICES HAVE IN DEGREE AND OUT      */
3   /* DEGREE GREATER THAN ZERO. SINCE THE IN DEGREES ARE LISTED IN THE      */
3   /* ZERO ROW, AND THE OUT DEGREES ARE LISTED IN THE ZERO COLUMN. IT IS    */
3   /* ONLY NECCESSARY TO CHECK TO SEE IF ALL VERTICES EXCLUDING THE START   */
3   /* AND THE TERMINAL, HAVE NON-ZERO ENTRIES. IF A NODE HAS BEEN DELETED,*/
3   /* IT WILL HAVE BOTH IN DEGREE AND OUT DEGREE OF ZERO. A P_GRAPH GIVES */
3   /* P_FLAG =1, ELSE P_FLAG=0.                                             */
3   /*************************************************************************/
3
3   P_GRAPH:PROC;
4   DCL I FIXED BINARY;
4     P_FLAG=BIT1;
4     IF (GRAPH(START,0) =0) THEN P_FLAG=BIT0;
4     IF (GRAPH(0,TERMINAL)=0) THEN P_FLAG=BIT0;
4     I=0;
4
4     DO WHILE(P_FLAG & I < N);
5       I =I + 1;
5       IF (I ^= START) THEN IF (I ^= TERMINAL)
5        THEN IF (GRAPH(0,I)=0 | GRAPH(I,0)=0)
5          THEN IF (GRAPH(0,I) ^= GRAPH(I,0)) THEN P_FLAG=BIT0;
5     END;
4
4     IF (S_FLAG=1|BOTH_OUT=1) THEN DO;
5       IF (T_FLAG=1) THEN DO;
6         PUT SKIP EDIT('P_FLAG STATUS IS ',P_FLAG)(X(10),A,A);
6       END;
5     END;
4
4     IF (H_FLAG=1|BOTH_OUT=1) THEN DO;
5       IF (T_FLAG=1) THEN DO;
6         PUT FILE(OUTFILE) SKIP EDIT('P_FLAG STATUS IS  ',P_FLAG)(X(10),A,A);
6       END;
5     END;
4
4      RETURN;
4     END P_GRAPH;
3
3   /*************************************************************************/
3   /*     RELIABILITY TAKES THE ORIGINAL GRAPH RELIABILITY AND DIVIDES BY   */
3   /* THE RELIABILITIES OF THE EDGES WHICH HAVE BEEN REMOVED. THESE EDGES */
3   /* ARE IN THE ARRAY 'D_SEQ' AND THE NUMBER OF EDGES WHICH HAVE BEEN      */
3   /* REMOVED IS 'D_NUM'. D_NUM IS A POINTER WHICH GIVES THE CURRENT POSI-*/
3   /* TION IN 'D_SEQ'.                                                      */
3   /*************************************************************************/
3
3   RELIABILITY: PROC;
4     DCL (EDGES(M),I,J,M_NUM,N_NUM) FIXED BINARY;
```

```
DCL SUBGRAPH_REL FLOAT;

EDGES=0; I=0; J=0; M_NUM=0; N_NUM=0;
SUBGRAPH_REL=RELIA;

DO I =1 TO D_NUM;
  J= -GRAPH(D_SEQ(I,1),D_SEQ(I,2));
  SUBGRAPH_REL=SUBGRAPH_REL/EDGE_REL(-GRAPH(D_SEQ(I,1),D_SEQ(I,2)));
  EDGES(J)=1;
END;

DO I =1 TO N;
  IF (GRAPH(0,I) > 0) THEN DO;
    N_NUM=N_NUM + 1;
    M_NUM=M_NUM + GRAPH(0,I);
  END;
END;

SIGN=(-1)**(M_NUM-N_NUM);
SUBGRAPH_REL= SUBGRAPH_REL * SIGN;

IF (I_FLAG=1) THEN DO I = 1 TO M;
  IF (EDGES(I) =0) THEN
    IMPORTANCE(I)=IMPORTANCE(I) +SUBGRAPH_REL/EDGE_REL(I);
END;

GRAPH_REL=GRAPH_REL + SUBGRAPH_REL;
IF (S_FLAG=1!BOTH_OUT=1) THEN DO;
  IF (L_FLAG=1) THEN DO;
    PUT SKIP(2);
    PUT EDIT(SIGN,'')(F(8),COL(20),A);
    DO I =1 TO M;
      IF (EDGES(I) =0) THEN PUT EDIT(I)(F(3));
    END;

    PUT EDIT (SUBGRAPH_REL)(COL(60),F(8,4));

    LINE=LINE + 1;
    IF LINE > 8 THEN CALL ADVANCE2;
    IF LINE > 8 THEN CALL HEADER1;
  END;
END;

IF (H_FLAG=1!BOTH_OUT=1) THEN DO;
  IF (L_FLAG=1) THEN DO;
    PUT FILE(OUTFILE) SKIP(3);
    PUT FILE(OUTFILE) EDIT(SIGN,'')(F(8),COL(20),A);
    DO I =1 TO M;
      IF (EDGES(I) =0) THEN PUT FILE(OUTFILE) EDIT(I)(F(3));
    END;

    PUT FILE(OUTFILE) EDIT(SUBGRAPH_REL)(COL(60),F(8,4));

    LLINE=LLINE + 1;
    IF LLINE > 16 THEN CALL HEADER2;
  END;
 END;
```

```
RETURN;
END RELIABILITY;

END B;

ADVANCE1: PROC;
   DCL ANY CHAR(80) VAR;
   PUT SKIP(3) EDIT('PRESS RETURN TO CONTINUE    ')(X(10),A);
   GET LIST(ANY);
   RETURN;
   END ADVANCE1;

HEADER: PROC;
 PUT SKIP(2) EDIT('EDGE NO.','FROM ','TO ','DIRECTION','RELIABILITY')
                    (X(5),A,X(5),A,X(5),A,X(5),A,X(5),A);
 RETURN;
 END HEADER;

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

DATA_IN: PROC;
   DCL(I,J,K,K1,K2,L1) FIXED BINARY;
   DCL COMD CHAR(1);

ALL_EDGE: K=0;
   K1=0;
   PUT PAGE;
   PUT SKIP(3) EDIT('YOU MUST REMEMBER THE NUMERIC ORDER OF THE EDGE')(X(17),A);
   PUT SKIP EDIT('SO CORRECTIONS MAY BE MADE SELECTIVELY')(X(17),A);
   PUT SKIP(3);

   DO I= 1 TO M;
      K =K + 1;
      K1=K1 + 1;
      PUT SKIP;

      PUT SKIP EDIT(I,'.','EDGE FROM      = ')(F(3),A,A);
      GET LIST(DATA(K,1));

      PUT SKIP EDIT('TO              = ')(X(4),A);
      GET LIST(DATA(K,2));

      PUT SKIP EDIT('STATUS(0 OR 1)= ')(X(4),A);
      GET LIST(DATA(K,3));

      PUT SKIP EDIT('RELIABILITY IS= ')(X(4),A);
      GET LIST(REL(K));

 END;

CHANGE_INPUT3: PUT PAGE;
   PUT SKIP(3) EDIT('REQUEST TO CHANGE EDGE DATA')(X(21),A);
   PUT SKIP(3) EDIT('1. SELECTIVE CHANGE REQUESTED')(X(16),A);
   PUT SKIP EDIT    ('2. REENTER ALL')(X(16),A);
   PUT SKIP EDIT    ('3. NO CHANGES REQUIRED, ADVANCE TO NEXT STEP')
                    (X(16),A);
```

```
3      PUT SKIP(3) EDIT('ENTER 1,2 OR 3   ')(X(10),A);
3      GET LIST (COMD);
3
3      IF COMD='1' THEN GO TO SELECT_E;
3      IF COMD='2' THEN GO TO ALL_EDGE;
3      IF COMD='3' THEN RETURN;
3
3   ERROR6: PUT SKIP(3) EDIT('***OUT OF RANGE, REENTER***')(X(15),A);
3      CALL ADVANCE2;
3      GO TO CHANGE_INPUT3;
3
3   SELECT_E:PUT PAGE;
3      PUT SKIP(3) EDIT('WHICH EDGE INFORMATION IS TO BE ALTERED?')(X(20),A);
3      PUT SKIP    EDIT('ENTER THE NUMERIC ORDER OF THE EDGE')(X(22),A);
3      PUT SKIP(3) EDIT('ENTER ONE NUMBER NOW    ')(X(15),A);
3      GET LIST(L1);
3
3      IF (L1 < 1) | (L1 > K1) THEN GO TO ERROR6;
3      I=L1;
3      K=L1;
3
3   ENTER_E: PUT PAGE;
3      PUT SKIP EDIT(I,'.','EDGE FROM      = ',DATA(K,1))(F(3),A,A,F(3));
3      PUT SKIP EDIT('TO                = ',DATA(K,2))(X(4),A,F(3));
3      PUT SKIP EDIT('STATUS(0 OR 1)= ',DATA(K,3))(X(4),A,F(3));
3      PUT SKIP EDIT('RELIABILITY IS= ',REL(K))(X(4),A,F(5,3));
3
3   CHANGE_EDGE: PUT SKIP(2) EDIT ('YOU NEED TO CHANGE:')(X(30),A);
3      PUT SKIP EDIT    ('1. EDGE FROM')(X(22),A);
3      PUT SKIP EDIT    ('2. TO')       (X(22),A);
3      PUT SKIP EDIT    ('3. STATUS')    (X(22),A);
3      PUT SKIP EDIT    ('4. RELIABILITY')(X(22),A);
3      PUT SKIP EDIT    ('5. ALL OF ABOVE')(X(22),A);
3      PUT SKIP(3) EDIT ('ENTER 1,2,3,4 OR 5     ')(X(18),A);
3      GET LIST(COMD);
3
3
3      IF COMD='1' THEN GO TO EDGE_FROM;
3      IF COMD='2' THEN GO TO TO_EDGE;
3      IF COMD='3' THEN GO TO STATUS_E;
3      IF COMD='4' THEN GO TO RELIA_E;
3      IF COMD='5' THEN GO TO EDGE_FROM;
3
3   ERROR7: PUT SKIP(3) EDIT('***OUT OF RANGE,REENTER***')(X(15),A);
3      CALL ADVANCE2;
3      GO TO CHANGE_EDGE;
3
3   EDGE_FROM:PUT SKIP(2);
3      PUT SKIP EDIT(I,'.','EDGE FROM      = ')(F(3),A,A);
3      GET LIST(DATA(K,1));
3      IF COMD='5' THEN GO TO TO_EDGE;
3      GO TO CHECK1;
3
3   TO_EDGE:PUT SKIP(2);
3      PUT SKIP EDIT('TO               = ')(X(4),A);
3      GET LIST(DATA(K,2));
3      IF COMD='5' THEN GO TO STATUS_E;
```

```
  GO TO CHECK1;

STATUS_E: PUT SKIP(2);
  PUT SKIP EDIT('STATUS(0 OR 1)= ')(X(4),A);
  GET LIST(DATA(K,3));
  IF COMD='5' THEN GO TO RELIA_E;
  GO TO CHECK1;

RELIA_E:PUT SKIP(2);
  PUT SKIP EDIT('RELIABILITY IS= ')(X(4),A);
  GET LIST(REL(K));

CHECK1: PUT SKIP(3) EDIT('MORE CHANGES? ENTER Y/N  ')(X(20),A);
  GET LIST(COMD);
  IF COMD='Y' THEN GO TO SELECT_E;
  IF COMD='N' THEN GO TO CHANGE_INPUT3;

ERROR5:PUT SKIP(3) EDIT('***INCORRECT INPUT,REENTER***')(X(15),A);
  CALL ADVANCE2;
  GO TO CHECK1;

ERROR6:PUT SKIP(3) EDIT('***OUT OF RANGE, REENTER***')(X(15),A);
  CALL ADVANCE2;
  GO TO SELECT_E;

/*                                                                  */




RETURN;
END DATA_IN;

HEADER1: PROC;
  PUT SKIP EDIT('******LINKSET******')(X(25),A);
  PUT SKIP EDIT('SIGN','SUBGRAPH','RELIABILITY')(X(6),A,X(18),A,X(20),A);
  PUT SKIP(2);
  LINE=0;
  RETURN;
  END HEADER1;

HEADER2: PROC;
  PUT FILE(OUTFILE) SKIP EDIT('******LINKSET******')(X(25),A);
  PUT FILE(OUTFILE) SKIP EDIT('SIGN','SUBGRAPH','RELIABILITY')
                             (X(6),A,X(18),A,X(20),A);
  PUT FILE(OUTFILE) SKIP;
  LLINE=0;
  RETURN;
  END HEADER2;

ADVANCE2: PROC;
  DCL ANY CHAR(80) VAR;
  PUT SKIP(3) EDIT ('PRESS RETURN TO CONTINUE  ')(X(10),A);
  GET LIST(ANY);
  RETURN;
  END ADVANCE2;
```

```
 1139     2 END A;
 1140     1 END INSTRAP;
```

COMMAND LINE
------- ----

PLI INSTRAP/LIS

APPENDIX D

Same example as in Appendix B is used here. The system graph is shown again in the Exhibit D-1. There are 6 vertices and 12 edges in the system graph. The start vertex is 1 and the terminal vertex is 6. Hence, N=6, M=12, START=1, and TERMINAL=6. Since the system graph is directed, the desired status is equal to 1 for every edge. The reliability of each edge is assumed to be .90.

Instead of showing the interactive data entry mode, the following section illustrates how one can load the existing VAX file with proper data format to the INSTRAP program. Regardless of the data entry mode, the result is still the same.

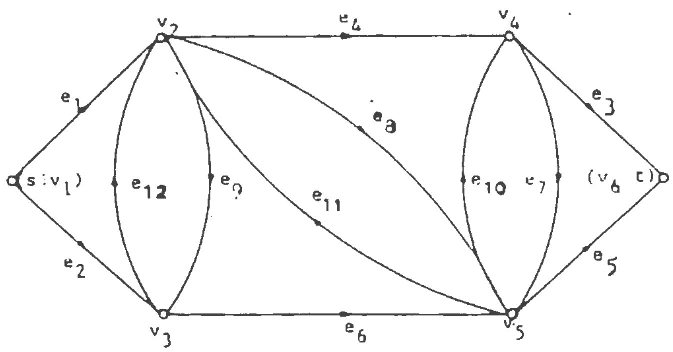

Exhibit D-1 The System Reliability Graph

```
OKLAHOMA STATE UNIVERSITY COMPUTER CENTER ASYNCHRONOUS COMMUNICATION NETWC
02.059
ENTER SYSTEM NAME IN CAPITAL LETTERS (IBM OR VAX)
VAX
COM

Username: U5162AA
Password:

               !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
               !!!          OKLAHOMA STATE UNIVERSITY       !!!
               !!!                   VMS 4.1                !!!
               !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

NOTE: one of the disk drives is having hardware problems.  This is disk
      dua2,  none of the normal user accounts are on it so users should
      not be affected.
    Last interactive login on Thursday, 14-NOV-1985 21:51
%DCL-W-UNDFIL, file has not been opened by DCL - check logical name

 DISK QUOTA INFORMATION:
  User [U5162AA] has 931 blocks used, 1069 available,
  of 2000 authorized and permitted overdraft of 1000 blocks on DUAC

$ RUN INSTRAP

                         INTERACTIVE STRAP PROGRAM
                         *************************

                            ***MAIN MENU***

                1. TO ENTER THE DATA INTERACTIVELY
                2. TO ENTER THE DATA THROUGH FILE
                3. EXIT

      ENTER 1,2 OR 3    2
```

```
YOUR DATA FILE SHOULD BE ALREADY CREATED
IT MUST BE NAMED INFSTRAP.DAT
THE DATA FILE SHOULD BE IN THE FOLLOWING FORMAT

N,M,S,T,
B,E,D,R,
    .
    .
B,E,D,R

NOTE:USE COMMA TO SEPARATE THE VALUES, AND BETWEEN LINES

IF NOT, TYPE IN E TO EXIT TO MAIN MENU
IF YES, TYPE IN C TO CONTINUE  C
```

```
                OUTPUT SELECTIONS

  1. ECHO PRINT,LINKSET, AND RELIABILITY
  2. OPTION 1. AND IMPORTANCE CALCULATION
  3. TRACE ROUTINE
  4. ALL OF THE ABOVE

ENTER 1,2,3, OR 4   2
```

OUTPUT MODE

1. SCREEN OUTPUT REQUESTED
2. HARD COPY REQUESTED
3. ALL OF THE ABOVE

ENTER 1,2, OR 3   1

ECHO PRINT OF INPUT DATA

NO. OF VERTICES = 6

NO. OF EDGES = 12

START VERTEX = 1

TERMINAL VERTEX = 6

PRESS RETURN TO CONTINUE

EDGE NO.	FROM	TO	DIRECTION	RELIABILITY
1.	1	2	1	0.900
2.	1	3	1	0.900
3.	4	6	1	0.900
4.	2	4	1	0.900
5.	5	6	1	0.900
6.	3	5	1	0.900
7.	4	5	1	0.900
8.	2	5	1	0.900
9.	2	3	1	0.900
10.	5	4	1	0.900
11.	5	2	1	0.900
12.	3	2	1	0.900

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
1	2 5 6	0.7290
-1	2 3 5 6 10	-0.5905
1	2 3 6 10	0.6561
1	1 2 5 6 8 12	0.5314
-1	2 5 6 8 12	-0.5905
1	2 5 8 12	0.6561
-1	1 2 5 8 12	-0.5905
1	1 5 8	0.7290
-1	1 2 5 6 8	-0.5905

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
-1	1 2 3 5 6 8 10 12	-0.4305
1	2 3 5 6 8 10 12	0.4783
-1	2 3 6 8 10 12	-0.5314
1	2 3 8 10 12	0.5905
-1	2 3 5 8 10 12	-0.5314
1	1 2 3 6 8 10 12	0.4783
-1	1 2 3 8 10 12	-0.5314
1	1 3 8 10	0.6561
-1	1 2 3 6 8 10	-0.5314

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
1	1 2 3 5 8 10 12	0.4783
-1	1 3 5 8 10	-0.5905
1	1 2 3 5 6 8 10	0.4783

1	1 2 3 4 5 6 10 11 12	0.3874
-1	2 3 4 5 6 10 11 12	-0.4305
1	2 3 4 6 10 11 12	0.4783
-1	2 3 4 6 11 12	-0.5314
1	2 3 4 12	0.6561
1	2 3 4 6 11	0.5905

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
-1	2 3 4 6 10 12	-0.5314
-1	2 3 4 6 10 11	-0.5314
1	2 3 4 5 6 11 12	0.4783
-1	2 3 4 5 6 12	-0.5314
-1	2 3 4 5 6 11	-0.5314
1	2 3 4 5 6 10 12	0.4783
1	2 3 4 5 6 10 11	0.4783
-1	1 2 3 4 6 10 11 12	-0.4305
1	1 2 3 4 6 11 12	0.4783

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
-1	1 2 3 4 12	-0.5905
1	1 3 4	0.7290
-1	1 2 3 4 6 11	-0.5314
1	1 2 3 4 6 10 12	0.4783
-1	1 2 3 4 6 10	-0.5314
1	1 2 3 4 6 10 11	0.4783
-1	1 2 3 4 5 6 11 12	-0.4305
1	1 2 3 4 5 6 12	0.4783
-1	1 2 3 4 5 6	-0.5314

PRESS RETURN TO CONTINUE

******LINKSET******

1	1 2 3 4 5 6 11	0.4783
-1	1 2 3 4 5 6 10 12	-0.4305
1	1 2 3 4 5 6 10	0.4783
-1	1 2 3 4 5 6 10 11	-0.4305
1	1 2 3 4 5 6 8 10 12	0.3874
-1	2 3 4 5 6 8 10 12	-0.4305
1	2 3 4 6 8 10 12	0.4783
-1	2 3 4 8 10 12	-0.5314
1	2 3 4 5 8 10 12	0.4783

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
-1	2 3 4 5 8 12	-0.5314
1	2 3 4 5 6 8 12	0.4783
-1	1 2 3 4 6 8 10 12	-0.4305
1	1 2 3 4 8 10 12	0.4783
-1	1 3 4 8 10	-0.5905
1	1 2 3 4 6 8 10	0.4783
-1	1 2 3 4 5 8 10 12	-0.4305
1	1 2 3 4 5 8 12	0.4783
-1	1 3 4 5 8	-0.5905

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
1	1 3 4 5 8 10	0.5314
-1	1 2 3 4 5 6 8 12	-0.4305
1	1 2 3 4 5 6 8	0.4783
-1	1 2 3 4 5 6 8 10	-0.4305
1	1 2 3 4 5 6 7 8 12	0.3874
-1	2 3 4 5 6 7 8 12	-0.4305

SIGN	SUBGRAPH	RELIABILITY
-1	2 4 5 7 8 12	-0.5314
1	2 4 5 7 12	0.5905

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
-1	2 4 5 6 7 12	-0.5314
1	2 3 4 5 7 8 12	0.4783
-1	2 3 4 5 7 12	-0.5314
1	2 3 4 5 6 7 12	0.4783
-1	1 2 4 5 6 7 8 12	-0.4305
1	1 2 4 5 7 8 12	0.4783
-1	1 2 4 5 7 12	-0.5314
1	1 4 5 7	0.6561
-1	1 4 5 7 8	-0.5905

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
1	1 2 4 5 6 7 12	0.4783
-1	1 2 4 5 6 7	-0.5314
1	1 2 4 5 6 7 8	0.4783
-1	1 2 3 4 5 7 8 12	-0.4305
1	1 2 3 4 5 7 12	0.4783
-1	1 3 4 5 7	-0.5905
1	1 3 4 5 7 8	0.5314
-1	1 2 3 4 5 6 7 12	-0.4305
1	1 2 3 4 5 6 7	0.4783

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
-1	1 2 3 4 5 6 7 8	-0.4305
1	1 2 3 4 5 6 7 8 9	0.3874

-1	1 3 4 5 6 7 8 9	-0.4305
1	1 4 5 6 7 8 9	0.4783
-1	1 5 6 8 9	-0.5905
1	1 5 6 9	0.6561
-1	1 4 5 6 7 9	-0.5314
1	1 3 4 5 6 8 9	0.4783
-1	1 3 4 5 6 9	-0.5314

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
1	1 3 4 5 6 7 9	0.4783
-1	1 2 4 5 6 7 8 9	-0.4305
1	1 2 5 6 8 9	0.5314
-1	1 2 5 6 9	-0.5905
1	1 2 4 5 6 7 9	0.4783
-1	1 2 3 4 5 6 8 9	-0.4305
1	1 2 3 4 5 6 9	0.4783
-1	1 2 3 4 5 6 7 9	-0.4305
1	1 2 3 4 5 6 8 9 10	0.3874

PRESS RETURN TO CONTINUE

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
-1	1 3 4 5 6 8 9 10	-0.4305
1	1 3 5 6 8 9 10	0.4783
-1	1 3 6 8 9 10	-0.5314
1	1 3 6 9 10	0.5905
-1	1 3 5 6 9 10	-0.5314
1	1 3 4 6 8 9 10	0.4783
-1	1 3 4 6 9 10	-0.5314
1	1 3 4 5 6 9 10	0.4783
-1	1 2 3 5 6 8 9 10	-0.4305

******LINKSET******

SIGN	SUBGRAPH	RELIABILITY
1	1 2 3 6 8 9 10	0.4783
-1	1 2 3 6 9 10	-0.5314
1	1 2 3 5 6 9 10	0.4783
-1	1 2 3 4 6 8 9 10	-0.4305
1	1 2 3 4 6 9 10	0.4783
-1	1 2 3 4 5 6 9 10	-0.4305
GRAPH RELIABILITY		0.977186

EDGE IMPORTANCE

```
IMPORTANCE( 1) = 0.107581
IMPORTANCE( 2) = 0.090679
IMPORTANCE( 3) = 0.090679
IMPORTANCE( 4) = 0.019218
IMPORTANCE( 5) = 0.107581
IMPORTANCE( 6) = 0.019218
IMPORTANCE( 7) = 0.000860
IMPORTANCE( 8) = 0.011279
IMPORTANCE( 9) = 0.000860
IMPORTANCE(10) = 0.008872
IMPORTANCE(11) = 0.000066
IMPORTANCE(12) = 0.008872
```

PRESS RETURN TO CONTINUE

```
*************************

         ***MAIN MENU***

    1. TO ENTER THE DATA INTERACTIVELY
    2. TO ENTER THE DATA THROUGH FILE
    3. EXIT


      ENTER 1,2 OR 3   3
$ LO
  U5162AA       logged out at 14-NOV-1985 22:34:59.15
DISC
```

APPENDIX E

TELENET ACCESS

OSU COMPUTER CENTER NETWORK MESSAGES

TELENET ACCESS

The OSU Computer Center's asynchronous network is accessible via the nation-wide Telenet public data network. Telenet, as a public data network, offers data communications service much like the phone company offers voice communications. With Telenet you do not need to make a long distance phone call to Stillwater from your remote location.

Telenet provides dial-in telephone/modem lines in many cities across the country. You call one of these numbers, connect a terminal to the phone line and indicate to Telenet the host computer system you desire to use. After the selection is entered, Telenet establishes a connection with the host system and the timesharing session proceeds as usual. The procedures below discuss this sign-on process in greater detail.

To obtain a current copy of Telenet phone numbers follow these steps: (1) Dial into Telenet (624-1112) in Stillwater and properly connect to the Telenet system; (2) after the @ prompt, enter MAIL; (3) after the username = prompt, enter PHONES; (4) after the password prompt, enter PHONES; and (5) follow the procedures Telenet gives.

To Sign-on

Dial the Telenet access number. When you hear the high-pitched tone, couple the telephone and modem in the appropriate mannner.

Enter two carriage returns.

Telenet will respond with a network herald followed by your terminal port address and prompt you to identify your terminal model. Type D1 (D1 implies terminal characteristics common to most terminals today), and hit carriage return.

Telenet will respond with an @ symbol. This is Telenet's prompt character and indicates that Telenet is waiting for you to enter a command. In response to this prompt, enter: ID ;40530/A030SU01

Telenet will prompt for the corresponding password with the PASSWORD= prompt. Enter 167136 for the password.

Telenet will attempt to connect you to the OSU Computer Center network and will tell you the status of the connection with one of these messages: CONNECTED, NOT OPERATING, BUSY or REFUSED.

If the CONNECTED message is displayed, you are connected to the OSU Computer Center network. Enter two carriage returns, and you should receive the normal messages from the network which prompt you for a system name.

SPECIFY THE DESIRED SYSTEM

In response to the ENTER SYSTEM NAME prompt, you should enter one of the valid system names that was listed in parentheses and depress the RETURN key (be sure to enter the system name in capital letters). If a port is available on the selected system, you will be connected to it and the message, COM, will be displayed at the terminal. You have 30 seconds to specify a system name, or the network will send you the "...TIMEOUT" message and disconnect you.

Current UCC system names are:

IBM - IBM system
IBM3270 - IBM system full screen
VAX - VAX system using 1200 bps communications

LOGON TO THE SELECTED SYSTEM

After receiving the COM message, depress the RETURN key once more and wait for the first message to come from the system you have selected. When this first message has been displayed, enter the required information to logon to the selected system.

SYSTEM: VAX

If you have selected the VAX system name, you will be connected to an available port on the VAX 11/780 system. The first prompt from the VAX system will be the following:

Username:

To this prompt, enter your VAX userid. Next you will be prompted for your password. If the userid and password you have entered are valid, you will be logged on to the VAX system.

LOGOFF FROM THE SELECTED SYSTEM OR APPLICATION

At the end of the session, logoff in the appropriate manner from the system you have selected. Consult the documentation for the system you are using for an exact description of the procedure. For most systems and applications the command is either LOGOUT or LOGOFF.

SYSTEM: VAX

When you logoff from the VAX system you will automatically be disconnected from the network (after logging off you will receive the message DISC).

Directly Connected Terminals

If you were using a directly connected terminal, you do not need to do anything else to disconnect from the network.

Dial-up Terminals:

If you dialed up into the network then break the phone-to-modem connection and hang up the telephone.

NETWORK MESSAGES

OSU uses a Rixon DCX network processor to manage and control its asynchronous network. The Rixon processor issues a variety of messages to indicate the success, failure or status of your connection and interaction with the network.

<u>COM</u>

Meaning: Connection made. You have now been connected to a port on the system you requested.

Action: Depress the RETURN key once and wait for the message from the selected computer system inviting you to logon.

<u>MOM</u>

Meaning: Connection attempt failed. The network was unable to satisfy your request.

Action: Wait for the next message.

<u>ERR</u>

Meaning: There was an error in system selection (e.g., a non alphanumeric character was entered as part of an alphanumeric name or an invalid number of characters was entered), or the requested port or node is closed.

Action: Try again, following correct procedure.

<u>INV</u>

Meaning: Invalid selection request (e.g., destination port set to incompatible speed); a request was made to queue the connection request to the Automatic Retry Queue when no connection attempt has been entered; or the network response on the previous attempt was ERR, INV, NP, or DER.

Action: Enter valid connection request.

<u>NP</u>

Meaning: Nonexistent port, short-form address or alphanumeric name (e.g., port not configured, or spare); no route configured for internode connections, or the number of internode links for the connection has exceeded the maximum value.

Action: Try again, using a proper port number, short-form address, or alphanumeric name or route.

<u>OCC</u>

Meaning: The port requested is occupied or all ports associated with the system name specified are occupied.

Action: Try later or request that your selection be queued until a port is available. When a port is available, you will receive the prompt for application name if you requested the IBM system or the prompt for username if you requested the VAX. To queue your selection, type the letter Q and then press the RETURN key or the SEND key as shown below:

```
ENTER SYSTEM NAME IN CAPITAL LETTERS
OCC
ENTER SYSTEM NAME IN CAPITAL LETTERS
Q
```

DER

Meaning: The port requested is closed, out of order, or not open and working.
Action: Try again. If problems persist, contact UCC.

NA

Meaning: The port originating the connection request is not permitted to access the requested port, or queuing to the Automatic Retry Queue is inhibited.
Action: Verify that you are making a valid system selection.

NC

Meaning: No free channels available on link (internode connections), or requesting, or destination port link is overloaded.
Action: Recontact network when another attempt is required or queue the connection attempt to the Automatic Retry Queue.

QUEUED

Meaning: The connection attempt has been queued to the Automatic Retry Queue.
Action: Wait for another network message (either NP, NA, COM, INV, or DER).

DATA LOST

Meaning: The two connected ports have been forcibly disconnected.
Action: Contact UCC.

...OVFL

Meaning: The network's memory is almost completely committed, and will be slowing down data traffic thru the network.
Action: This situation may occur for short intervals when the network has a sudden heavy workload, but should clear itself in due course.

DISC

Meaning: The two connected ports have been disconnected.
Action: No action needs to be taken if you requested the disconnection.

CNX FAILURE PLEASE REQ RECONNECTION

Meaning: The two connected ports were at different nodes in the network and have been disconnected due to internode link failure.
Action: Reconnect to the network and reselect the system you were using.

...TIMEOUT

Meaning: The user did not specify a system name within 30 seconds of the ENTER SYSTEM NAME prompt. The user's terminal has been disconnected from the network.
Action: Reconnect to the network and specify a system name within 30 seconds.

VITA

MAUNG M. LAY
(aka Clifford Shoung)

Candidate for the Degree of

Master of Business Administration

Report: SYSTEM RELIABILITY PROGRAMS FOR VAX COMPUTER SYSTEM: INTERACTIVE SUM OF DISJOINT PRODUCTS (INSDP) & INTERACTIVE SIMPLIFIED TOPOLOGICAL RELIABILITY ANALYSIS PROGRAM (INSTRAP)

Major Field: Business Administration

Biographical

Personal Data: Born in Rangoon, Burma, October 17, 1956, the son of Shook Khyen Shoung and Daw Nyunt.

Education: Graduated from Ruamrudee International School, Bangkok, Thailand, June 1976; received Bachelor of Science degree from Iowa State University with a major in Industrial Engineering, May, 1980; also received Master of Science degree from Oklahoma State University with a major in Industrial Engineering, December, 1982; completed requirements for the Master of Business Administration at Oklahoma State University, January, 1986.

Experience: Graduate Teaching Assistant, College of Business Administration, OSU, 1984-1985; Graduate Research Assistant, College of Business Administration, OSU, 1984-1985; Graduate Teaching Assistant, the School of Industrial Engineering, OSU, 1981-1984; Research Associate, Oklahoma Productivity Center, OSU, 1982.