

197-053

①

DOCUMENTATION OF DECISION-AIDING SOFTWARE: EVAL SYSTEM SPECIFICATION

DECISIONS AND DESIGNS INC.

Linda B. Allardyce
Dorothy M. Arney
Phillip H. Feuerwerger
Roy M. Gulick

November 1979

N00014-79-C-0069

DTIC
NOV 20 1982
H

ADVANCED DECISION TECHNOLOGY PROGRAM

CYBERNETICS TECHNOLOGY OFFICE
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY
Office of Naval Research • Engineering Psychology Programs

DISTRIBUTION STATEMENT A

Approved for public release

When published in the open literature

82 11 26 193

AD A121844

DTIC FILE COPY

DOCUMENTATION OF DECISION-AIDING SOFTWARE:

EVAL SYSTEM SPECIFICATION

by

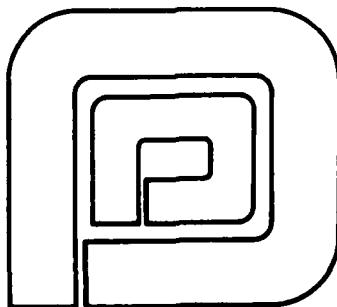
Linda B. Allardyce, Dorothy M. Amey, Phillip H. Feuerwerger, and Roy M. Gulick

Sponsored by

Defense Advanced Research Projects Agency
ARPA Order 3469

November 1979

DTIC
ELECTE
NOV 29 1982
H



DECISIONS and DESIGNS, INC.

Suite 600, 8400 Westpark Drive
P.O. Box 907
McLean, Virginia 22101
(703) 821-2828

DISTRIBUTION STATEMENT A

Approved for public release;
Distribution Unlimited

CONTENTS

	<u>Page</u>
FIGURES	iii
1.0 INTRODUCTION	1
1.1 Purpose of the System Specification	1
1.2 References	1
1.3 Terms	2
1.3.1 EVAL	2
1.3.2 HIPO	2
2.0 DESIGN DETAILS	3
2.1 Background	3
2.2 General Operating Procedures	3
2.3 System Logical Flow	3
2.4 HIPO Documentation	6



Accession For	
DTIC GRAI	☒
DTIC TAB	☐
Unannounced	☐
Justification	<i>Per File</i>
<i>on file</i>	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
<i>A</i>	

FIGURES

<u>Figure</u>		<u>Page</u>
2-1	LEGEND OF HIPO SYMBOLS	5
2-2	EVAL SYSTEM OVERVIEW	7
2-3	STRUCTURE VISUAL TABLE OF CONTENTS	8
2-4	RUN VISUAL TABLE OF CONTENTS	9

EVAL SYSTEM SPECIFICATION

1.0 INTRODUCTION

1.1 Purpose of the System Specification

The EVAL System Specification is a technical document written for software development personnel. Together with the EVAL Functional Description, it guides the software development effort by identifying the functional requirements and by providing structured logic diagrams that depict the flow, control, and processing of information within the system.

The System Specification is generic and is intended to guide and facilitate the preparation of the language-specific and computer hardware-specific documentation and coding that are necessary to implement and operate EVAL at an installation. *

1.2 References

1.2.1 IBM, HIPO--A Design Aid and Documentation Technique. Technical Publication GC20-1851-0. White Plains, New York: IBM, October 1974.-

1.2.2 Allardyce, Linda B.; Amey, Dorothy M.; Feuerwerger, Phillip H.; Gulick, Roy M. Documentation of Decision-Aiding Software: EVAL Functional Description. McLean, Virginia: Decisions and Designs, Incorporated, November 1979.

1.2.3 Allardyce, Linda B.; Amey, Dorothy M.; Feuerwerger, Phillip H.; Gulick, Roy M. Documentation

of Decision-Aiding Software: EVAL Users Manual.
McLean, Virginia: Decisions and Designs, Incorporated, November 1979.

1.3 Terms

1.3.1 EVAL - EVAL is an abbreviation for evaluation, reflecting the system's major area of applicability.

1.3.2 HIPO - The specification uses the standard Hierarchy plus Input-Process-Output (HIPO) diagramming technique to depict the structural design and logical flow of the system. A legend explaining the HIPO diagramming symbols is included. Reference 1.2.1 provides a complete description of the HIPO documentation technique.

2.0 DESIGN DETAILS

2.1 Background

Systems development personnel should refer to the EVAL Functional Description, Reference 1.2.2, in conjunction with the documentation contained in this specification. The Functional Description details the evaluation models implemented by EVAL and discusses the specific functions that the software must perform. In addition, systems development personnel may wish to refer to the EVAL User's Manual, Reference 1.2.3.

2.2 General Operating Procedures

EVAL is a menu-driven system. That is, the system is designed to interact with the user by presenting a sequential hierarchy of menus and asking the user to respond by selecting one option from the current menu. If the user does not select one of the menu options, the system displays the previous menu. In this manner, the user moves up and down the hierarchy, as desired. Whenever data entry is required as a result of option selection, the system specifically requests the data and specifies the format.

The system is also designed to be generally forgiving of procedural errors by the user.

2.3 System Logical Flow

EVAL is a hierarchically structured, modular software system. The system structure and logical flow lends itself to presentation in the form of HIPO diagrams, which are contained in this document.

The main purpose of the HIPO diagrams is to provide, in a pictorial manner, the complete set of modular elements necessary to the operation of EVAL, including all input, output, and internal functional processing. This is done by displaying the input items necessary to the process step which uses them, defining the process, and showing the resulting output of the process step.

The HIPO documentation diagrams are designed and drawn in a hierarchical fashion from the main calling routines to the detail-level operation/calculation routines. Extended written descriptions are given below a HIPO diagram whenever it is deemed necessary.

A complete description and explanation of the symbolic notation used in the HIPO diagrams is given in Reference 1.2.1. An abbreviated legend for the symbols used in this specification is given in Figure 2-1. Note that:

- a. External subroutines are depicted partly in the Process block and partly out. Internal subroutines are always shown within the Process block.
- b. Overview diagrams show general inputs and outputs only, whereas detail/subroutine-level diagrams show specific input/output tables and/or displays.
- c. Rectangular boxes inside the Input/Output block areas are generally used to denote single data items. Two or more boxes are grouped to show that several data items are input/output.
- d. Rectangular boxes inside the Process block indicate repetitive subprocesses.

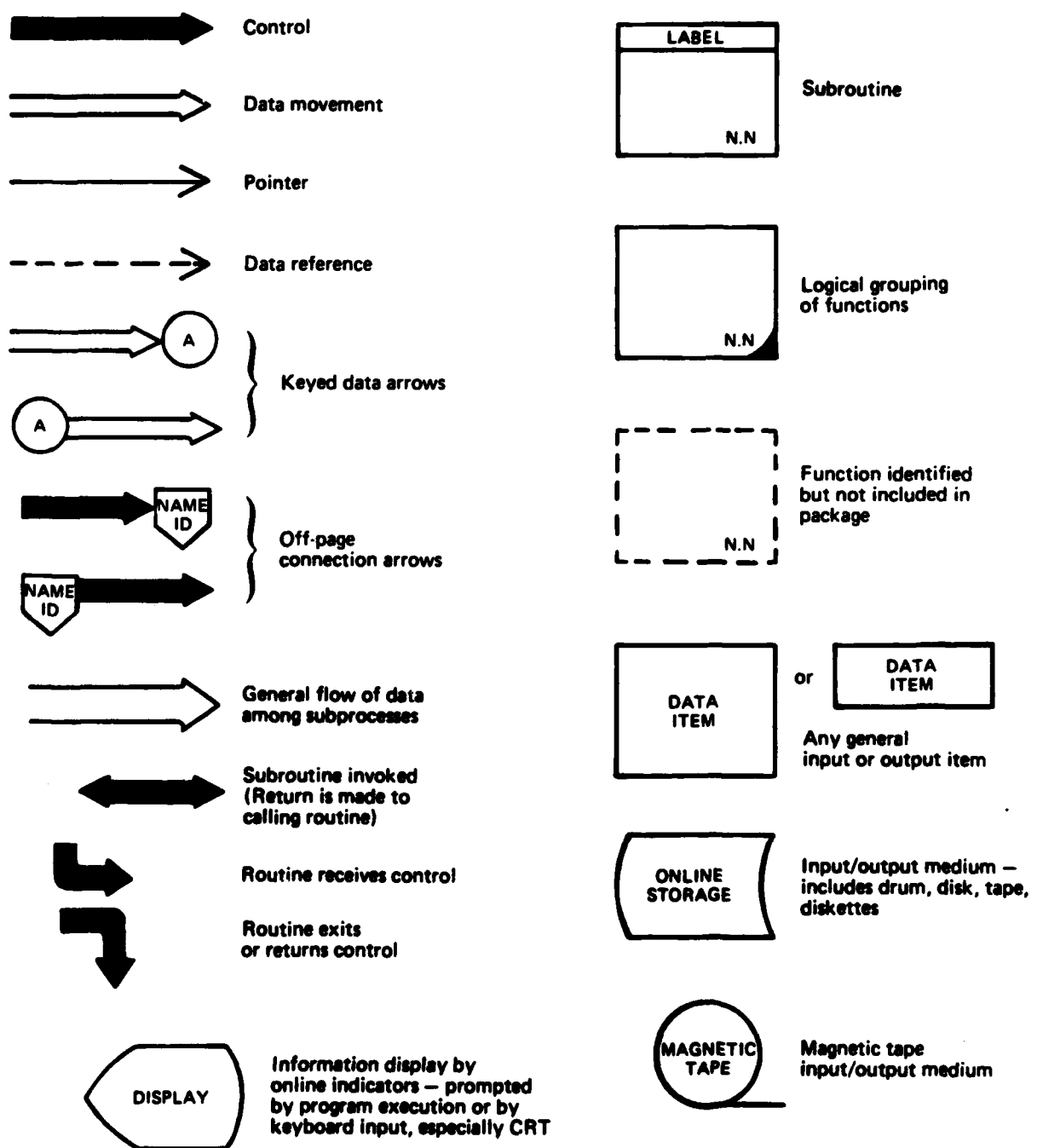


Figure 2-1
LEGEND OF HIPO SYMBOLS

The HIPO diagrams appear in the next section, which completes the system specification.

2.4 HIPO Documentation

The HIPO diagram identification numbers and figure numbers used in this section stand alone; i.e., they start with 1.0, increase hierarchically, and are independent of the numbering scheme used to this point in this document.

The EVAL software consists of two separate subsystems: STRUCTURE and RUN. Figure 2-2 is the system overview chart. The STRUCTURE subsystem is used to create a new evaluation structure or to revise an existing structure. The RUN subsystem is used to specify importance weights and utilities and to display the results of an evaluation model.

Figure 2-3 is a subsystem structure chart for the STRUCTURE subsystem. It represents the overall program logic flow in a visual table of contents. The Visual Table of Contents diagram shows the hierarchical structure, the functional description labels, and the diagram (chart) identifiers of the functions performed by STRUCTURE. Similarly, Figure 2-4 is a visual table of contents diagram for the RUN subsystem.

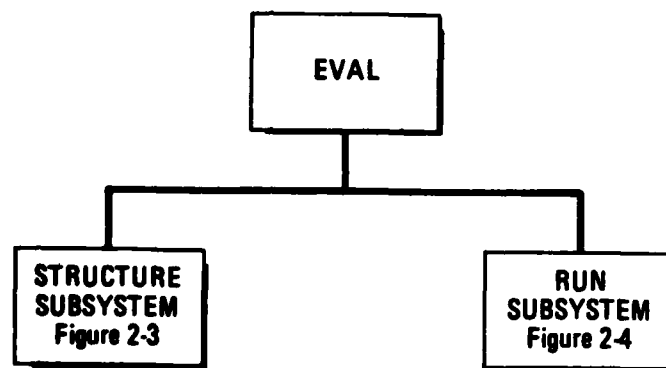


Figure 2-2
EVAL SYSTEM OVERVIEW

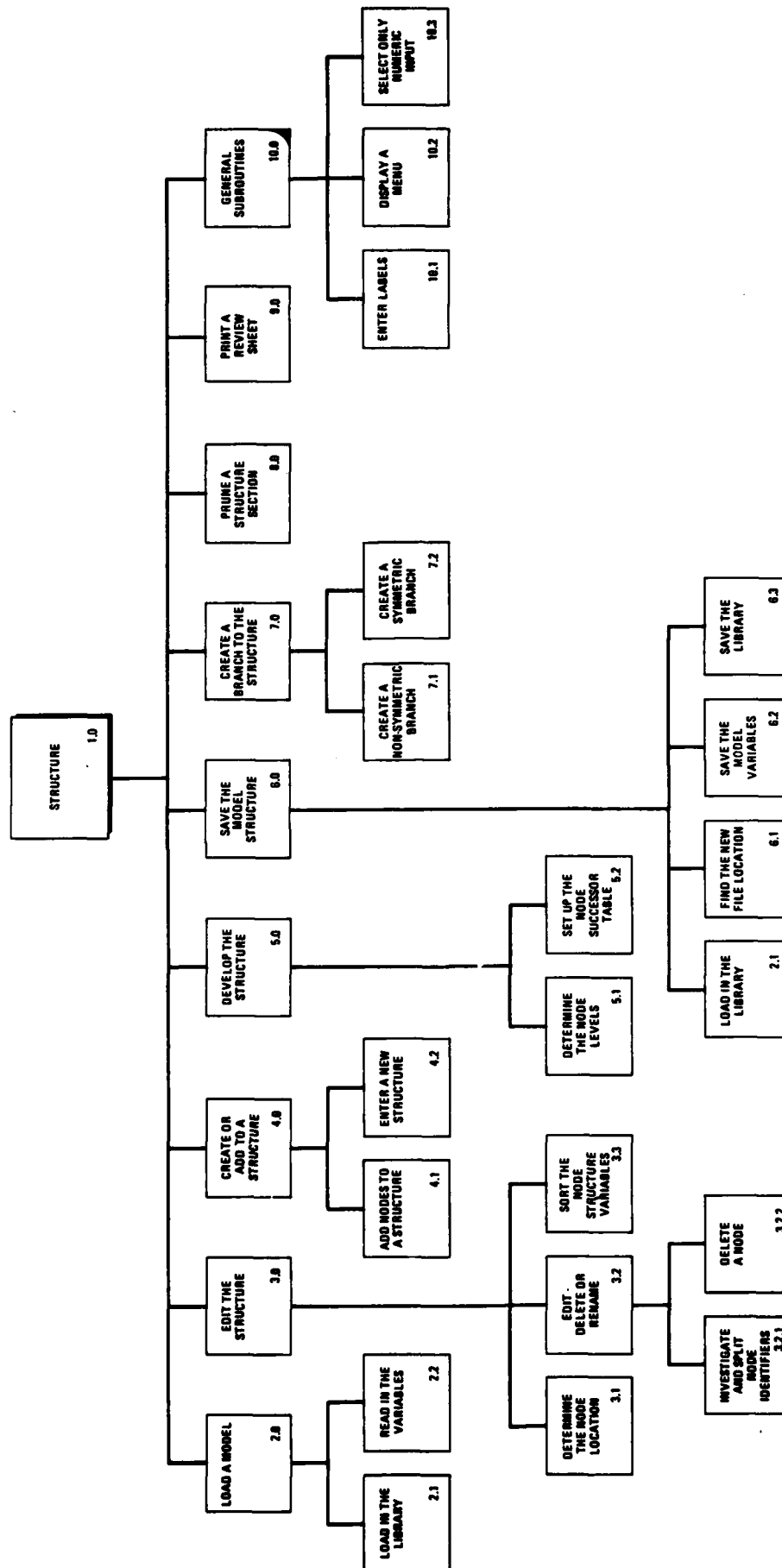
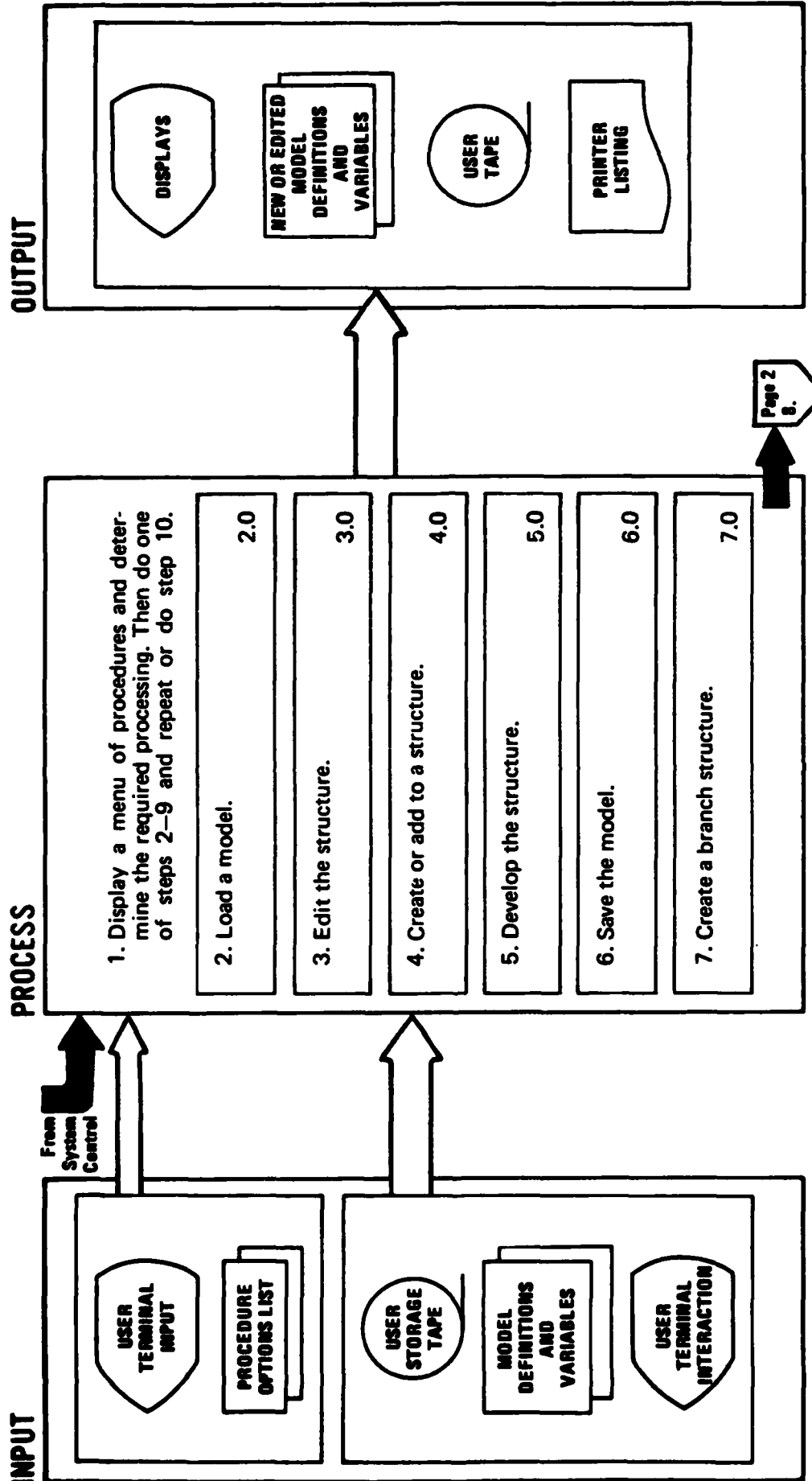


Figure 2-3
STRUCTURE VISUAL TABLE OF CONTENTS



Extended Description

The list of program procedures is displayed so that the user may select the next process to be performed. The list is displayed in menu format which allows the association of position numbers with different options in the list.

1. The user is prompted for a choice of operations. The chosen procedure is invoked via one of steps 2-9. If the user responds with blank or null input, then step 10 is executed.
2. The existence of EVAL/STRUCTURE models on tape (storage) is determined and a selected model is read.
3. The structure (or model) currently defined by the program variables may be

4. A new structure may be entered via user interaction or nodes may be added to an existing structure.

5. This step causes the completion of the model structure by setting up variables which interface with the RUN program. This step should always be performed before step 6.

6. The currently defined model structure may be stored via this step.

7. A branch or subtree may be defined and later added to a structure in procedure 4.

INPUT

PROCESS

8.0

8. Prune a section.

9.0

9. Print a review sheet.

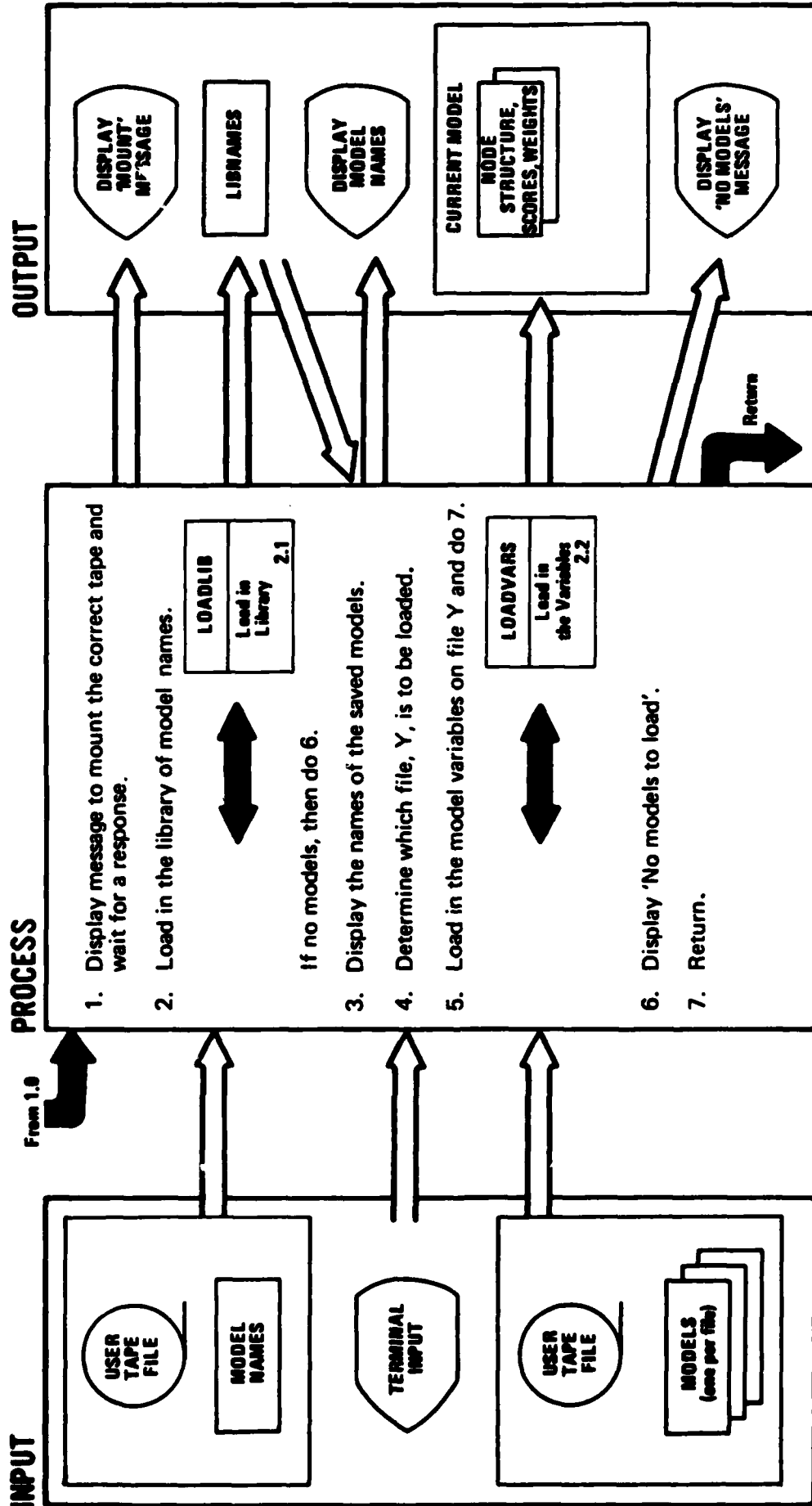
10. Terminate the session.

Return
to
System

OUTPUT

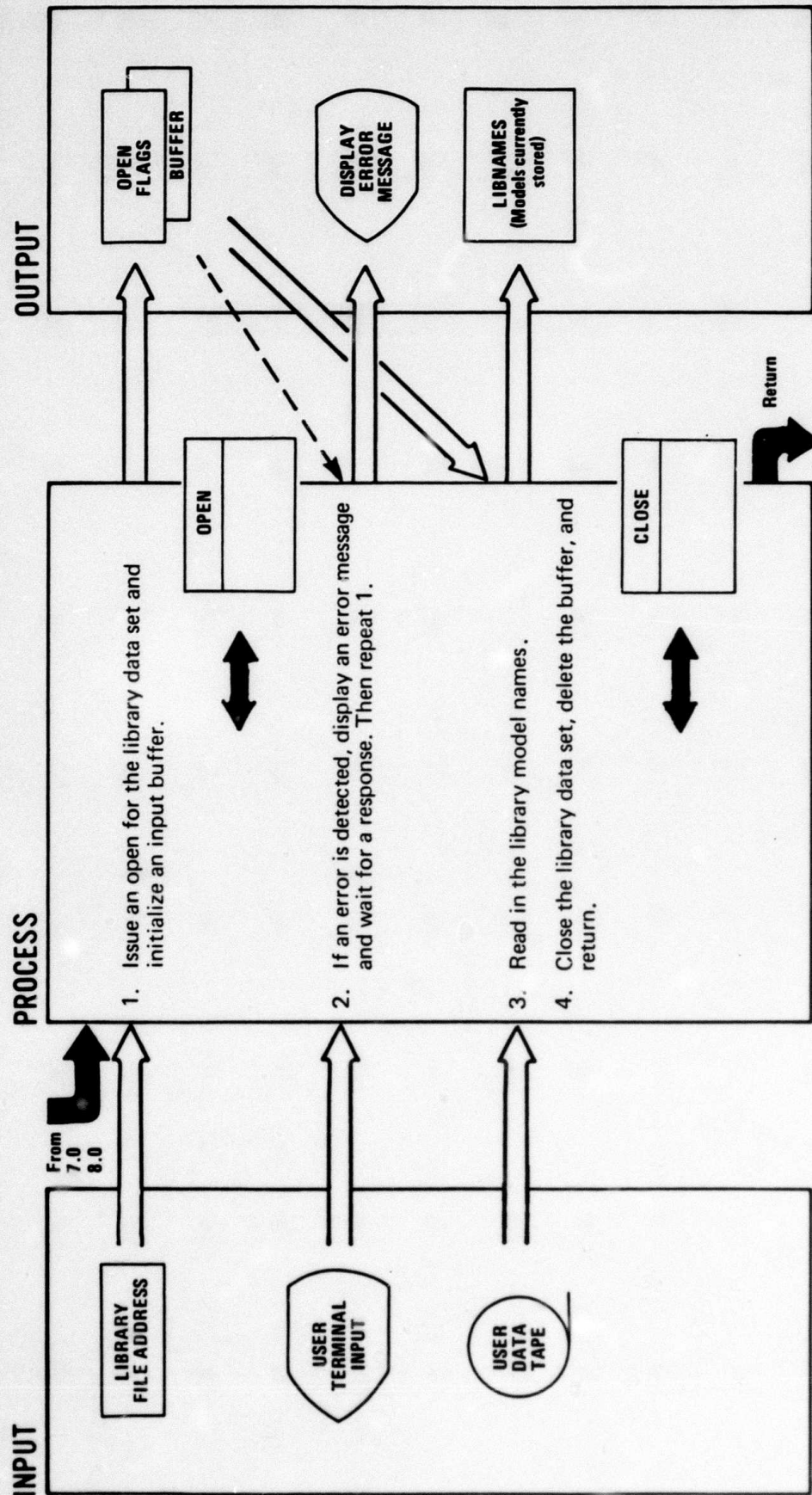
Extended Description

8. Groups of nodes may be deleted from the currently defined structure.
9. A printout of the structure as it is currently defined is obtained.
10. The program ends here: a restart option will cause step 1 to be executed again. When a session is terminated, all branch structures or subrees defined are deleted.



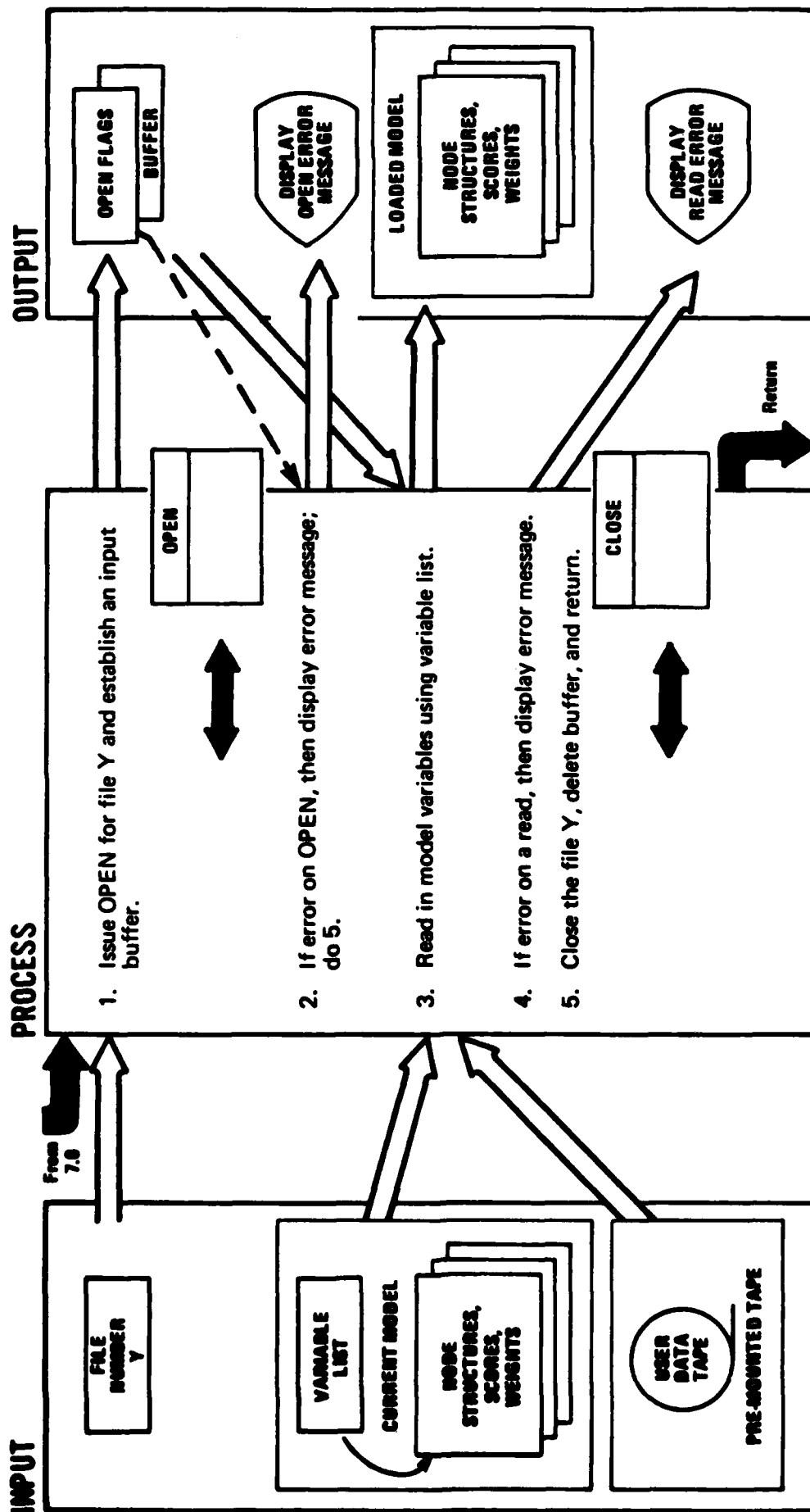
Extended Description

1. The user may have many tape files on which formatted models are stored. In this step, the user is prompted for a response indicating the desired tape is mounted and online.
3. The names of the models existing on the mounted tape are displayed in list or MENU format so that the user may select a model for loading.
4. The user is prompted for a model selection: the response may be the list item number or the model name. The requested model is stored in the same tape file as its position relative to the other model names in the displayed list.



Extended Description

- The library file of model names is available on each formatted data tape. The file is usually stored and retrieved as a character array and resides on the same device with model data and structure variables. A system **OPEN** command is needed to ensure that the data file is online and accessible for reading. An input buffer is needed and provides the link between stored information and program addressable information.
- The library model names are retrieved from storage. The character array used for holding these model names, **LIBNAMES**, is of a form which facilitates display; thus, the names may all be of equal character length.
- A system **CLOSE** command is issued to free the data file for later use.



Extended Description

3. A list of variable Names or identifiers is kept so that load and store routines will always process the variables in the same sequence order.

4. The Model variables retrieved from storage are used in all other program functions (see Diagram 1.0). The variables which must be loaded are the following:

- CUMULATIVE WEIGHTS
- NODE TYPES
- DATA LEVEL MASK
- AGGREGATE NODE INDICES
- SUCCESSOR TABLE
- SYSTEMS LABELS

1. The **OUTLINE TABLE** contains an element for each node in the model, sorted in increasing numerical sequence order. The value is an encoded representation of the node outline number supplied for a node when the model structure is created.

- OUTLINE TABLE
- NODE LABELS
- SCORES
- WEIGHTS

System/Program: STRUCTURE Name: LOADVARS

Diagram ID: 2.2 Description: Read in the Variables

Page: 2 of 3

INPUT

PROCESS

OUTPUT

Extended Description

2. The **NODE LABELS** contain descriptions (one per node in the same order as the outline table) of nodes that are supplied when the model structure is created.
3. **SCORES** is a numeric array which contains a set of values for each node of the structure. Each set of values consists of one number per system defined in the model.
4. **WEIGHTS** is a numeric vector containing the relative-importance values assigned to each node in the model structure. The elements must appear in the same order as the associated outline numbers. When a model structure is created, the vector is null or contains zeros.

5. For each element in the node outline table, there is an associated element in the **CUMULATIVE WEIGHTS** vector. The vector will contain the percentage of importance with respect to the entire model when all **WEIGHTS** have been entered.
6. The **NODE TYPES** are indicators of the type of calculation that is to be used in assessing **SCORES** and **WEIGHTS**.
7. The **DATA LEVEL MASK** indicates which nodes are at the data level (bottom level) versus the nodes that are aggregate or non-bottom-level nodes.
8. The **AGGREGATE NODE INDICES** contain the sequence number of elements in the model variables which correspond to only the aggregate nodes. An Aggregate node is a node which has one or more subsequent nodes contributing to it.

System/Program: STRUCTURE Name: LOADVARS

Diagram ID: 22 Description: Read in the Variables

Page: 3 of 3

INPUT

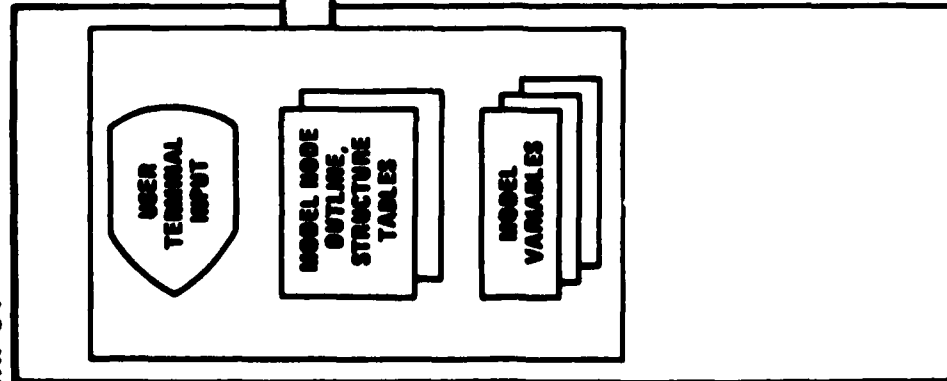
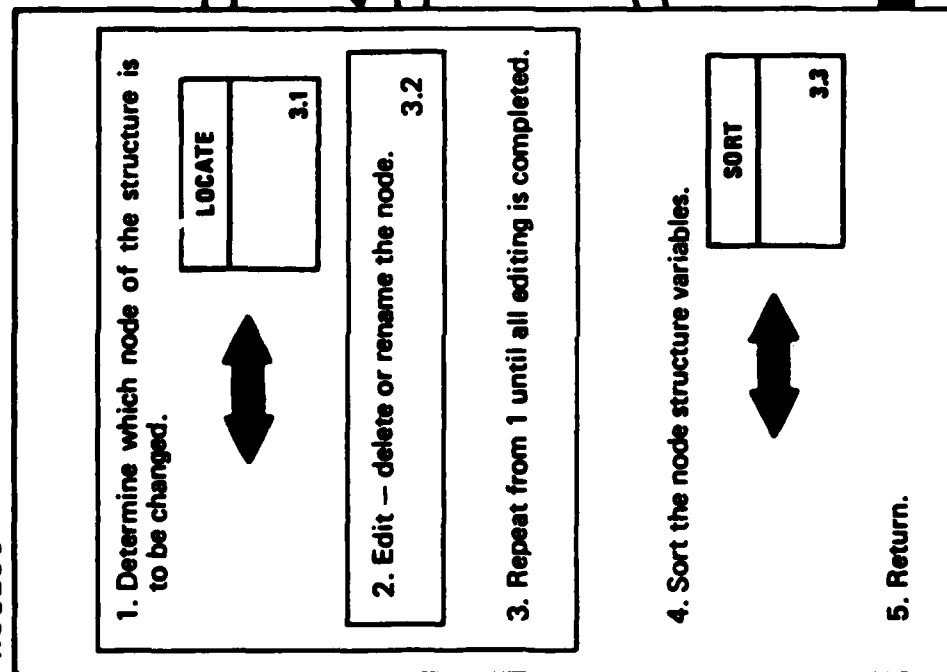
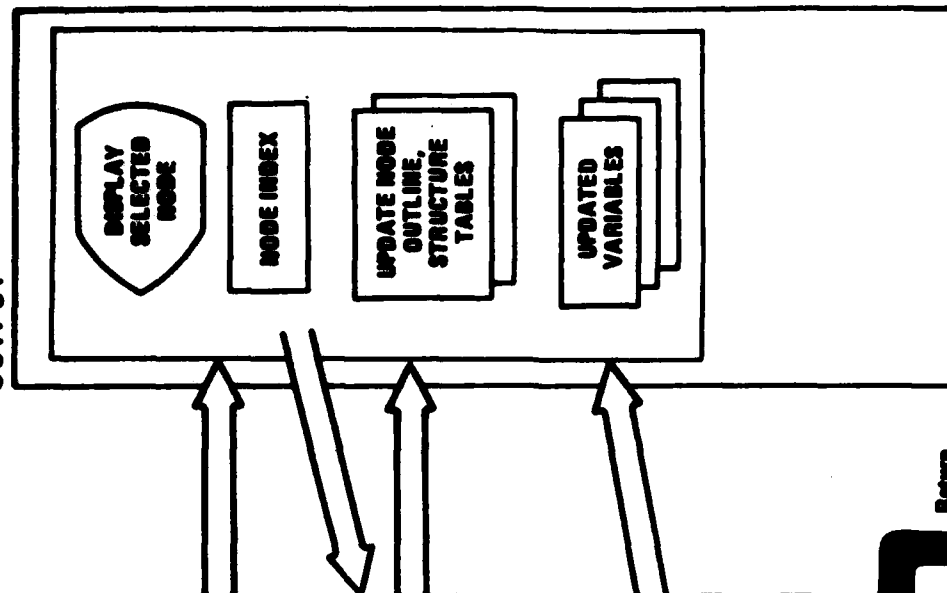
PROCESS

OUTPUT

Extended Description

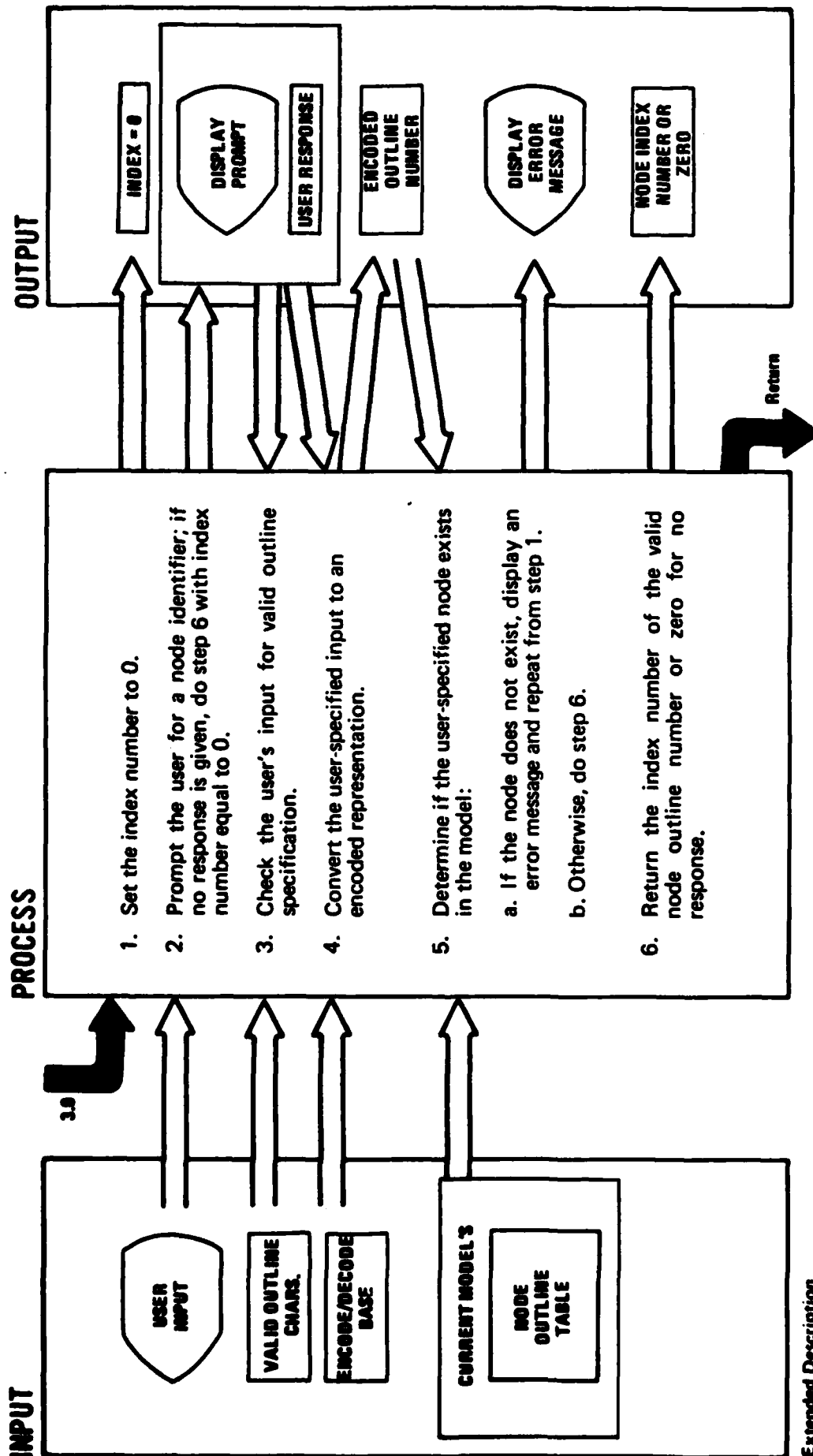
9. The SUCCESSOR TABLE is an array which contains, for each aggregate node, the set of indices of nodes which contribute to a node.

10. The SYSTEMS LABELS contain the user-specified character descriptions of the systems being evaluated.

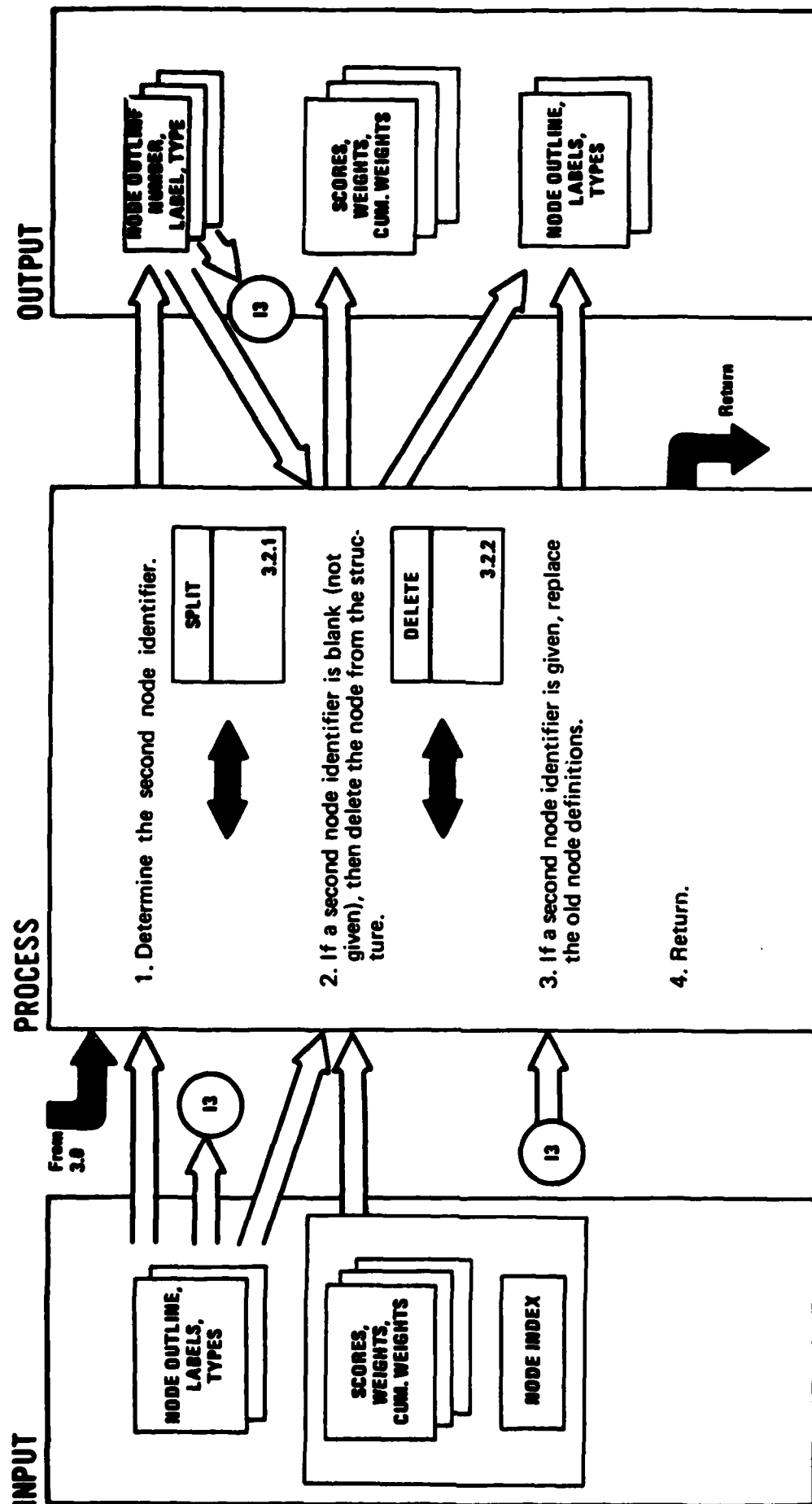
INPUT**PROCESS****OUTPUT****Extended Description**

This procedure will allow the deletion or renaming of nodes within an existing structure and operates on a single node at a time. If a group or subtree of nodes is to be deleted, the user should select the "Prune a section" procedure described in diagram 8.0.

1. The user is prompted for a node identifier. This identifier corresponds to the manner in which the node was named when it was placed in the structure. The outline number is a shortened form of the node's identification. An associated index number is determined which is relative to the node outline and structure tables.
4. The node structure variables are reorganized so that associated nodes are always grouped together after the structure has been edited.

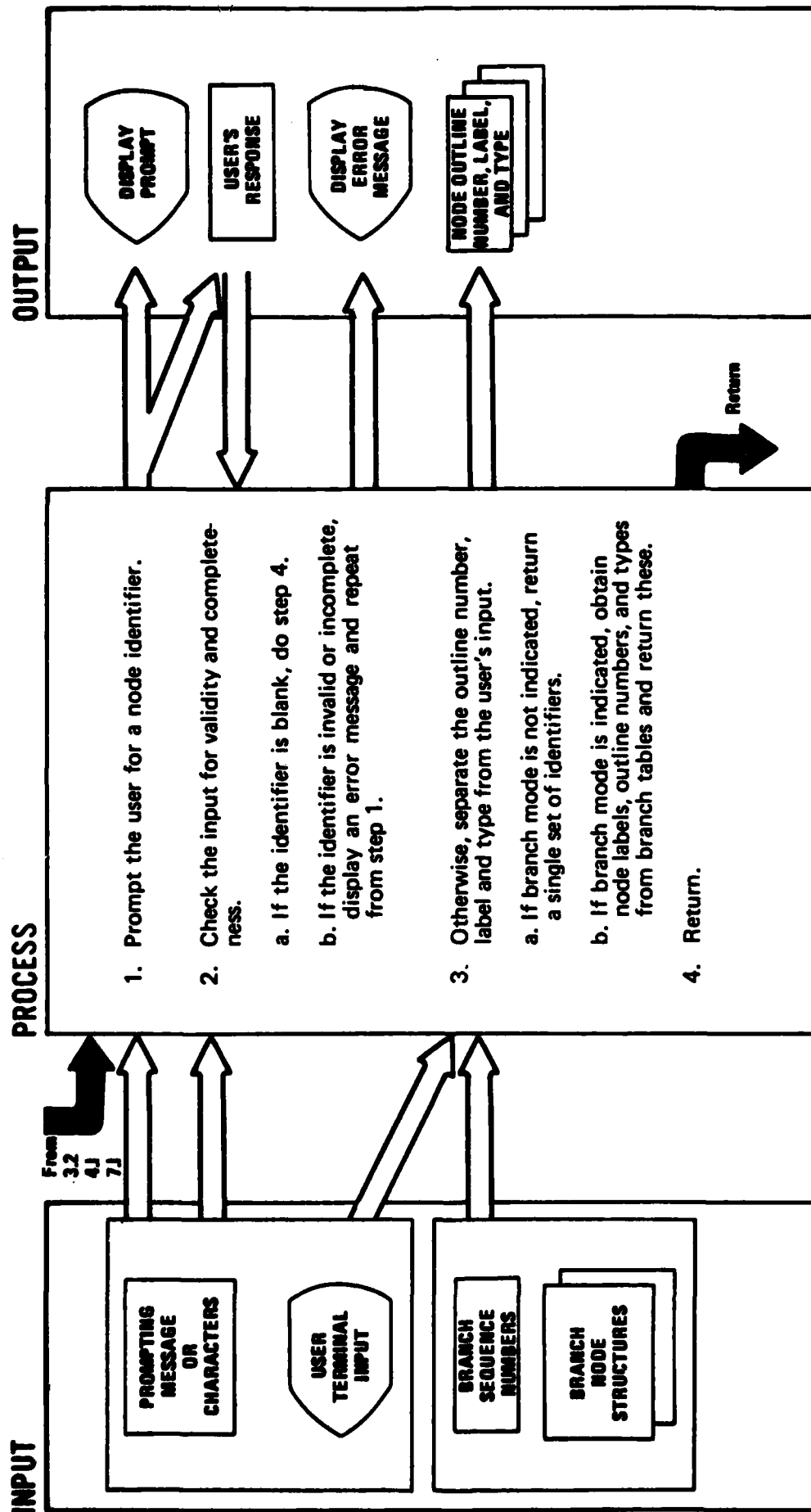


Extended Description
 5. The existing outline table is searched for a matching encoded outline number. It is the index into this table of the matching outline number which is returned to the calling routine in step 6.



Extended Description

1. The user is prompted for all node identification information — the node outline number, the node label or name and its type. (See diagram 2.2 for a description of these items.)
2. A null entry or blank response from the user indicates that the node is to be deleted from the current structure.
3. Replace the outline number, the node label and type in the appropriate arrays with the new ones.

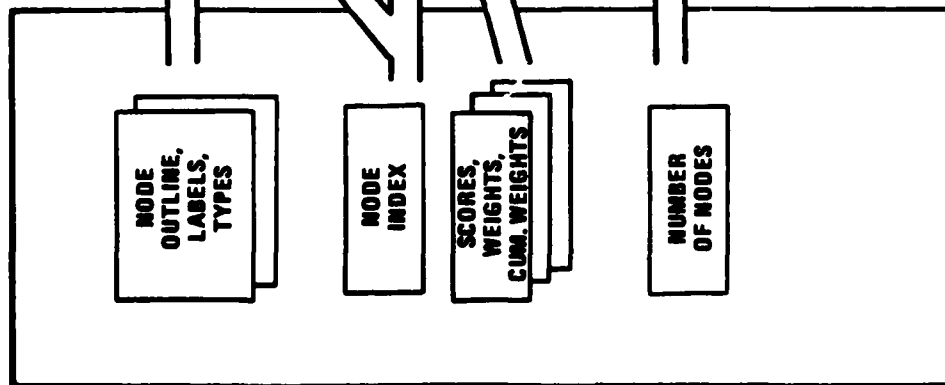
**Extended Description**

1. The user is required to input the identifying information for a particular node in either an existing structure or one that is currently being defined.
2. Proper node identification consists of an outline sequence number which has a hierarchical relationship to other nodes in the structure, a label or descriptive name, and a node "type" indicator. The three variables are usually entered with commas or some other punctuation separating each one from the other.

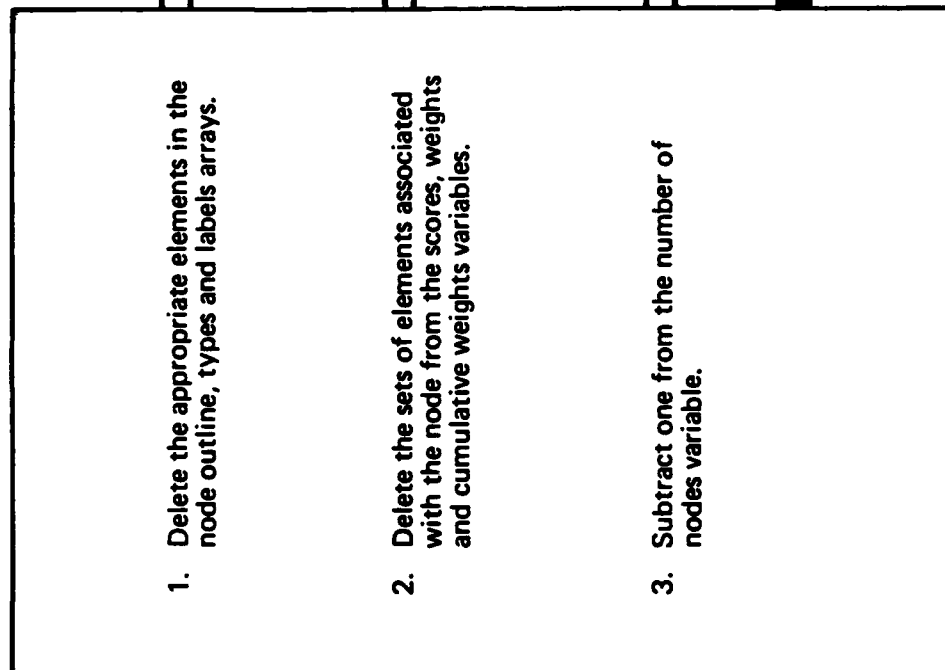
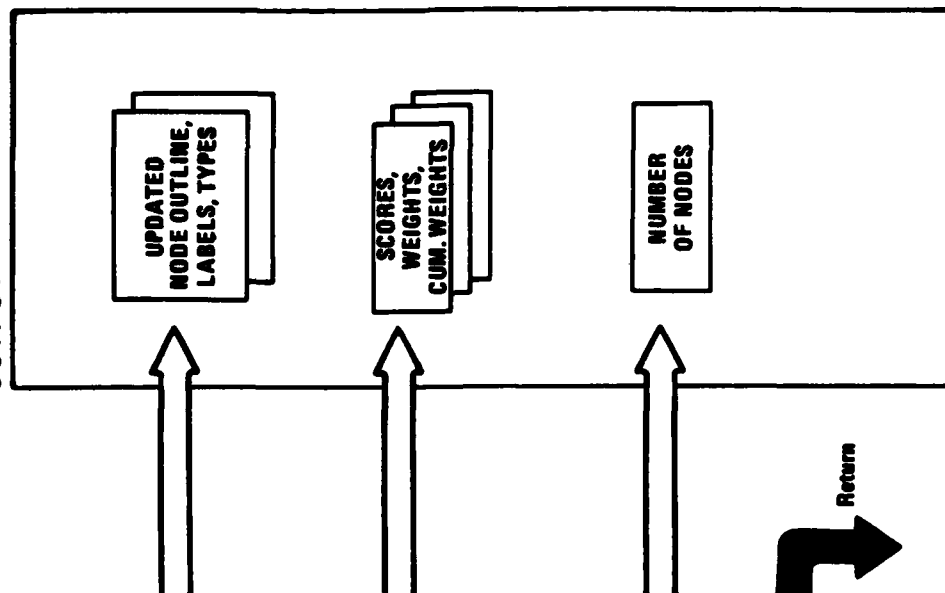
A special character, such as an asterisk (*) or pound sign (#), should be used to designate that a group or subtree is being specified. The special character would be the first in the input line of the user's response.

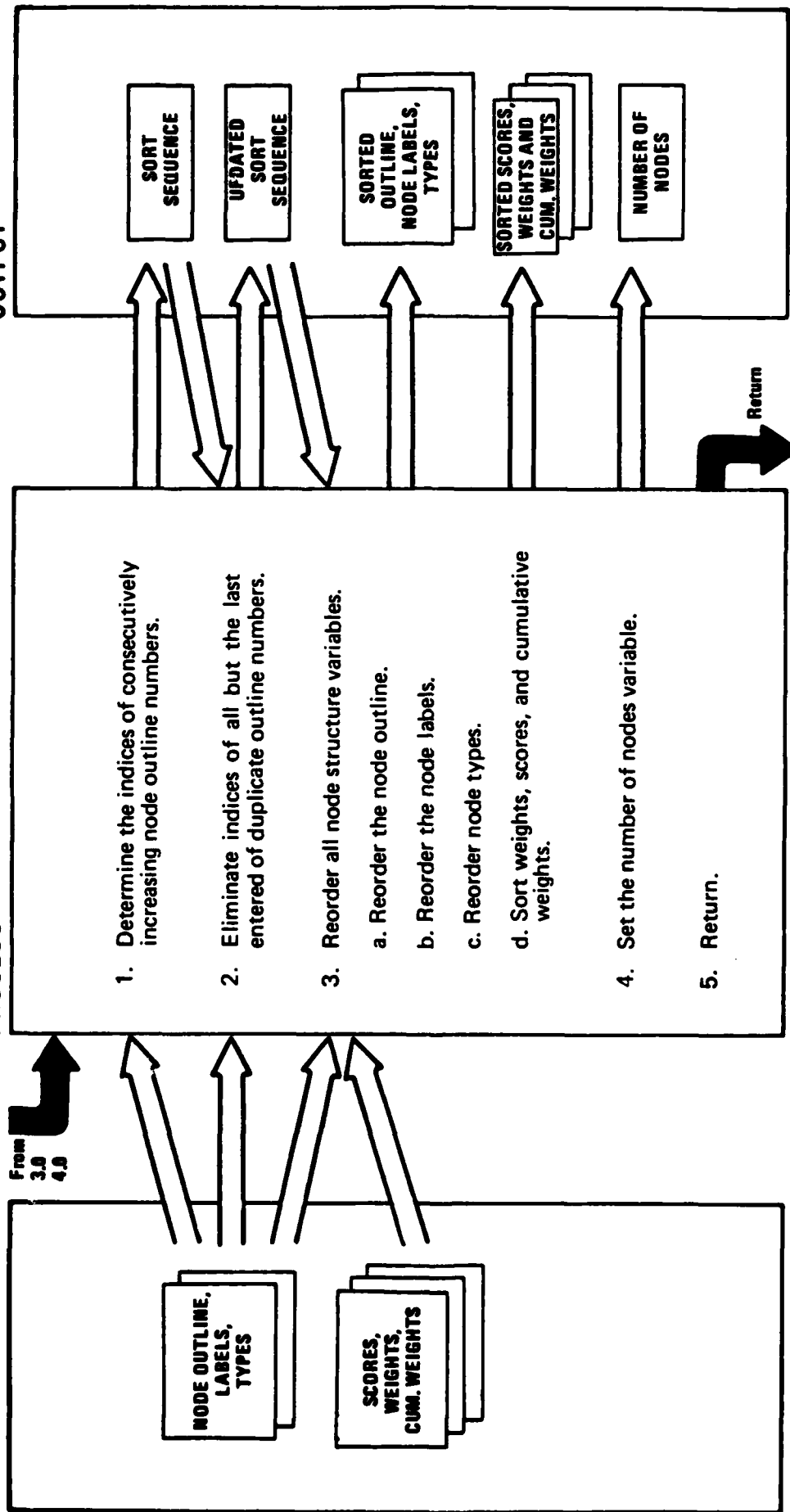
3. The outline number - numerically encoded to a sufficiently large number, the label, and type are returned as separate variables.

If a branch or subtree is being specified, the appropriate node labels, outline numbers and types are obtained from the branch structure tables. A group of encoded outline numbers, a group of labels and the group types are all returned to the calling routine. The new outline numbers have been encoded again to agree with the node after which the branch or subtree is being placed in hierarchical fashion.

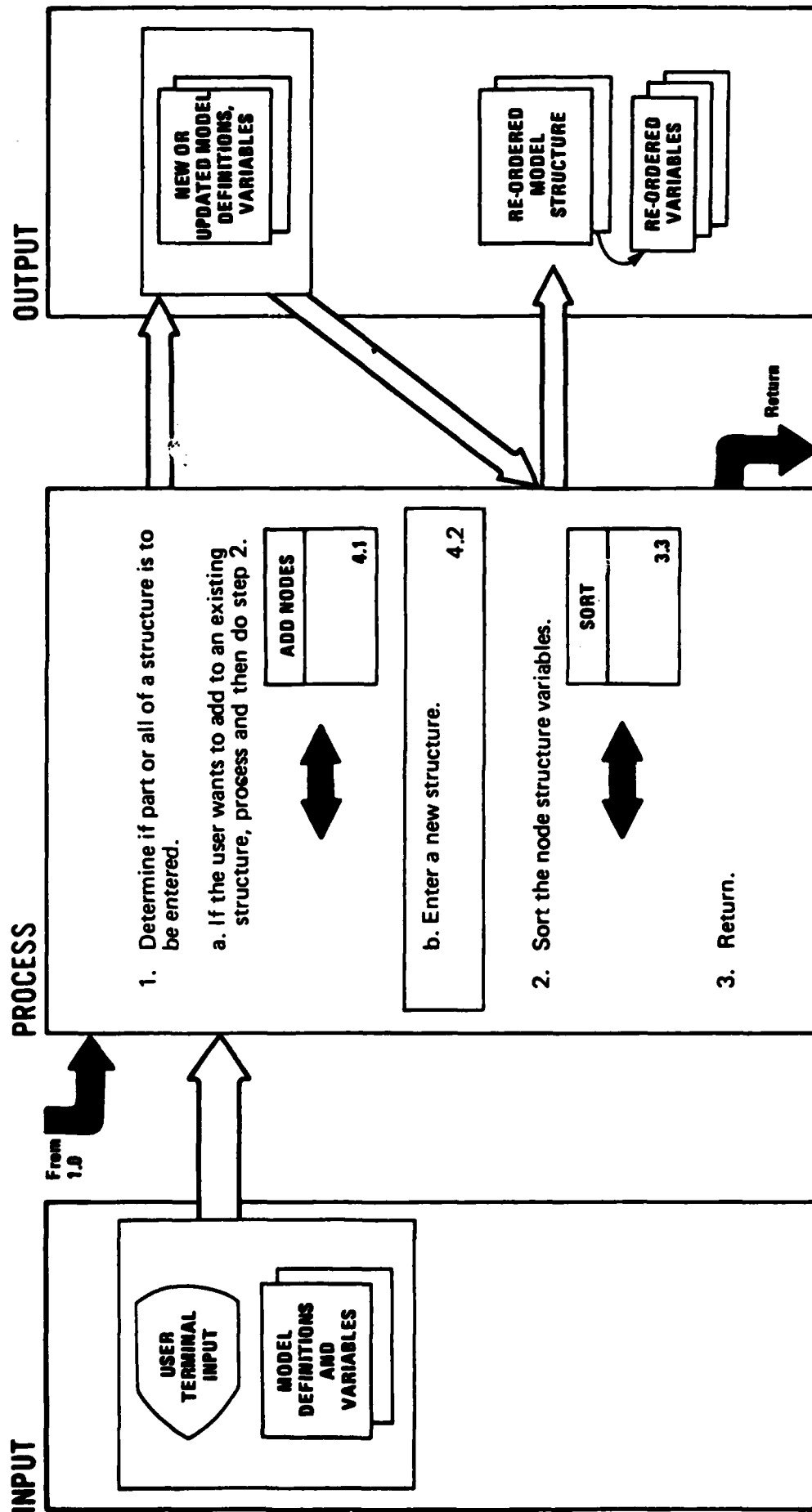
INPUT

From
3.2

PROCESS**OUTPUT**

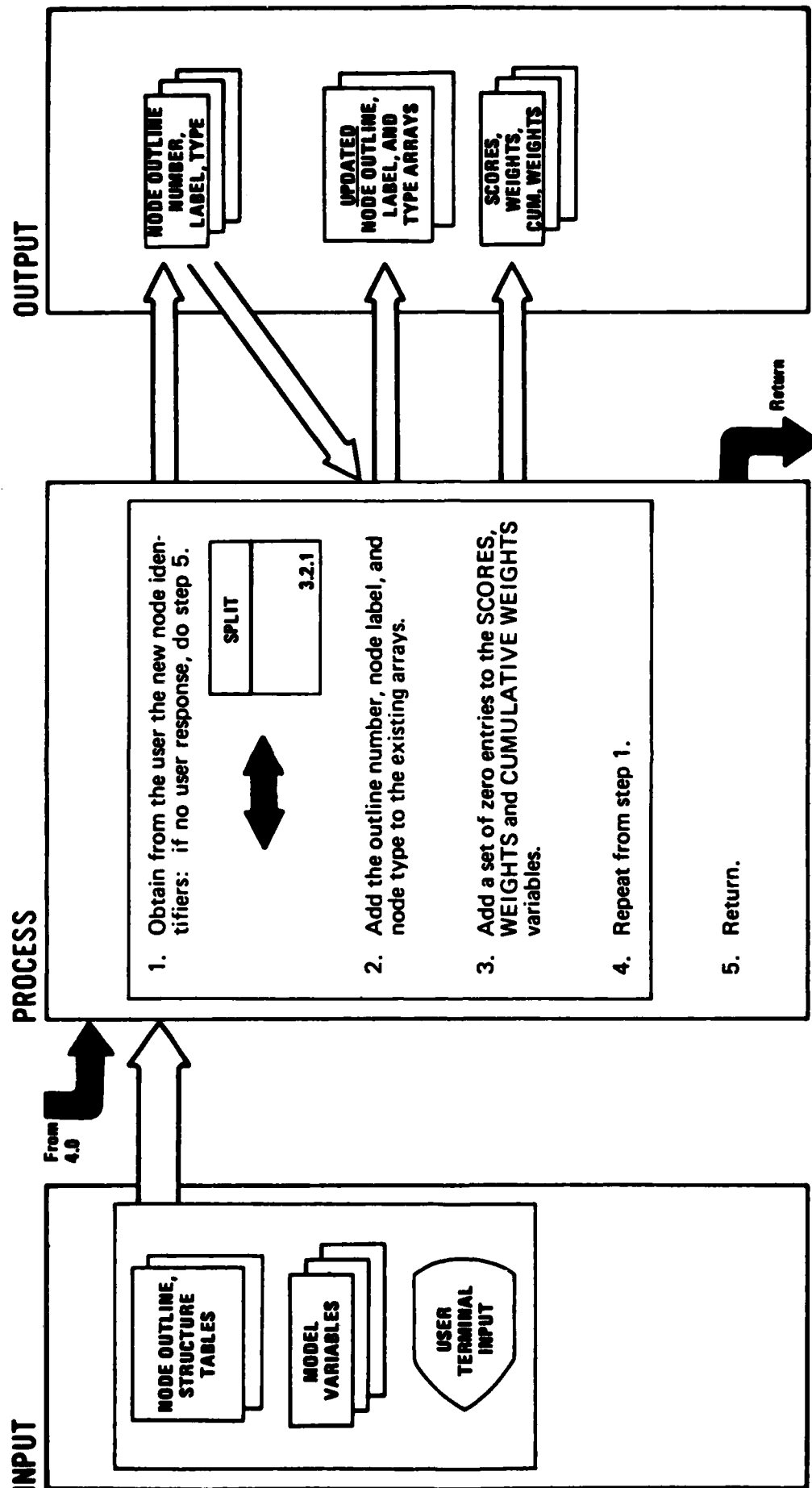
INPUT**PROCESS****OUTPUT****Extended Description**

1. The relative indices or locations in the numerically encoded set of outline numbers in increasing value are determined. These indices constitute the sort sequence and will be used to rearrange the structure variables.



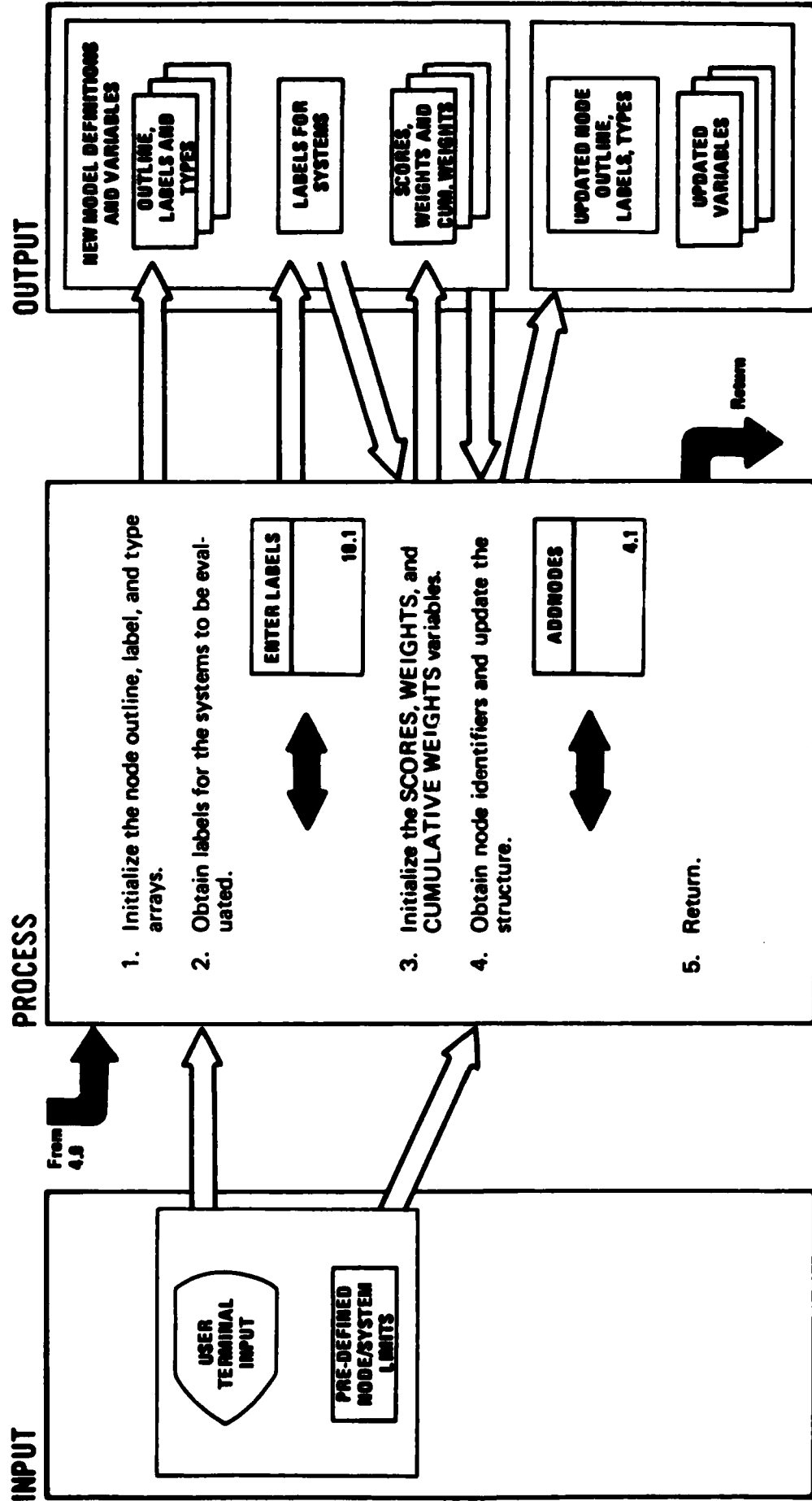
Extended Description

- Request a "yes" or "no" response directly from the user to determine whether a new structure is to be entered or nodes are to be added to an existing structure.
 - If a new structure is entered, all currently defined variables of the old structure are deleted.
- An explanation of the sorting function is given in diagram 3.3 of the STRUCTURE System Specifications.



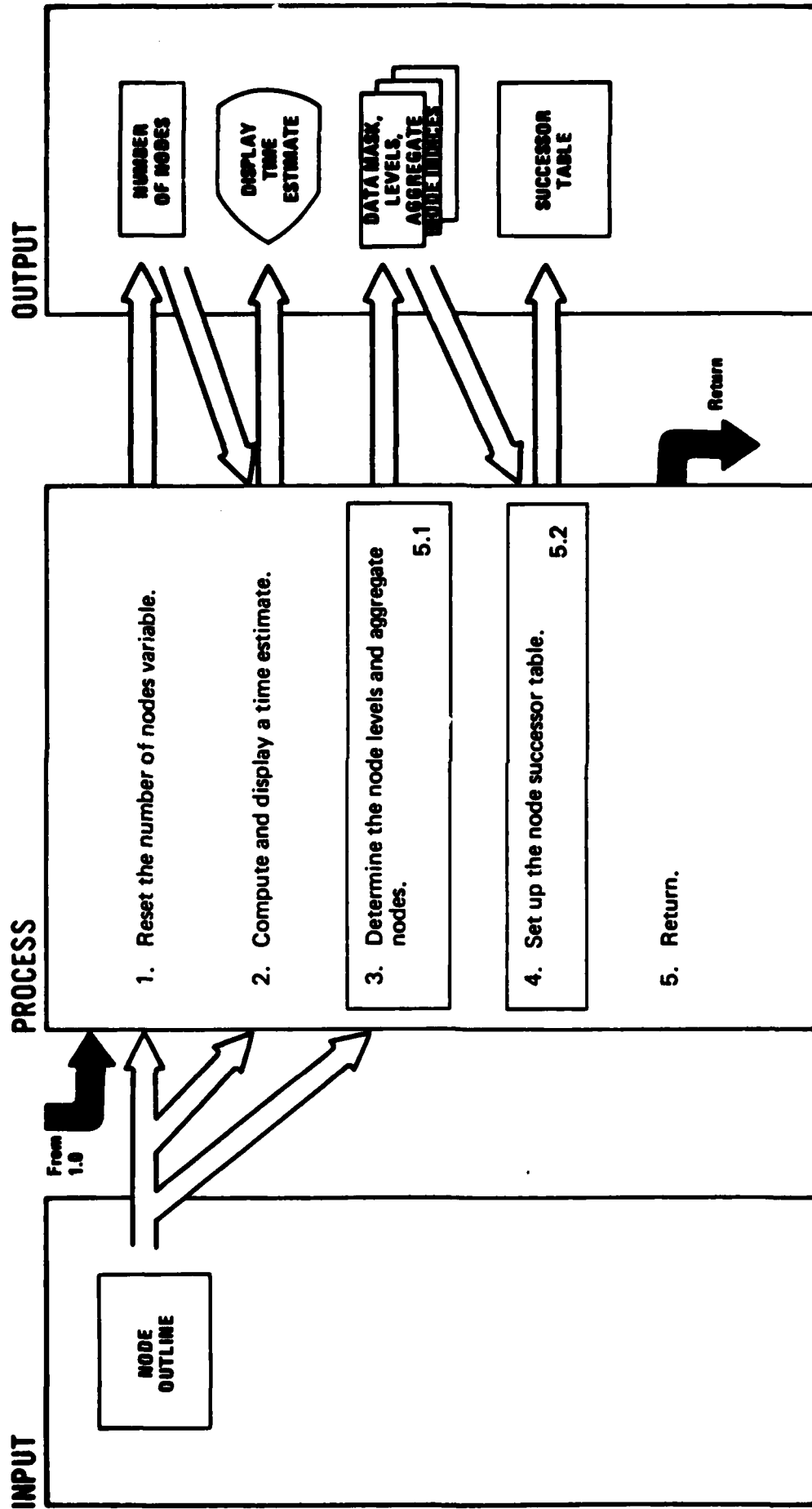
Extended Description

2 - 3. Additions to previously initialized or existing variables are accomplished by extending the arrays such that the corresponding orders of associated labels, scores, types, weights and decoded outline numbers are the same.



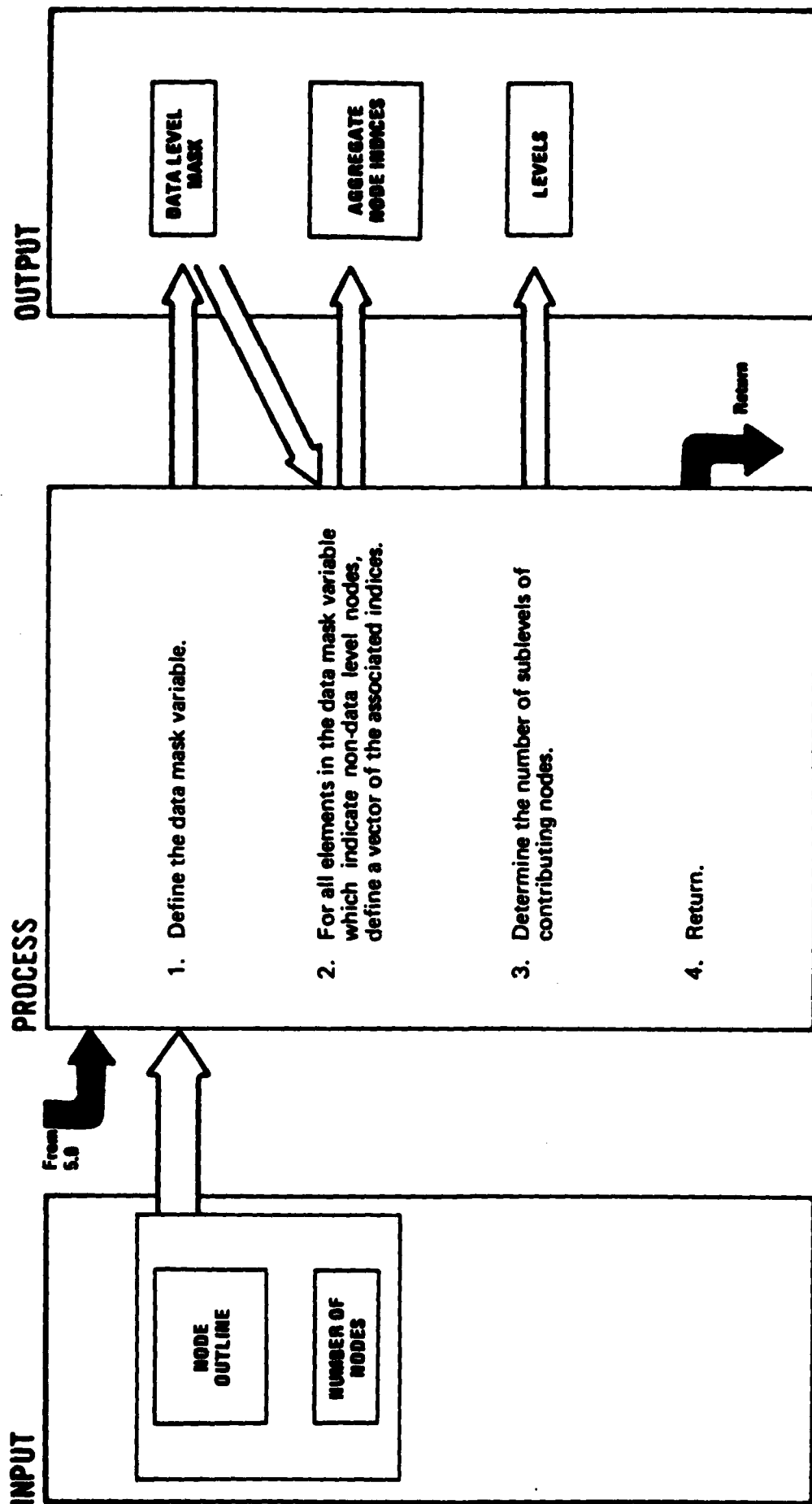
Extended Description

1. Initialization is caused by establishing null or blank vectors for the specified variables.
2. Labels for the systems to be evaluated are obtained from later storage and for the determination of the length of any set of SCORES.
4. The user is prompted for input which will be used to define a hierarchical tree structure described by outline numbers, labels and types of nodes within the structure.



Extended Description

1. The number of nodes is equal to the number of entries in the outline array.
2. A rough estimate of the amount of time required to perform the developing operation may be displayed. The estimate is derived from the number of nodes in the model.
3. The data level mask indicates which nodes in the model are at the data level and which nodes are aggregate nodes. The aggregate node indices are indices into the node outline of nodes which are not at the data level. The **LEVELS** variable shows how far away a particular node is from the lowest level.
4. The successor table provides a set of contributing node indices for each aggregate node in the same order as aggregate node appearance in the outline.



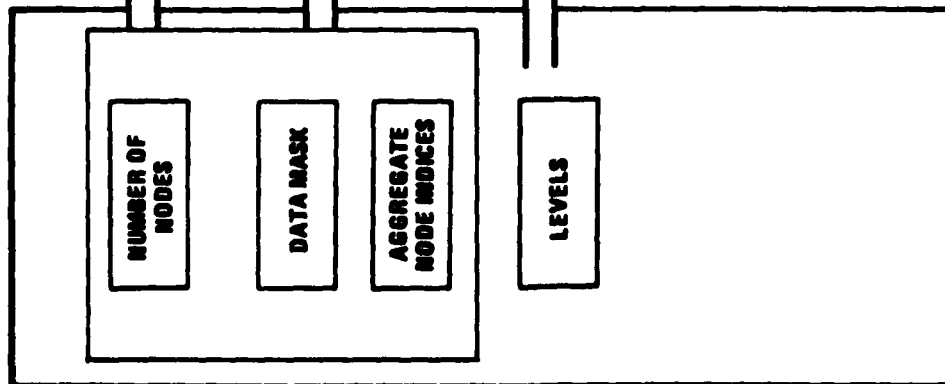
Extended Description

1. For each node in the model outline, an element is placed in a vector to indicate that node is a data level node or that it is an aggregate node having other contributing nodes.

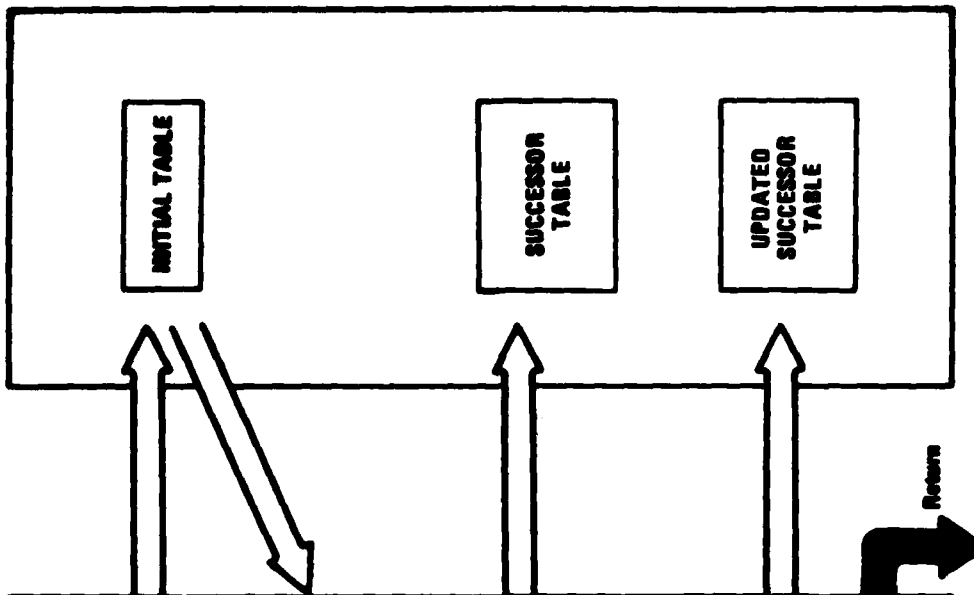
The indicator may be 0 for data level and 1 for the aggregate level or vice versa.

2. The data level mask indicator setting for each node in the outline is used to determine the aggregate node indices - indices into the node outline.

3. The farthest element or data level node from the topmost node is determined. The topmost node is assigned the number of levels between it and the data level farthest away (the depth of the path with the most sub-level tree branches). All other nodes are assigned a value equal to the top-level's minus its distance (number of levels) from the top.

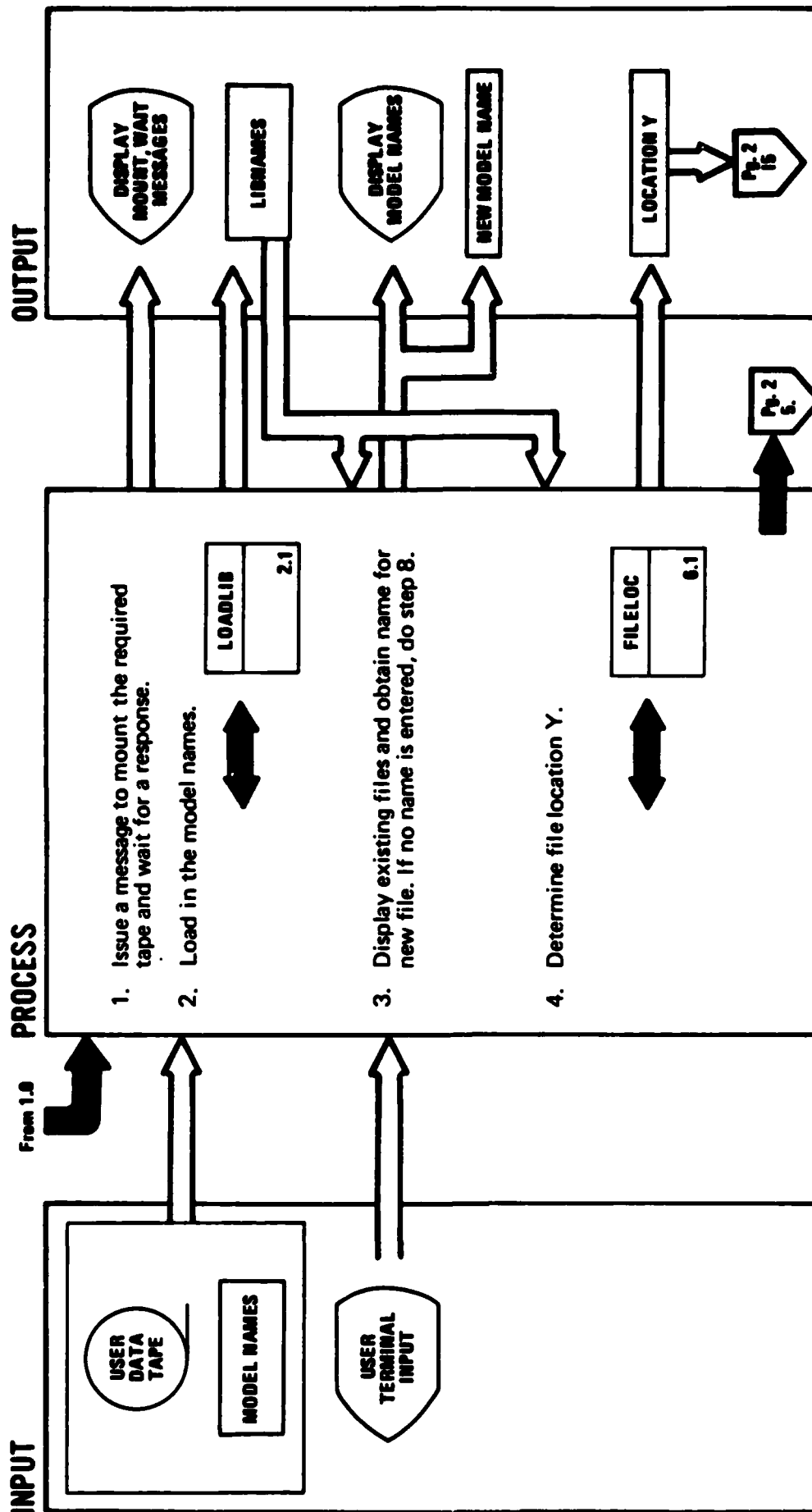
INPUT**PROCESS**

1. Initialize an array of maximum size for the successor table.
2. For each node of the model, check its associated element in the data mask.
 - a. If the data mask element indicates a data-level node, do step 2.c.
 - b. If the data mask element indicates an aggregate node, add a row to the successor table of all the contributing node indices.
 - c. Repeat from step 2 for the next node until all nodes have been checked.
3. Shrink the table from maximum size as appropriate.

OUTPUT**Extended Description**

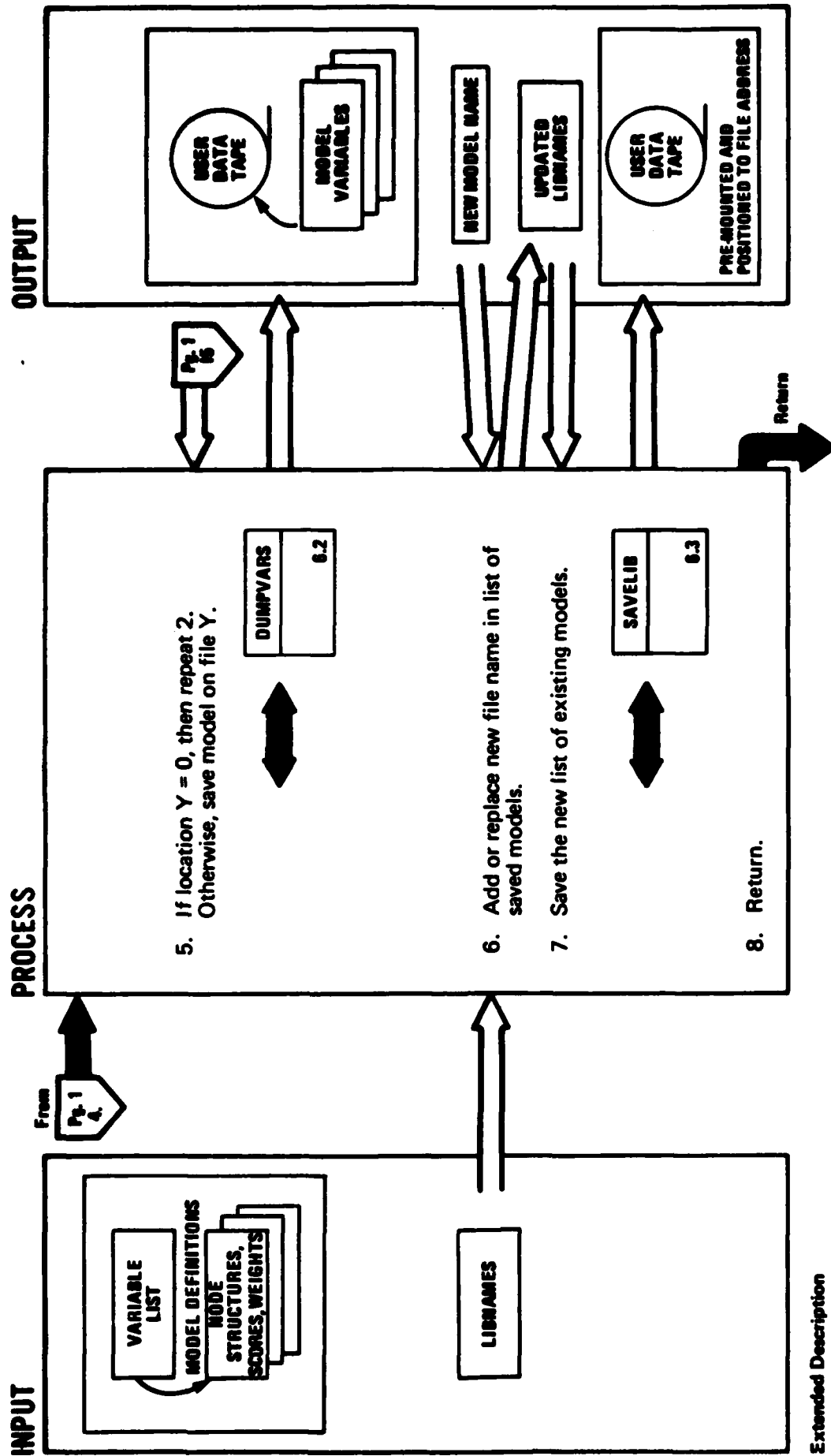
1. The maximum size table is prescribed by the number of aggregate nodes and the predefined limit to the number of contributing nodes on any single level.
2. This procedure steps through the data mask variable in sequential order: the contributing nodes of the topmost aggregate node will be added to the successor table first.
- 2.b. If the nodes' associated data mask element indicates an aggregate node, then the contributing nodes are all the nodes which follow in sequential order that have an associated LEVELS number that is less than the selected nodes LEVELS numbers, provided these nodes occur before any node with equal or higher LEVELS number.

3. Since the number of elements in any set of contributing nodes may be less than the predefined limit, the number of columns (or characters) in the table may be diminished.

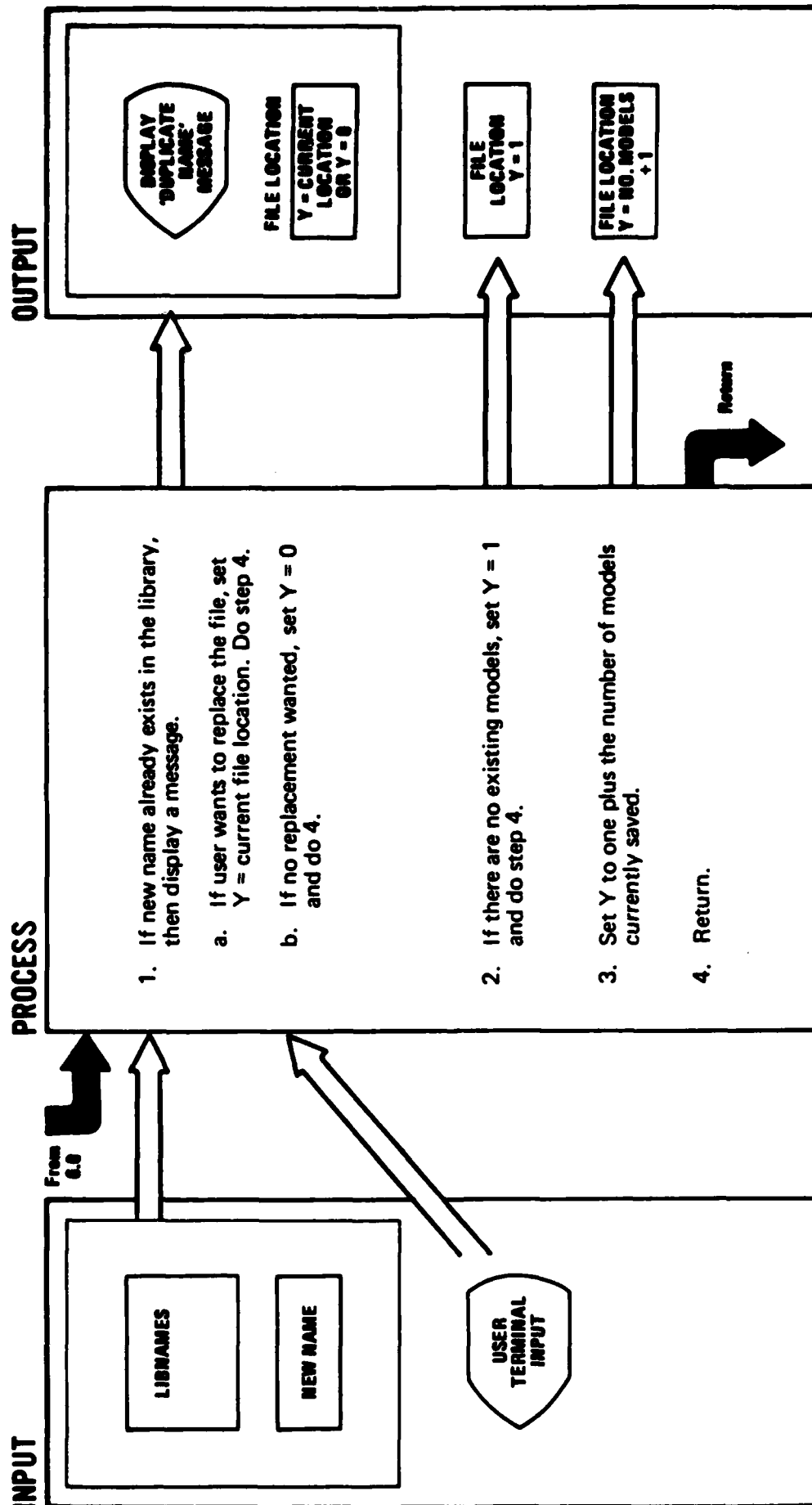


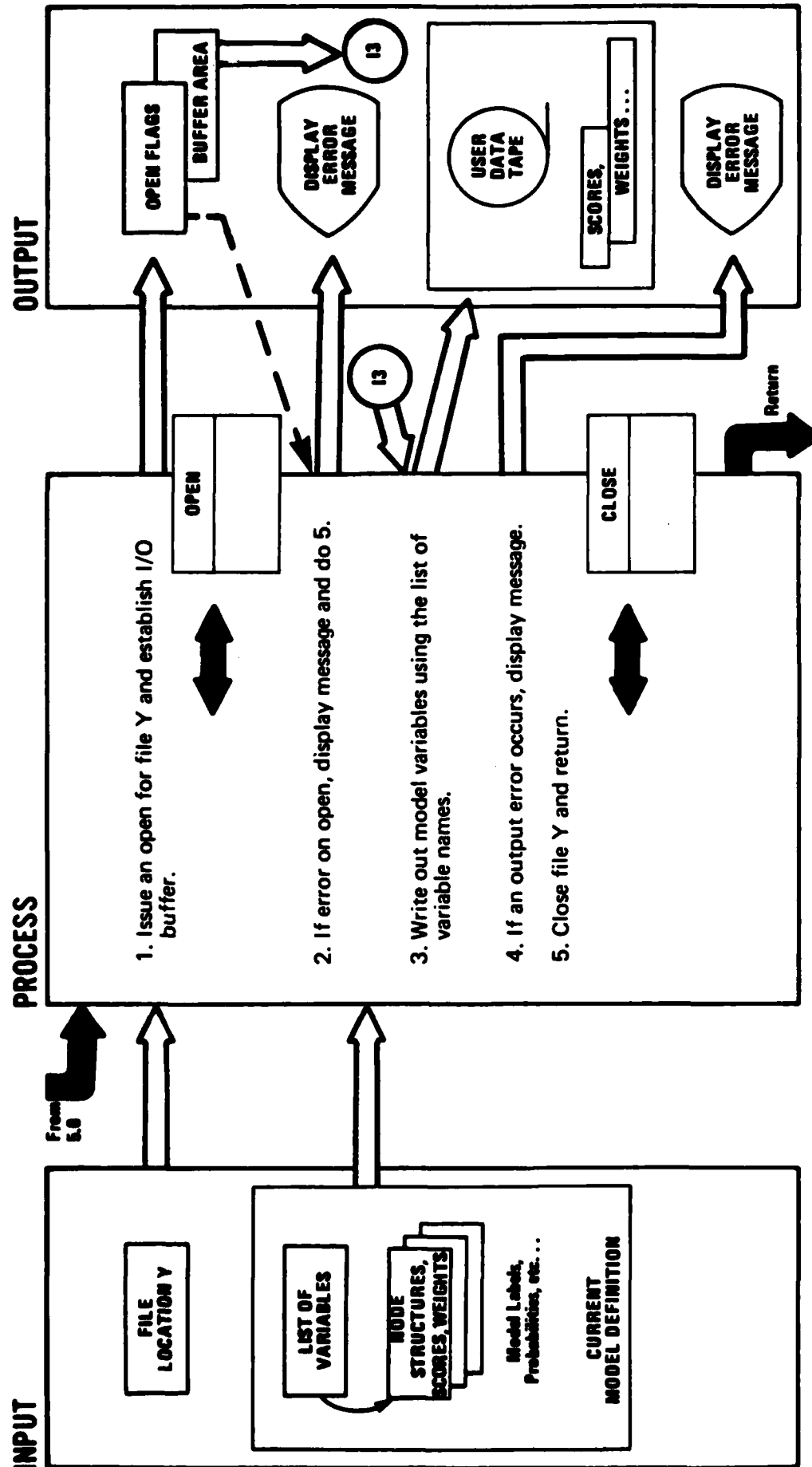
Extended Description

1. The computer program prompts for an indication that the desired storage file/device has been selected and placed online. Any response from the keyboard causes processing to resume.
4. The existing file structure and the amount of available space on the data tape are checked along with the user specification to determine where the model variables are to be stored.

**Extended Description**

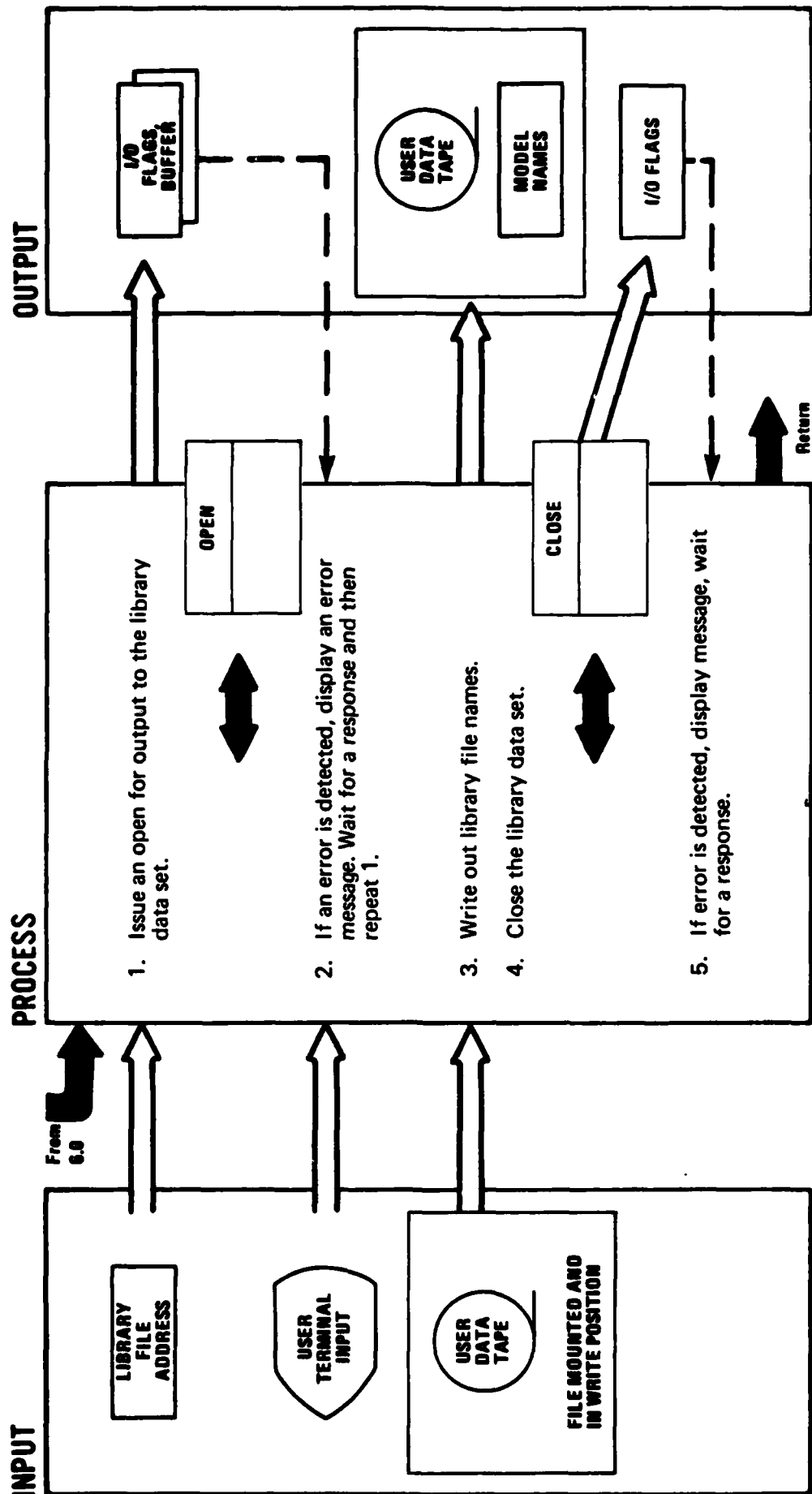
6. The library name list is updated to include the new file. The new model name's position in the LIBNAMES array must be the same relative position to other models stored on the device.

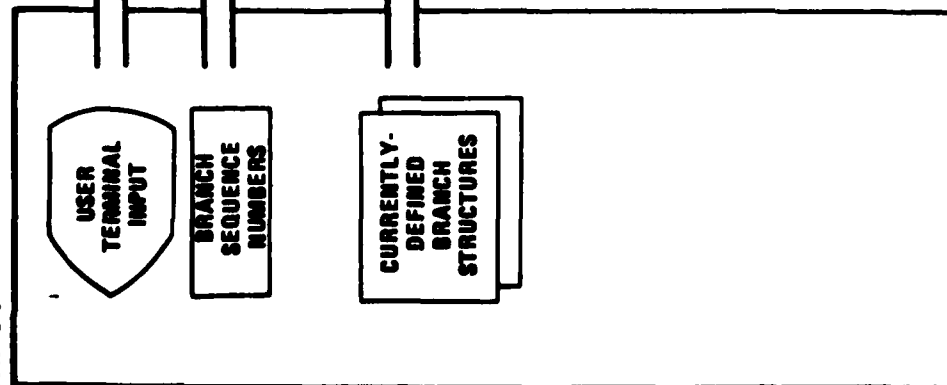




Extended Description

1. The file location Y is used to determine an exact storage position on the selected device.
3. The list of variable names is identical to the list of names used to Load a Model (see diagram 2.2)

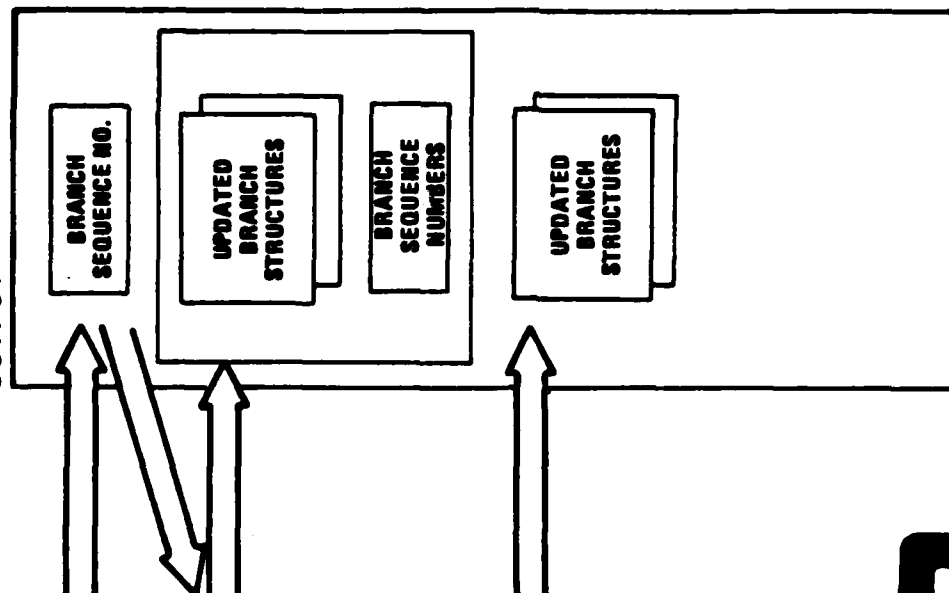


INPUT**PROCESS**

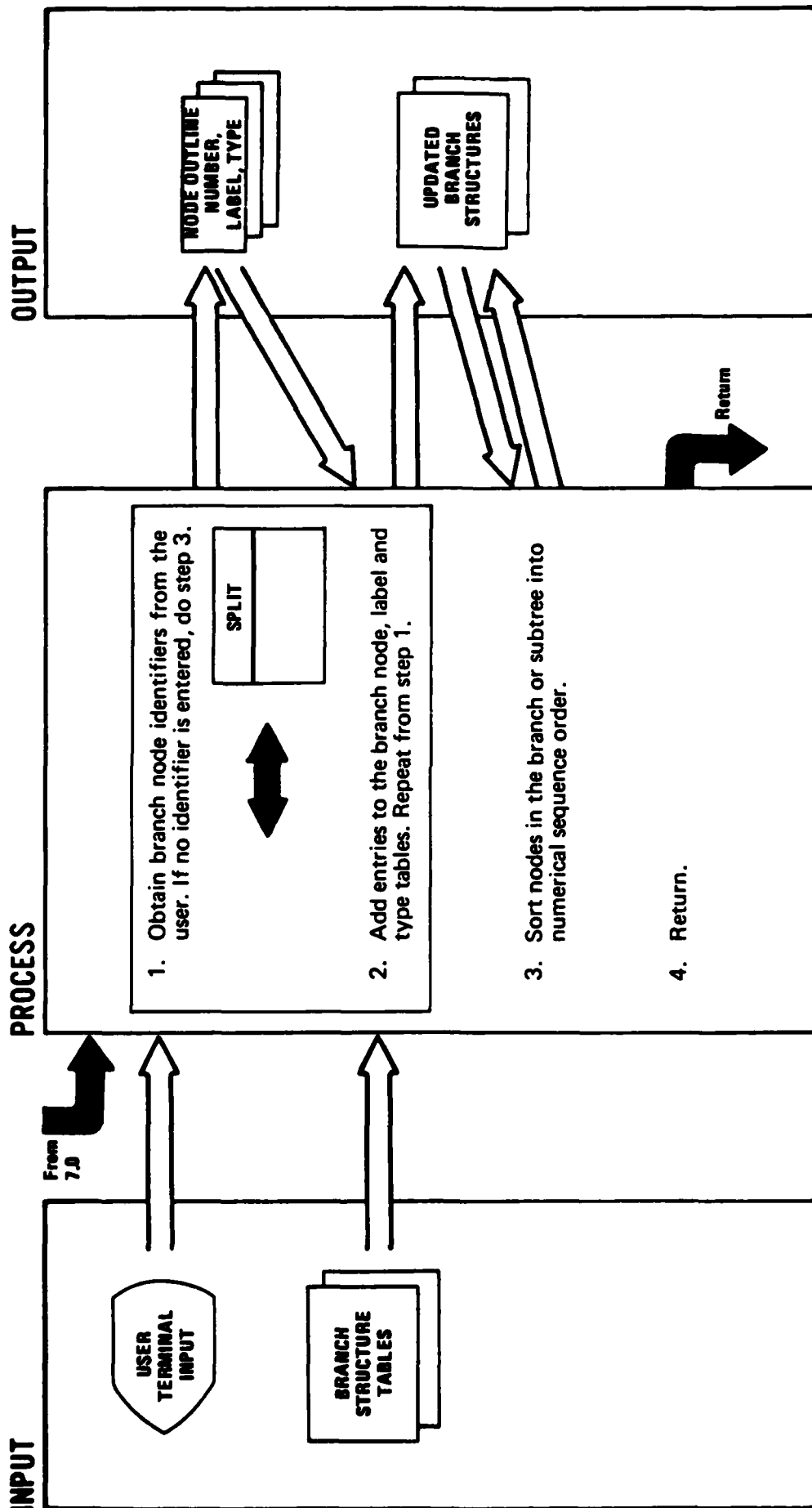
1. Prompt the user for the next branch sequence number; if no number is given, do step 5.
2. Determine whether or not the branch is a new one.
 - a. If the branch is new, add to the branch sequence number and initialize branch label, outline, and type.
 - b. If the branch has already been defined, add to the branch labels, outline, and type.
3. Determine if the branch is to be symmetric and process a. or b.

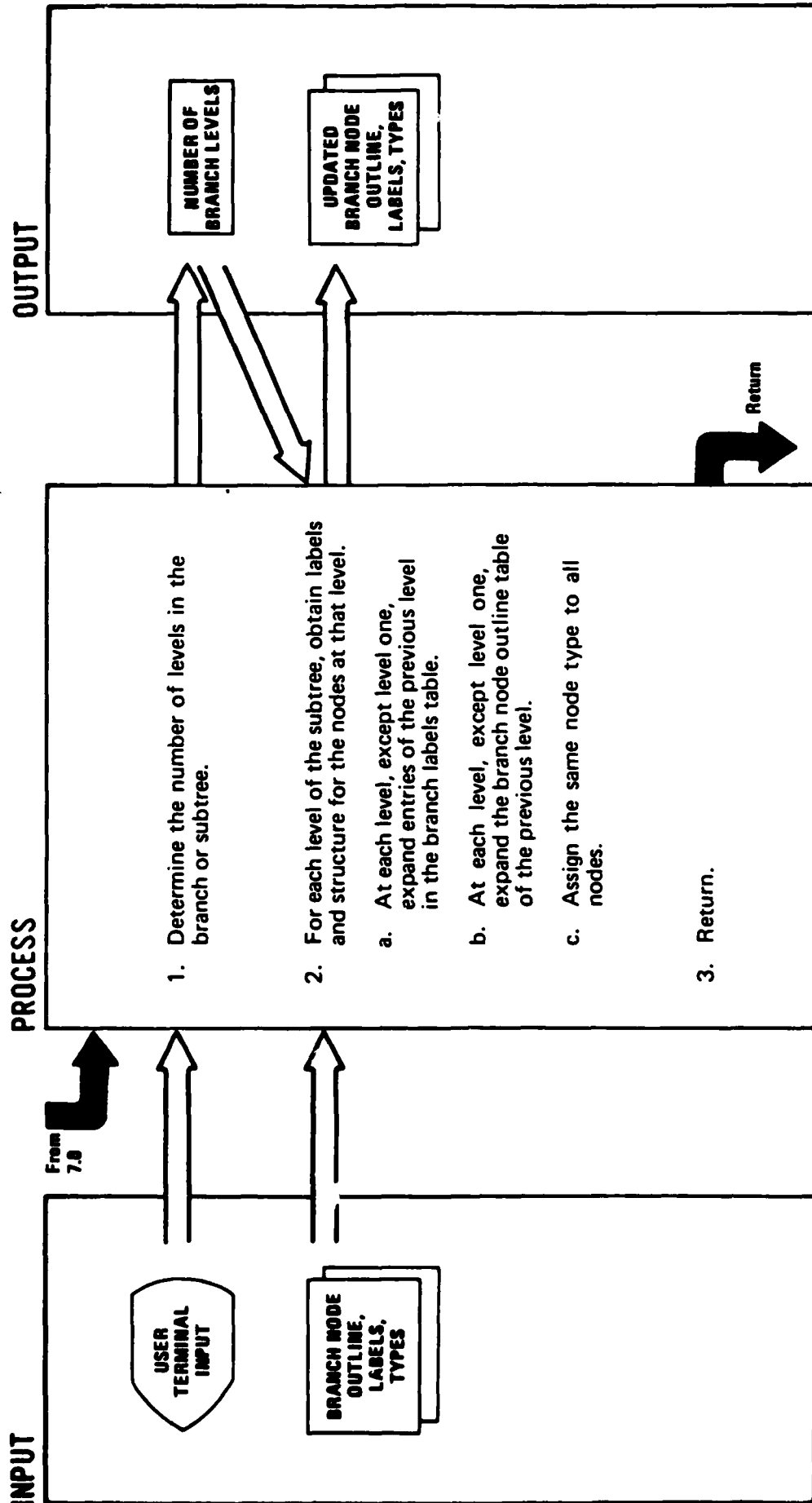
a. Create a non-symmetric branch. 7.1

b. Create a symmetric branch. 7.2
4. Repeat from step 1.
5. Return.

OUTPUT**Extended Description**

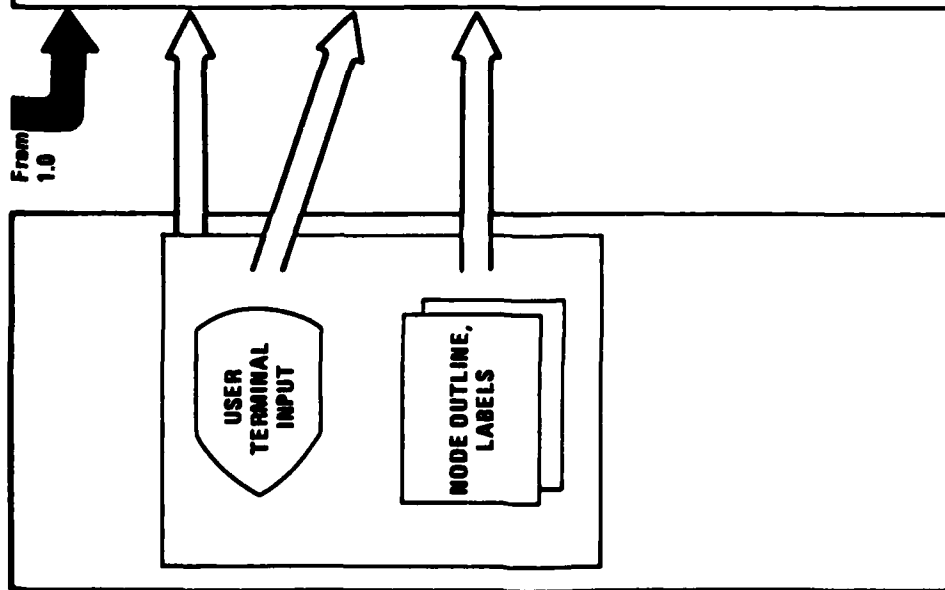
The user is allowed to create separate branch or subtree structures which may be added to the model structure under the "create a structure" process option.



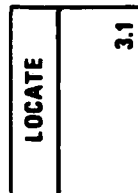
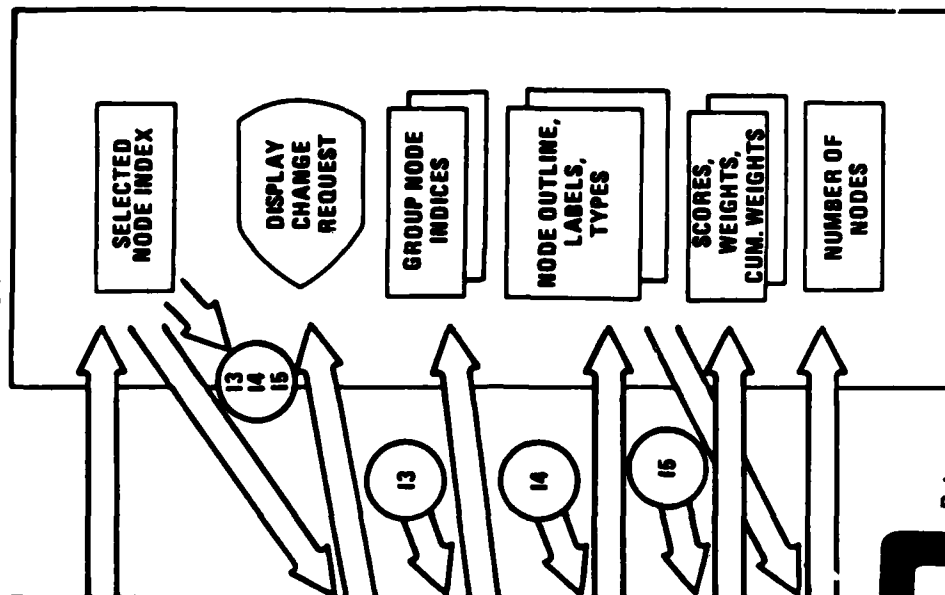


Extended Description

Step 2 processing ensures that for each subsequent level of a multilevel branch structure the outline number, types and labels are all added in the correct numerical sequence to the outline, types and label entries at the previous level. (This is done for every branch node defined at the previous level.)

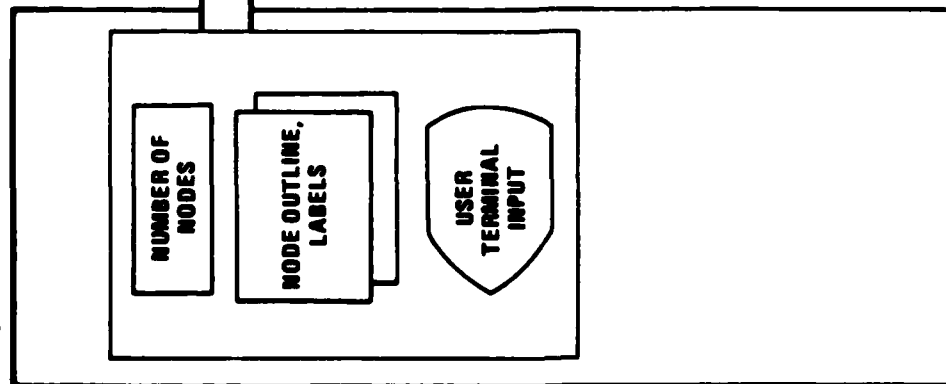
INPUT**PROCESS**

1. Determine which aggregate node is to be deleted. If no response from user, do step 7.
2. Prompt user for a final change of request: If so, repeat from step 1.
3. Determine the indices of all nodes beneath the selected node.
4. Delete all associated elements in the node outline, labels and types arrays.
5. Delete all associated scores, weights and cumulative weights.
6. Reset the number of nodes variable.
7. Return.

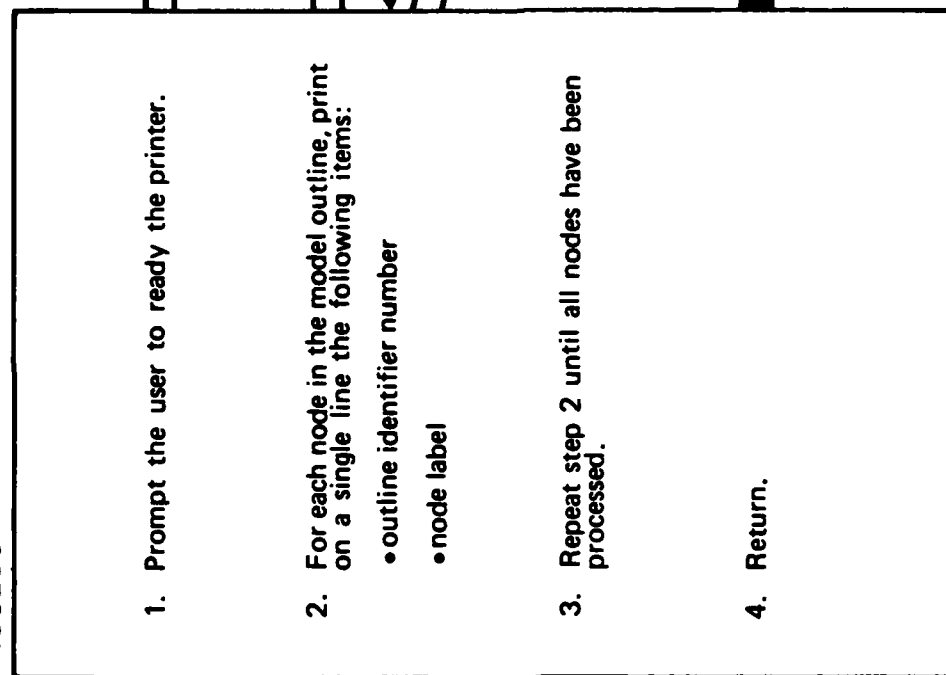
**OUTPUT****Extended Description**

The routine should be executed whenever a group of nodes is to be deleted from an existing node structure. The grouped nodes are all hierarchically placed below a certain aggregate node; hence, a user specification of an aggregate node in step 1 will cause that node and all its subsequent nodes to be deleted.

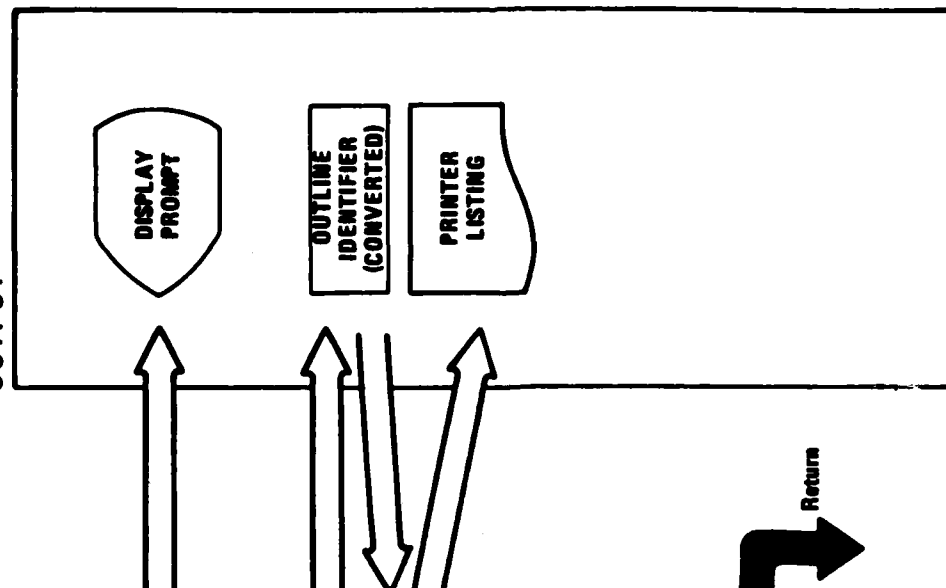
INPUT



PROCESS

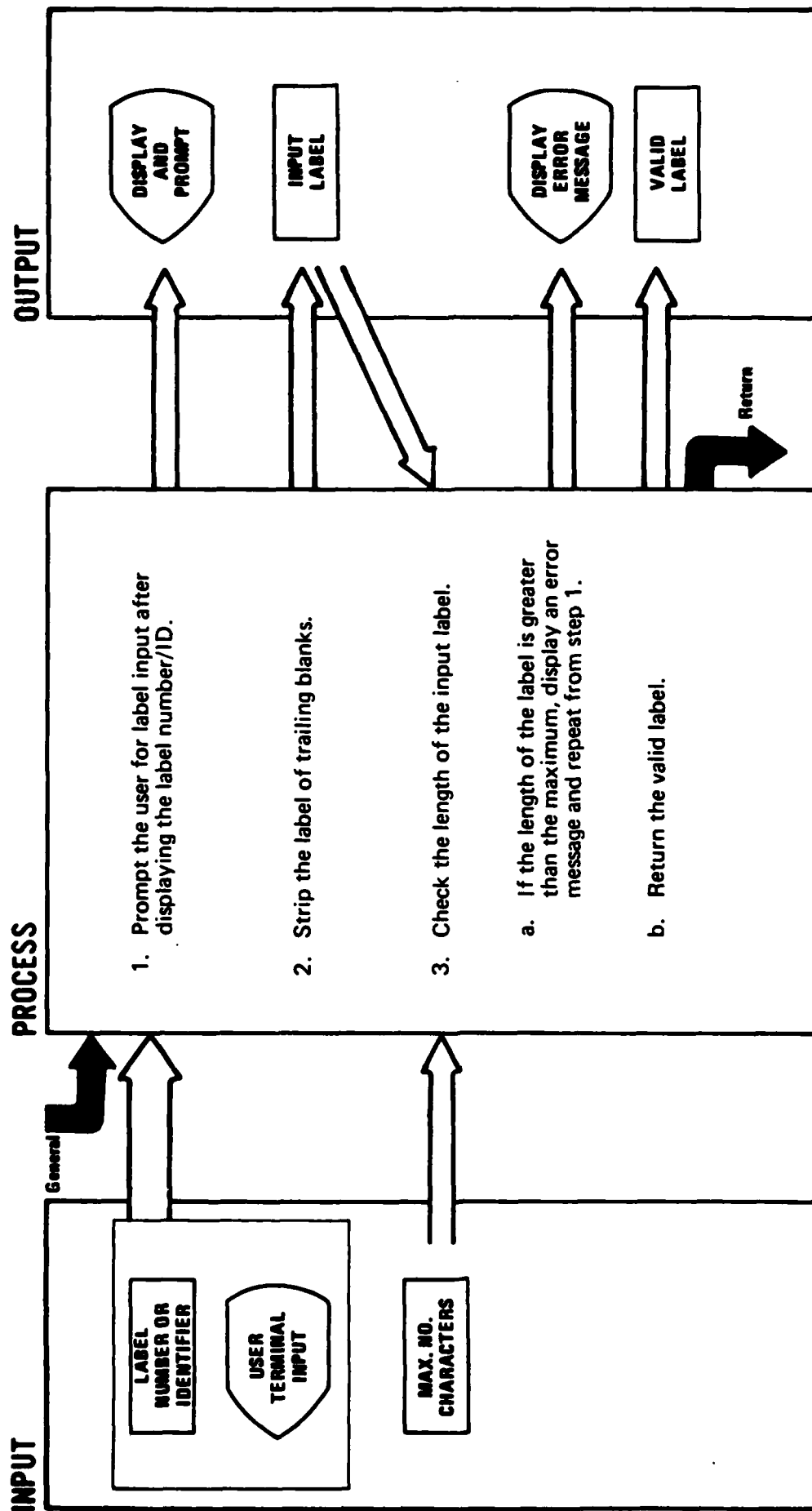


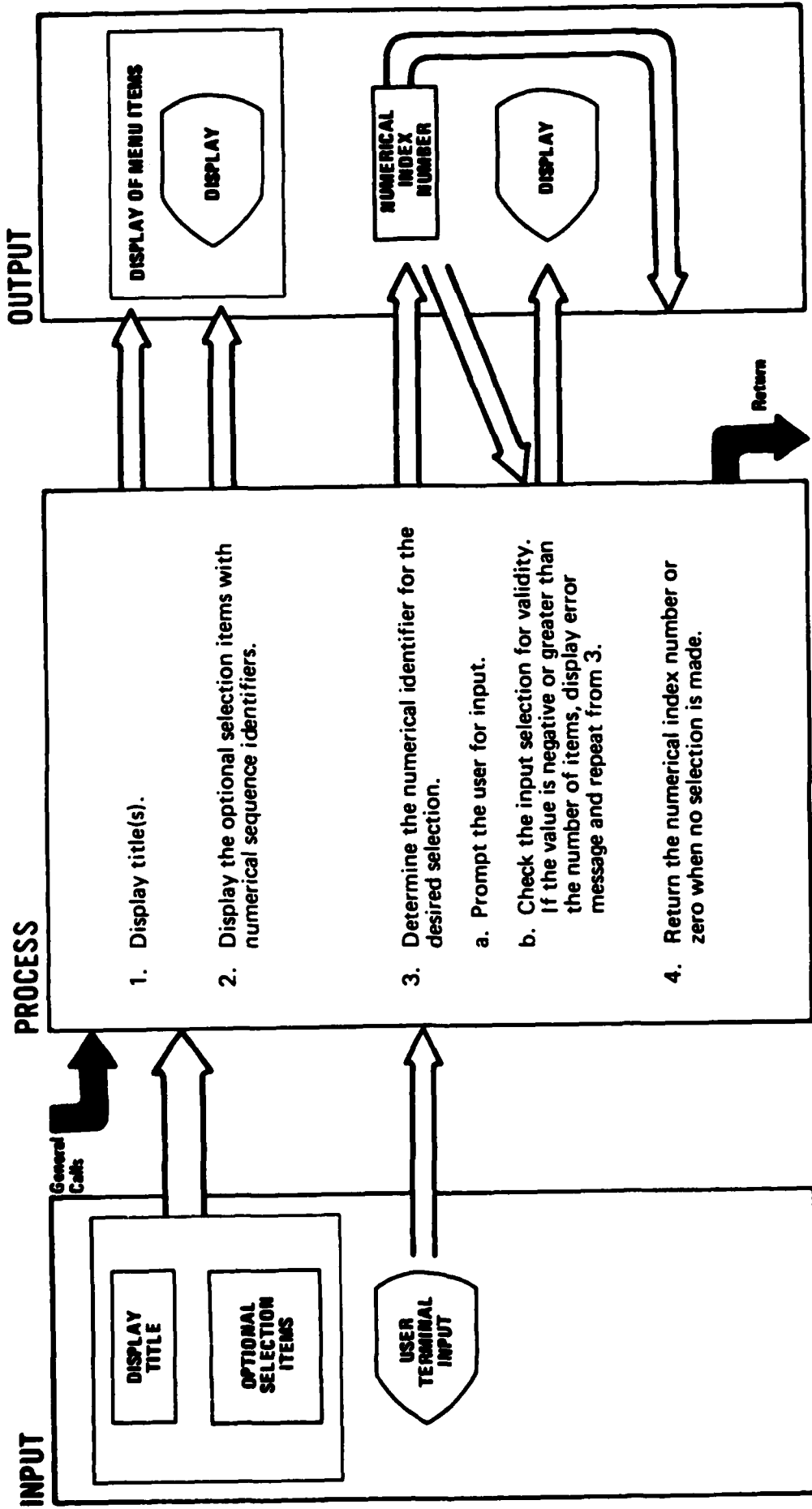
OUTPUT



Extended Description

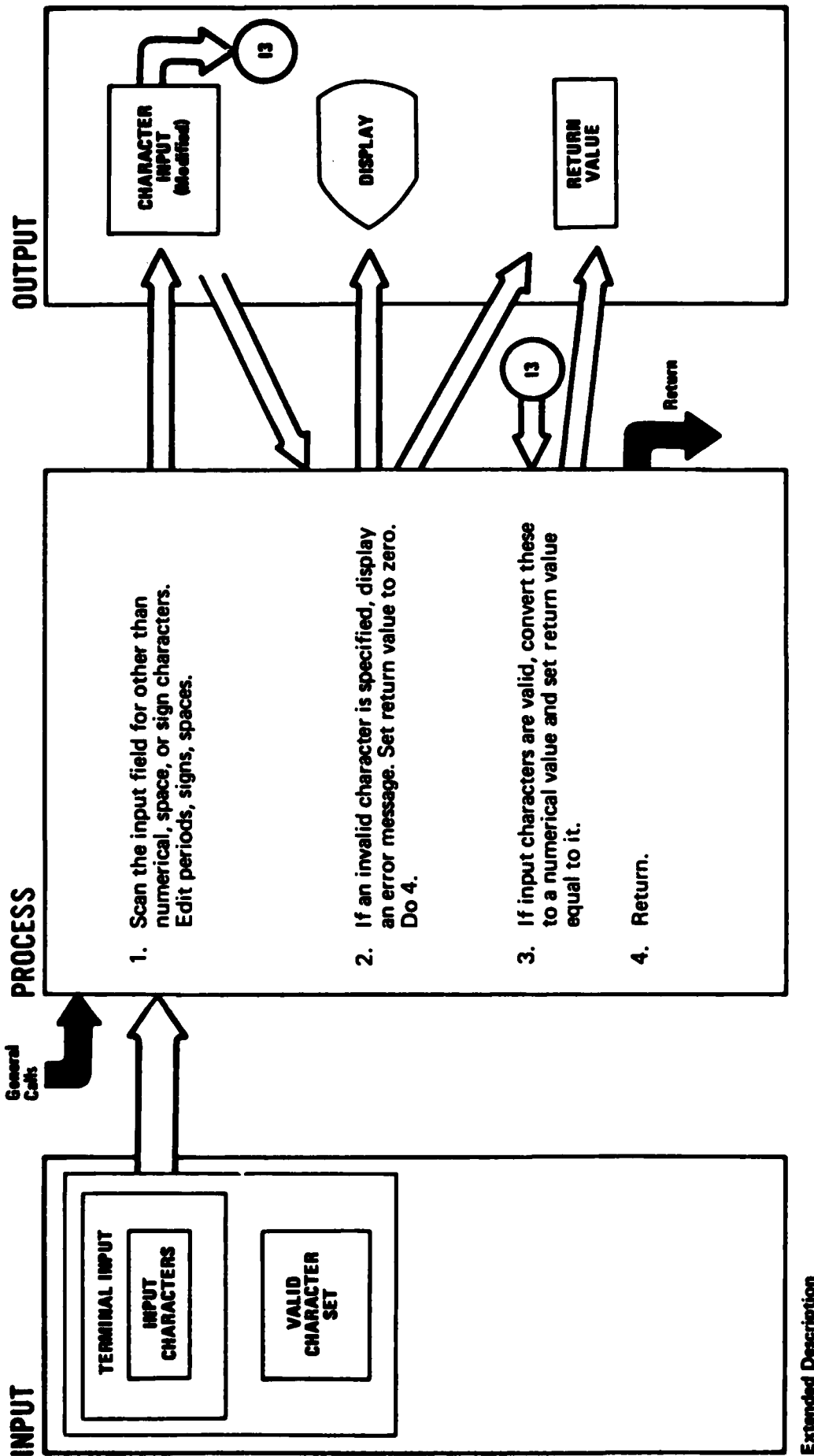
2. The decoded outline identifier number is formatted for output. The output should be equivalent to the user's original input during the creation of the structure.





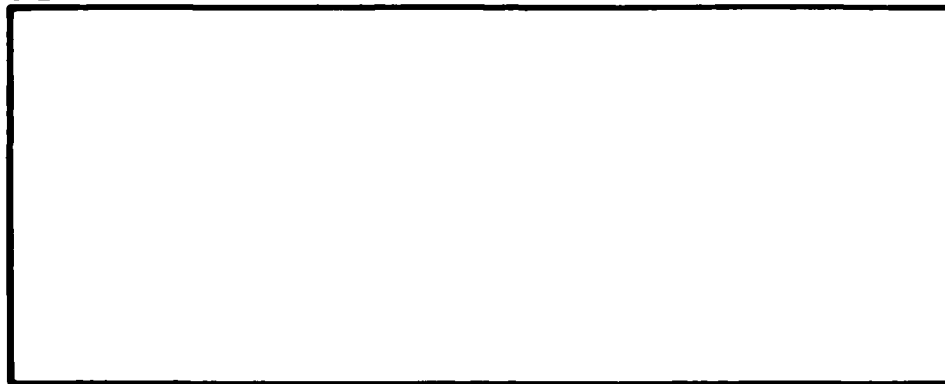
Extended Description

1. The title is passed to this routine so that the display will remain in context with the processing function. For example, a title may be 'DISPLAY RESULTS.'
2. The selections that describe what is optimal are passed as input and are displayed in a list or cookbook MENU format along with item sequence numbers.
3. Prompt the user for the item sequence number of the choice selection. Check the validity of the user input.

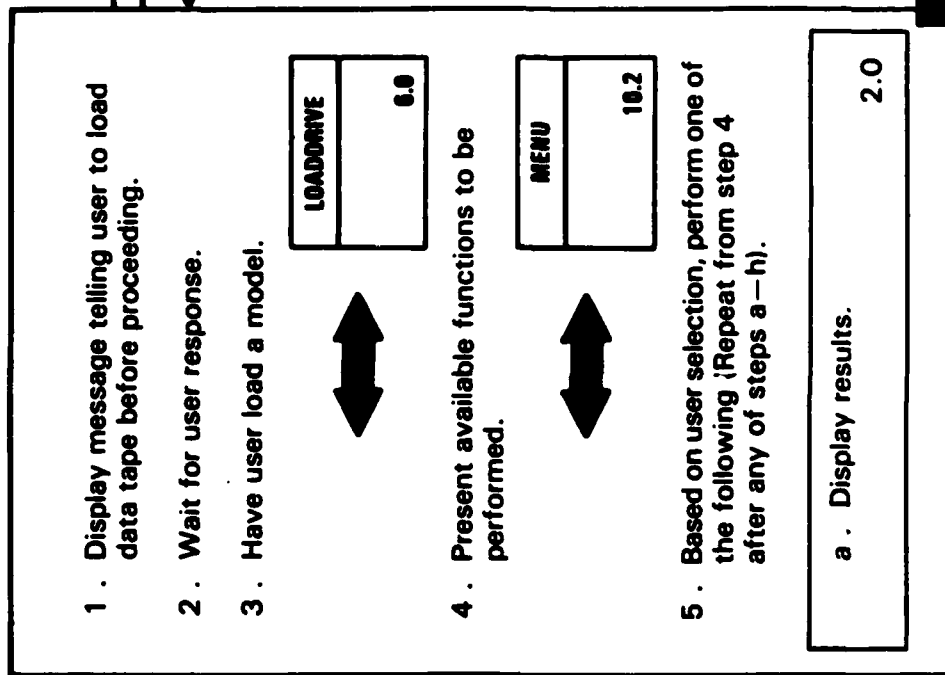


This routine will not be required if system error checking routines interface with the standard keyboard-display input.

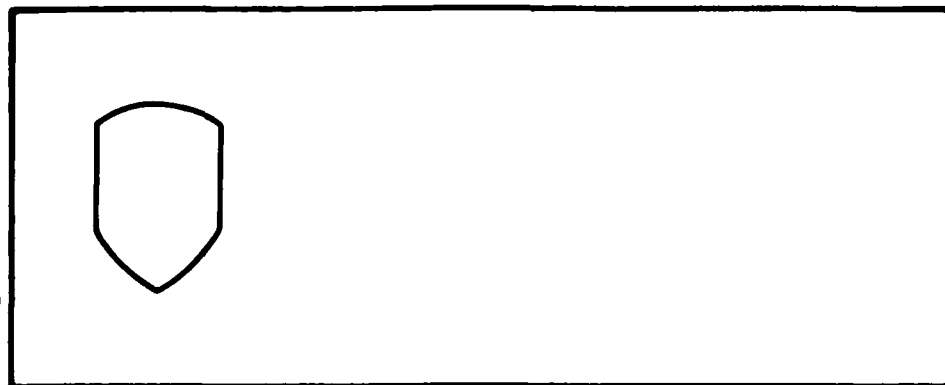
INPUT



PROCESS



OUTPUT



Extended Description

3. The model variables are all loaded into the current work area at this time, or whenever the user wishes to load a new model. Consequently, this documentation assumes that these variables are "global" and always available for reference, input to procedures, or modification.

Pg. 2
5.b.

System/Program: RUN

Name: RUN

Diagram ID: 1.0 Description Entrance to Eval Program

Page: 2 of 2

INPUT

--



PROCESS

b . Sensitivity analysis.	3.0
c . Edit values.	4.0
d . Print results.	5.0
e . Load model.	6.0
f . Save model.	7.0
g . New values.	8.0
h . Print data sheet.	9.0
i . Terminate program.	



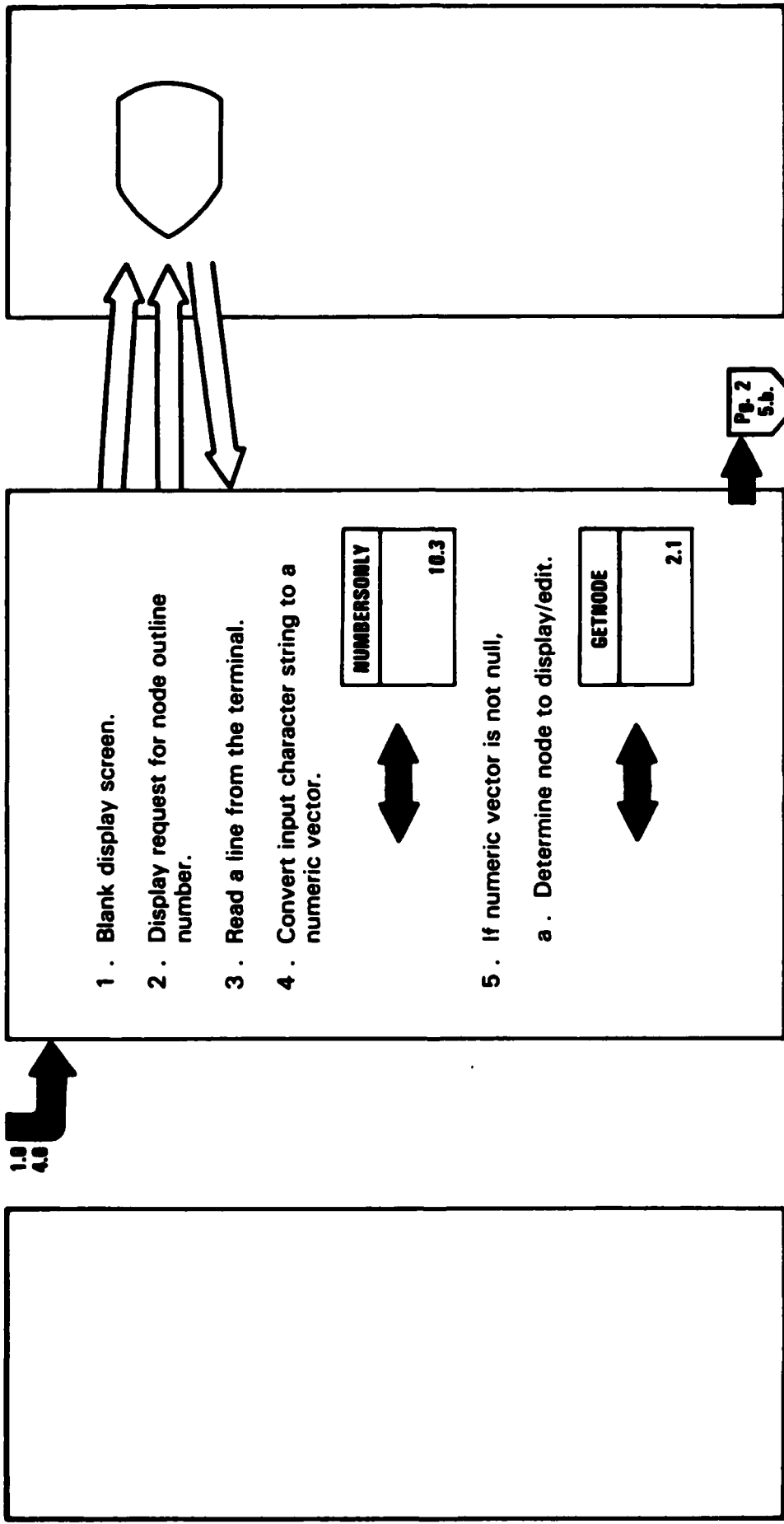
OUTPUT

--

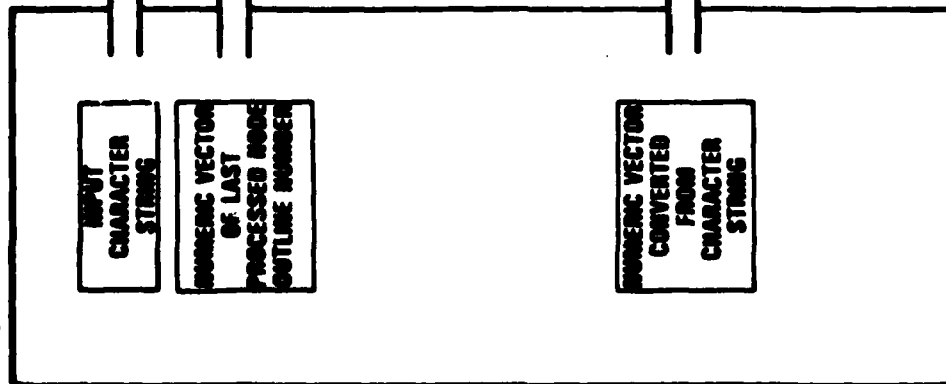
INPUT

PROCESS

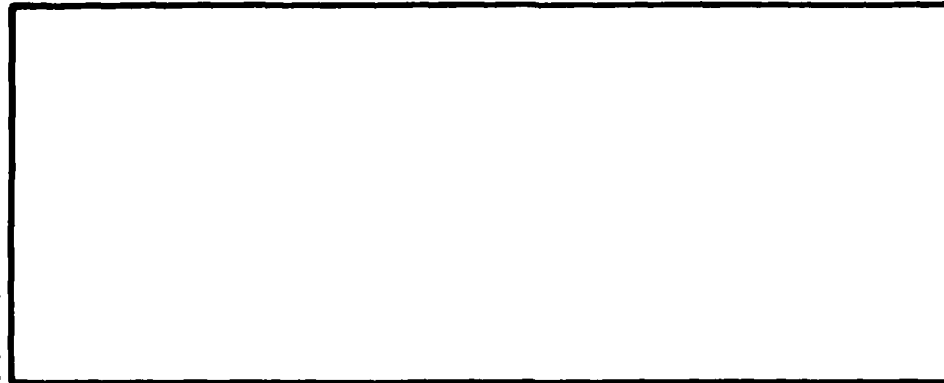
OUTPUT



Pg. 2
5.b.

INPUT**PROCESS**

1. If character string contains a special character indicating to scan up or down a path of the model,
 - a. Set the node outline number equal to the last processed node number.
 - b. If the first element of the numeric vector is non-zero add the value to end of the last processed outline number.
 - c. If the first element of the numeric vector is zero, delete the last element of the last processed outline number.
2. If the character string does not contain a scan character, set the node outline number equal to the numeric vector.
3. Convert new node outline number to same representation as stored in OUTLINE.

OUTPUT**Extended Description**

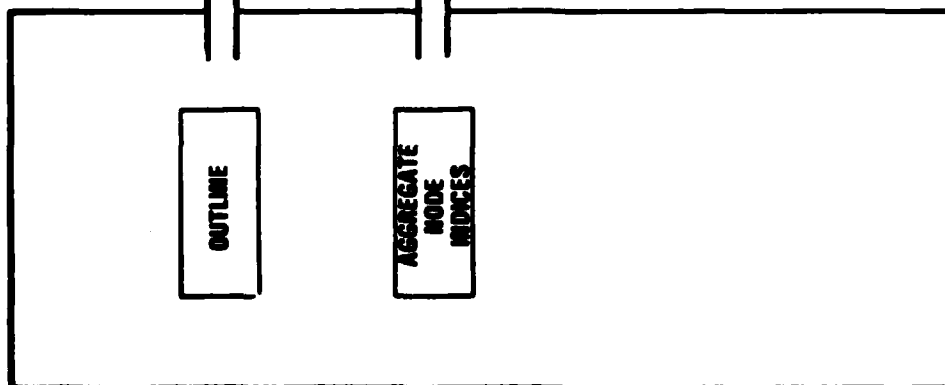
- b. This generates a node outline number one level deeper than the previously processed node. For example, if the previously processed number were 3.2.5 and the input '6' (where the right parenthesis is the scan operator) the new node outline number would be 3.2.5.6.
- c. This generates a node outline number one level higher than the previously processed node. For example, if the previously processed number were 3.2.5 and the input '0' (where the right parenthesis is the scan operator), the new node outline number would be 3.2.

System/Program: RUN Name: GEINODE

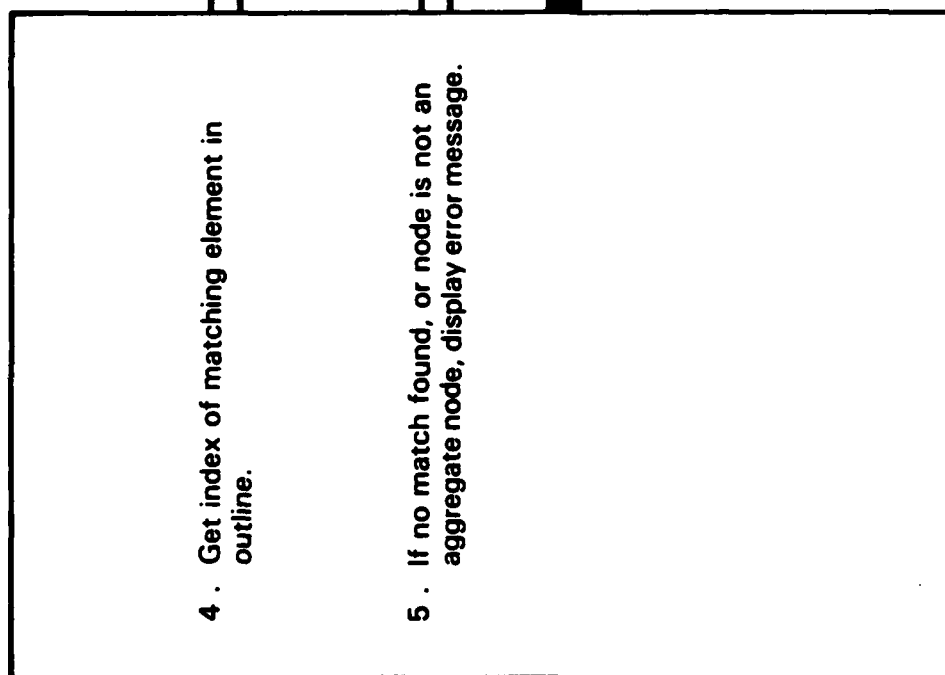
Diagram ID: 2.1 Description: Determine Aggregate Node to Display

Page: 2 of 2

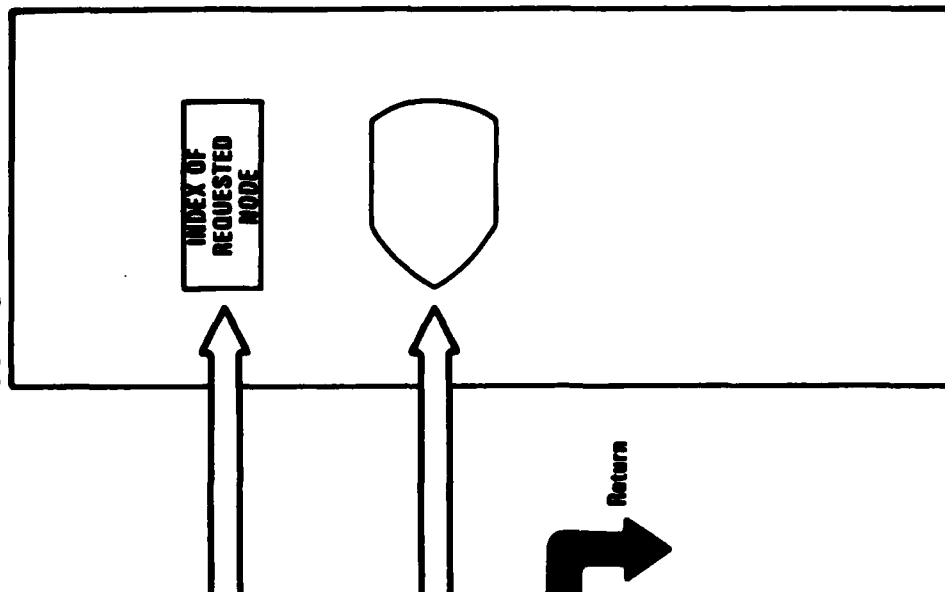
INPUT



PROCESS



OUTPUT



INPUT

PROCESS

1. Get labels of nodes of up to 3 levels up from requested node.

TRACE
2.2.1



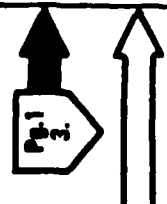
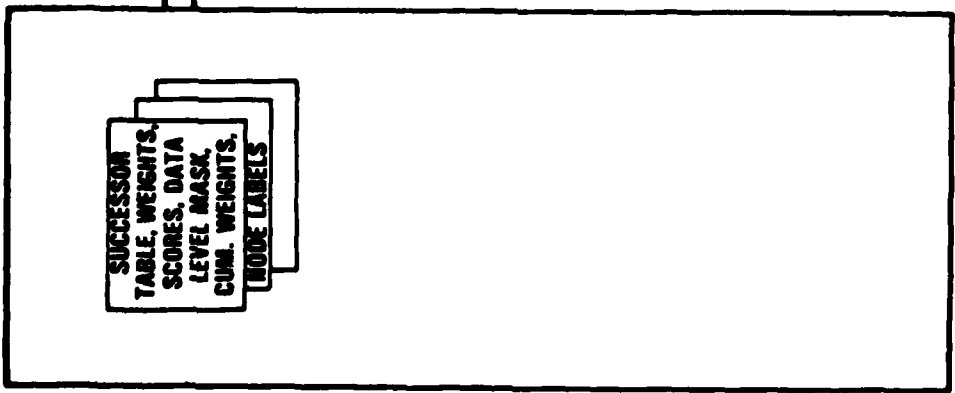
2. Display heading consisting of requested node number and label, up to 3 additional levels of labels and node type of requested node.

3. Display subheading indicating the meaning of the values specified in each column, including the system labels.

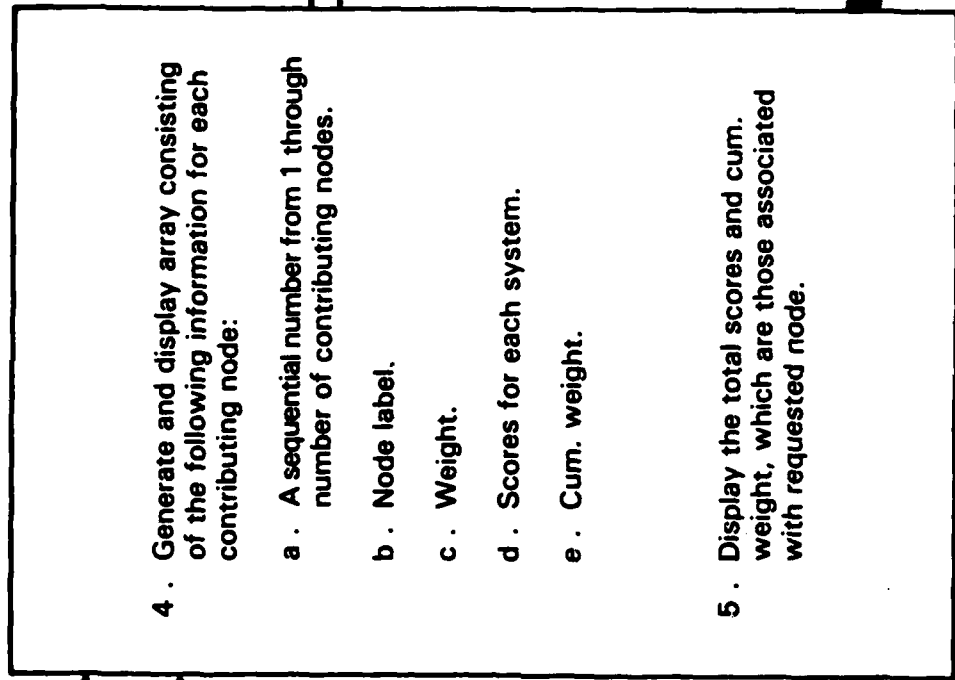
OUTPUT

Pg. 2
4.

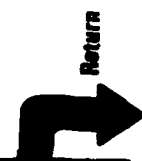
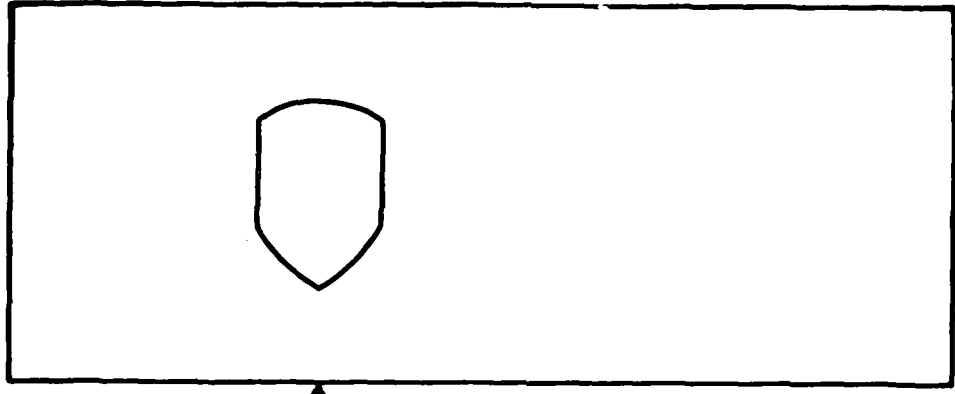
INPUT



PROCESS



OUTPUT



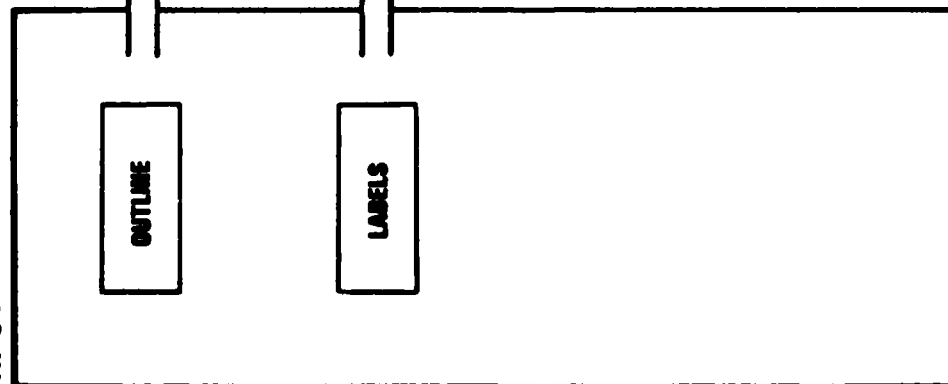
System/Program: RUN

Name: TRACE

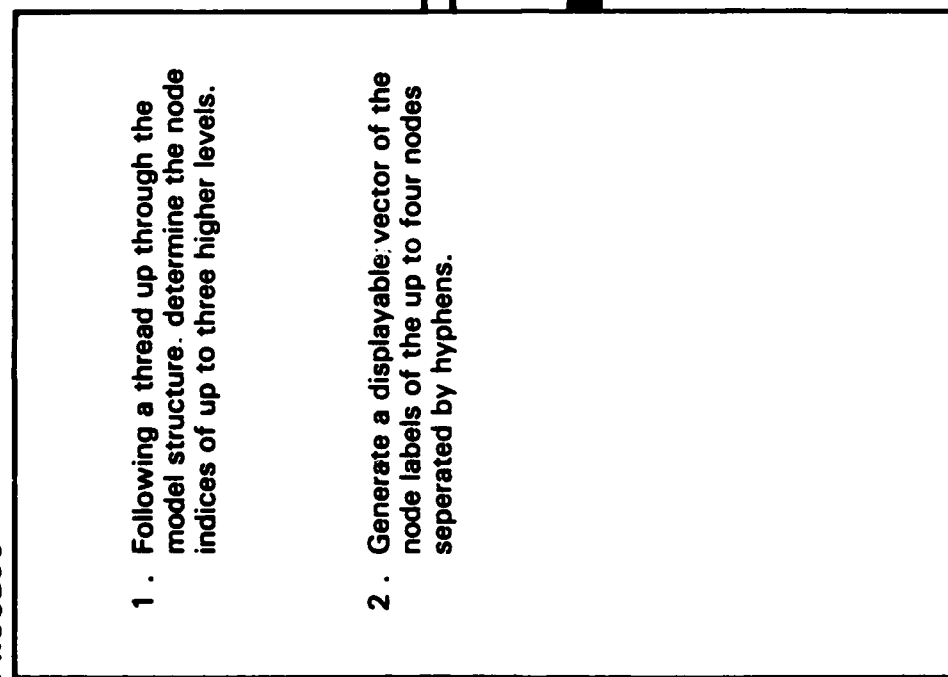
Diagram ID: 2.2.1 Description: Get Nodes of Thread Up Tree

Page: of

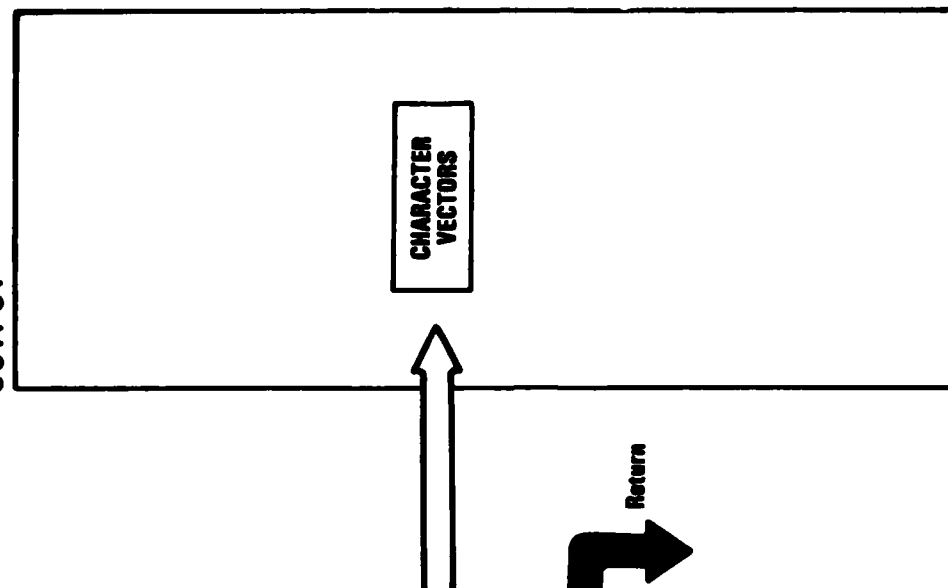
INPUT



PROCESS



OUTPUT

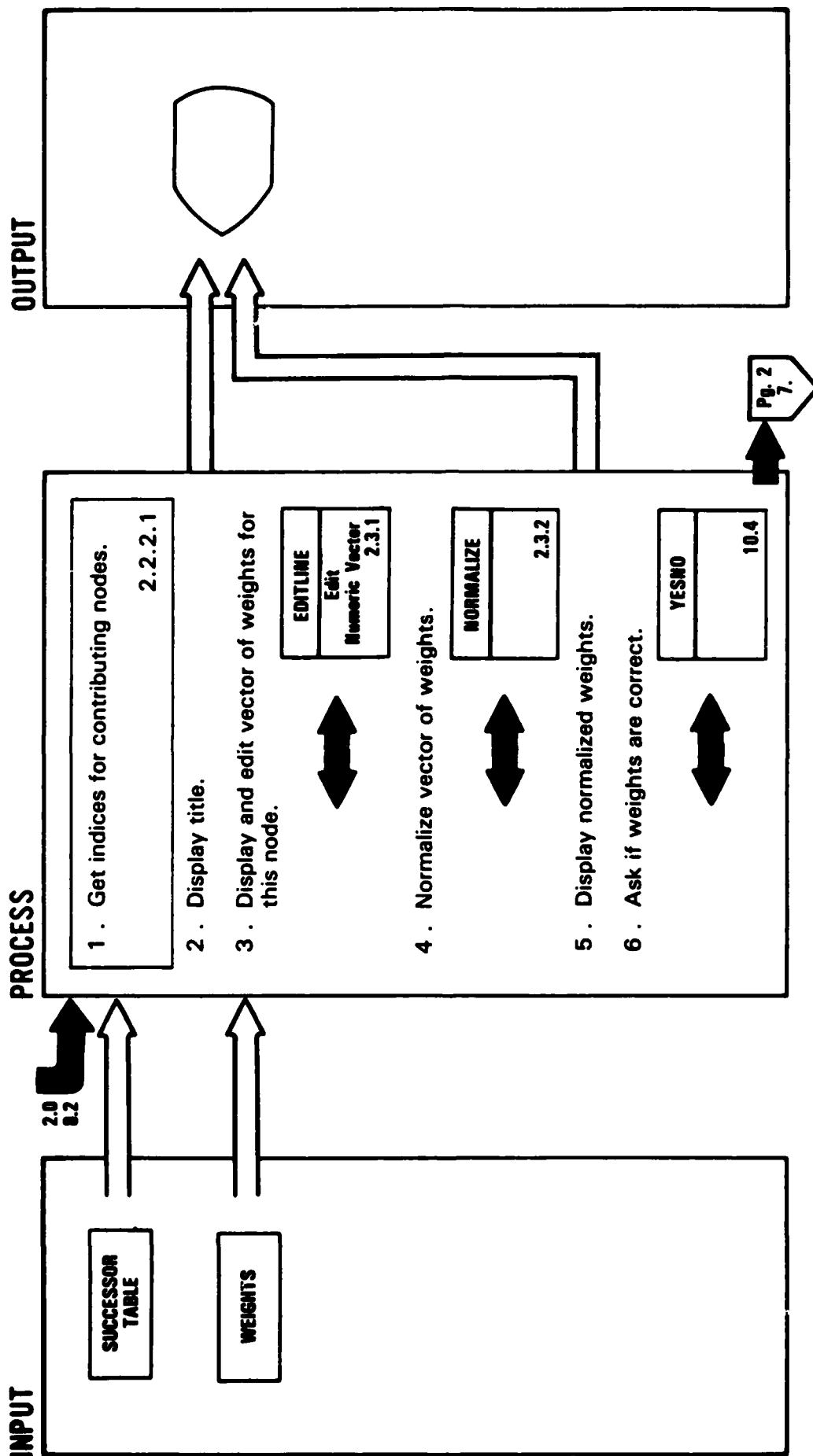


2.2

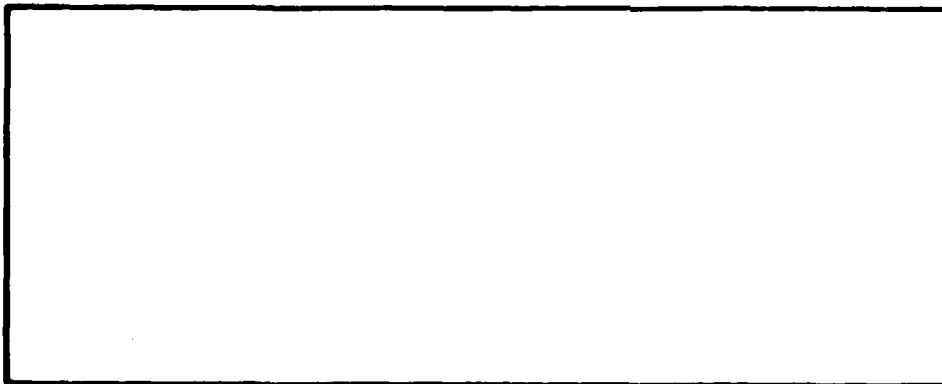
Return

Extended Description

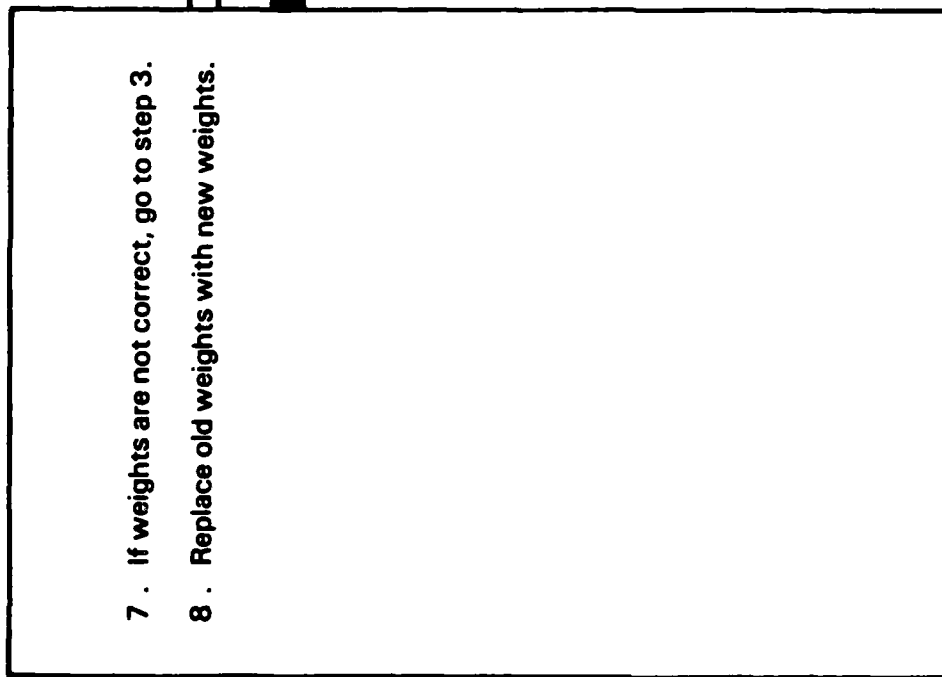
For instance, if the requested node number is 1.4.2.6, the next higher level would be 1.4.2, and the fourth (or highest) calculated level would be 1.4.



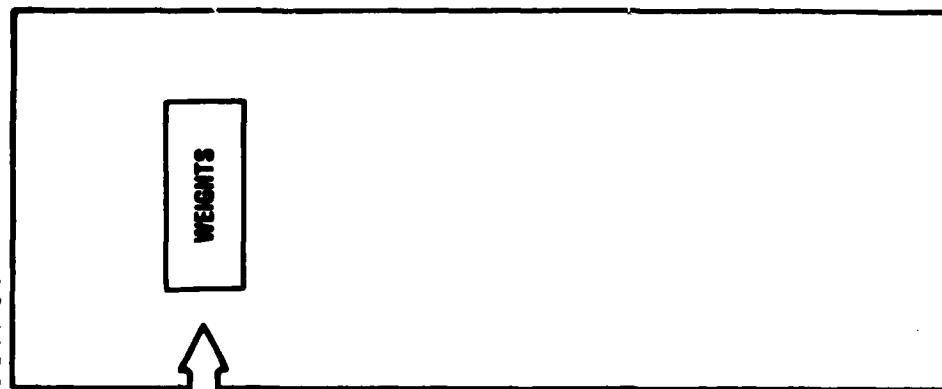
INPUT



PROCESS



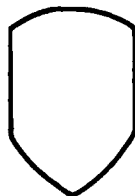
OUTPUT



INPUTNUMERIC
VECTORALPHANUMERIC
LABEL2.3
4.1**PROCESS**

1. Count the number of elements in the input vector.
2. Format the vector, add backspaces for cursor position, attach to label, and display.
3. Read a line from the display.
4. Delete label, convert string into numbers.
5. If the number of elements is less than the original vector, left justify the vector and fill with zeros.
6. If the number of elements is greater than the original vector, drop the extra elements.

NUMBERS ONLY
10.3

**OUTPUT**NUMERIC
VECTOR

Return

System/Program: RUN

Name: NORMALIZE

Diagram ID: 2.3.2 Description: Normalize a Vector of Numbers

Page: of

INPUT

NUMERIC
VECTOR

PROCESS

1. Divide each element of the vector by the sum of the elements of the vector, and multiply each element by 100.
2. Repeat step 1 once for all vector elements.

OUTPUT

NUMERIC
VECTOR

Extended Description

1. Performing this operation converts a group of arbitrary values to a group of values that add up to 100. The values all maintain the same relativity.
2. Performing this operation twice allows the case where the original values are all zero. The final result is a group of equal numbers that add up to 100.

INPUT

NODE
TYPES

1.0

PROCESS

1. Blank display screen.
2. If all nodes in structure are not type W (weighted additive) display error message and return.
3. If all nodes are type W,

a. Elicit index of desired node.

LOCATE	
	10.1

b. If index is not zero,

- 1) Elicit ranges for computing sensitivity and display title.

GETRANGE	
	3.1

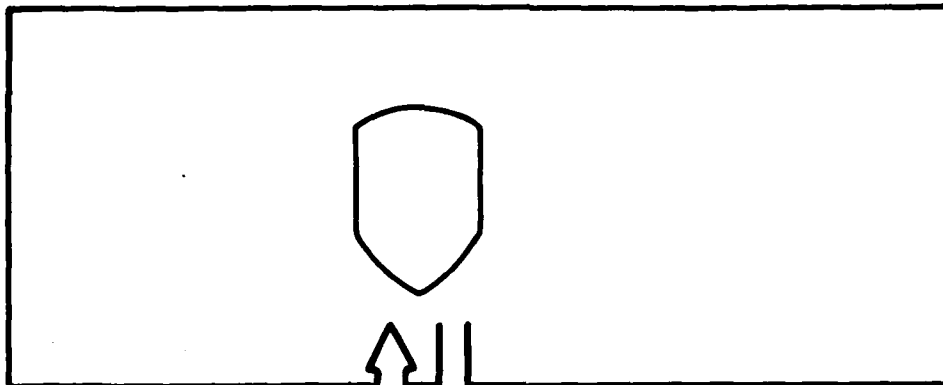
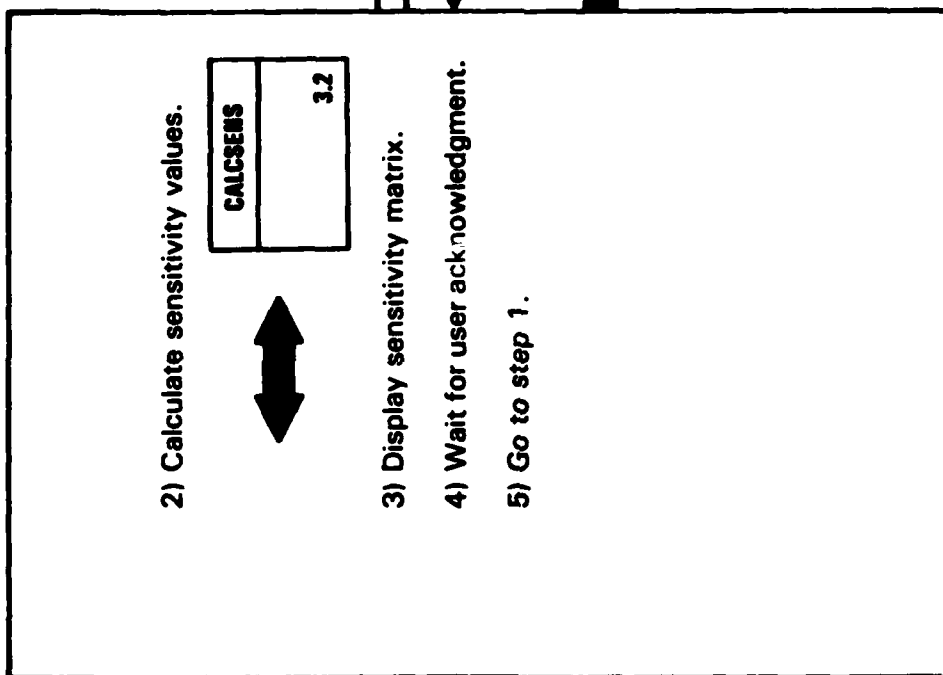
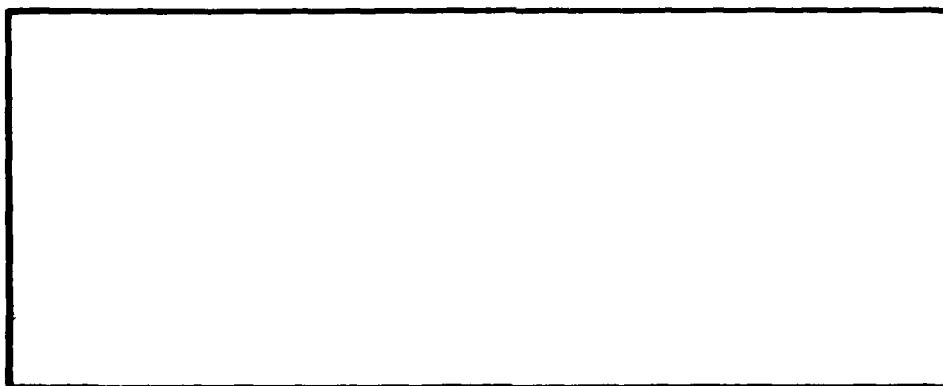
Pg. 2
3.2

OUTPUT

INPUT

PROCESS

OUTPUT



System/Program: RUN

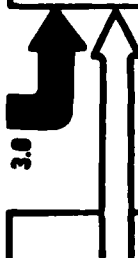
Name: GETRANGE

Diagram ID: 3.1 Description: Elicit Ranges for Computing Sensitivity Analysis

Page: of

INPUT

OUTLINE,
NODE LABELS,
CUMUL. WEIGHTS



PROCESS

1 . Display title consisting of node outline number, node label, and current cum. weight.

2 . Get minimum cum. weight to consider.

ENTERLINE	3.1.1
-----------	-------



3 . Get maximum cum. weight to consider.

ENTERLINE	3.1.1
-----------	-------

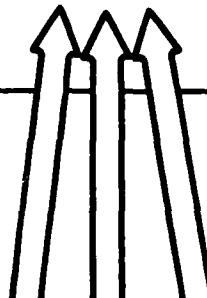
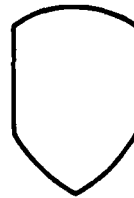
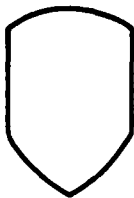


4 . Blank display screen.

5 . Display title consisting of node outline number, node label, and current cum. weight.

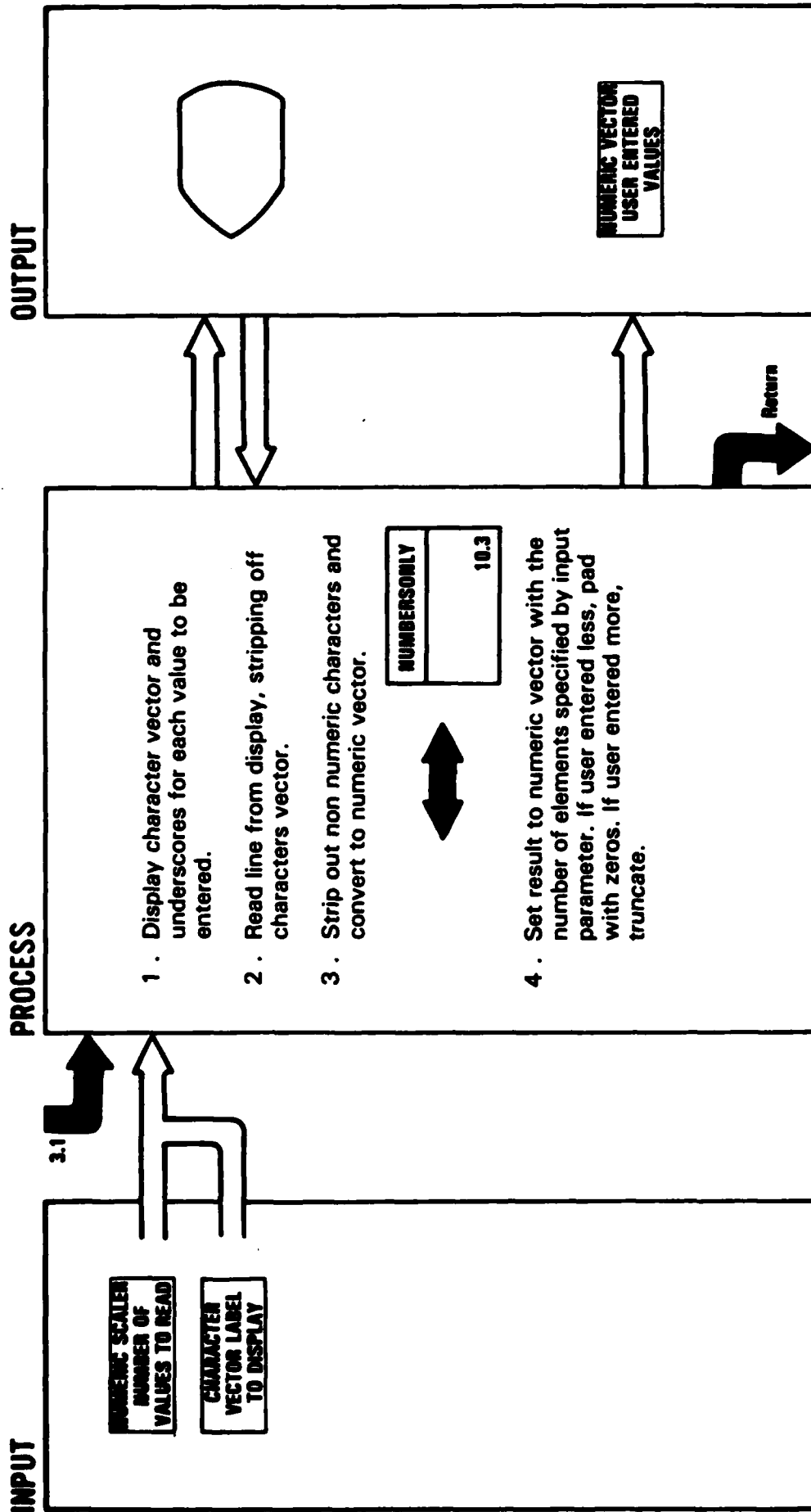
6 . Display column title consisting of 'weight' and system labels.

OUTPUT

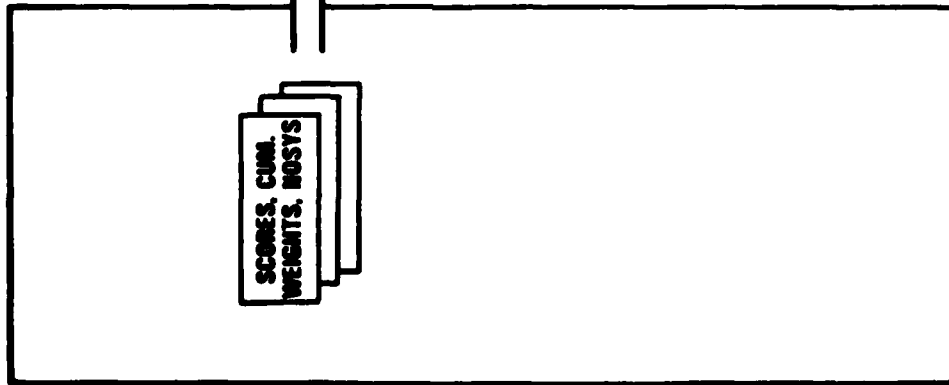


Return

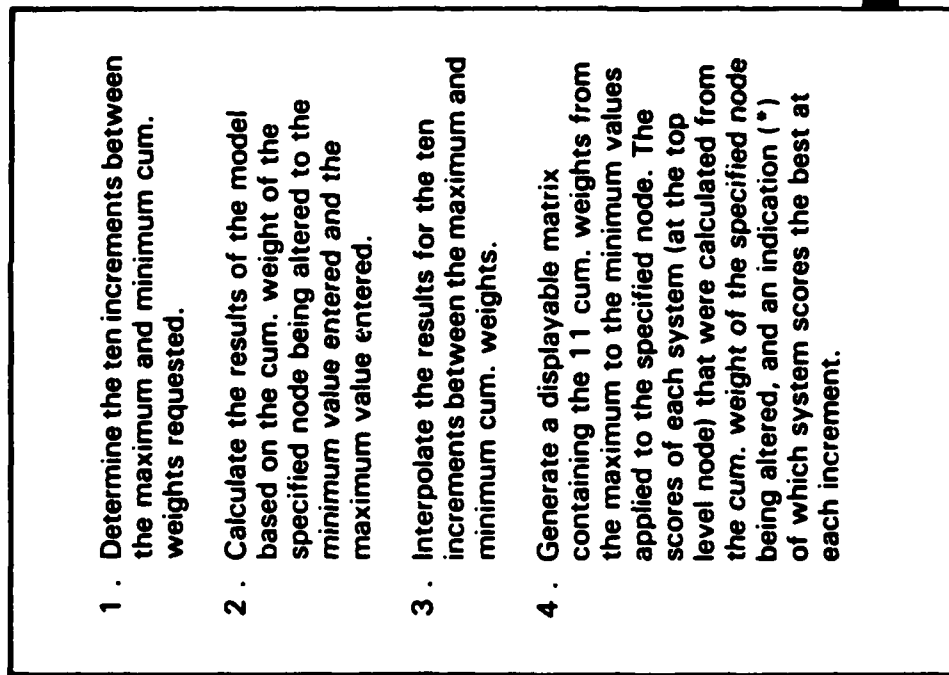




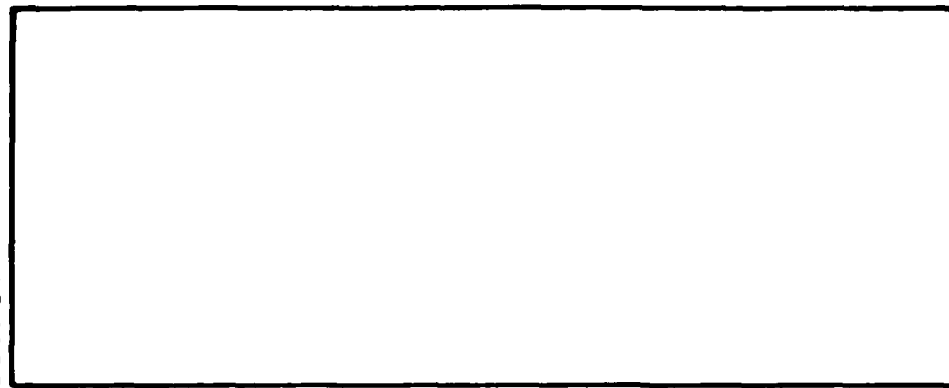
INPUT



PROCESS



OUTPUT



Extended Description

2. When the cum. weight of the specified node is altered, the total of the cum. weights of all other nodes automatically goes up or down such that the sum remains the same. The model variable containing the cum. weights does not have to be changed from its original set of values.

Return

System/Program: RUN Name: EDITOR

Diagram ID: 4.0 Description: Edit Values

Page: of

INPUT

--

PROCESS

1. Elicit number of desired option from user.

MENU	10.2
------	------

2. If option selected,

a. If edit weights selected,

SELECT	2.0
--------	-----

b. If edit scores selected,

EDITSC	4.1
--------	-----

c. Go to step 1.

3. If no option selected, calculate model.

ROLLBACK	4.2
----------	-----

Return

OUTPUT

--

System/Program: RUN

Name: EDITSC

Diagram ID: 4.1

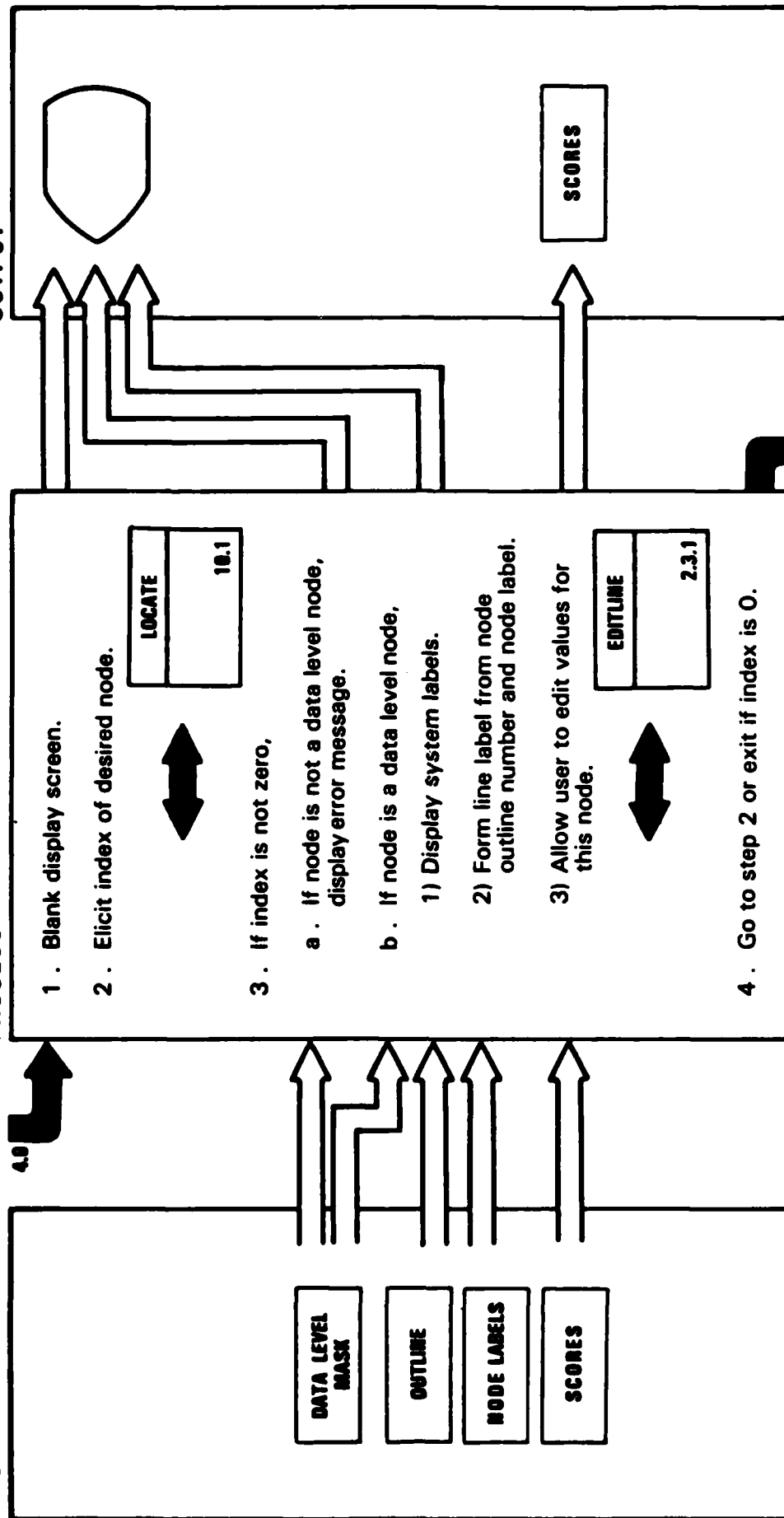
Description: Edit Data Level Node Scores

Page: of

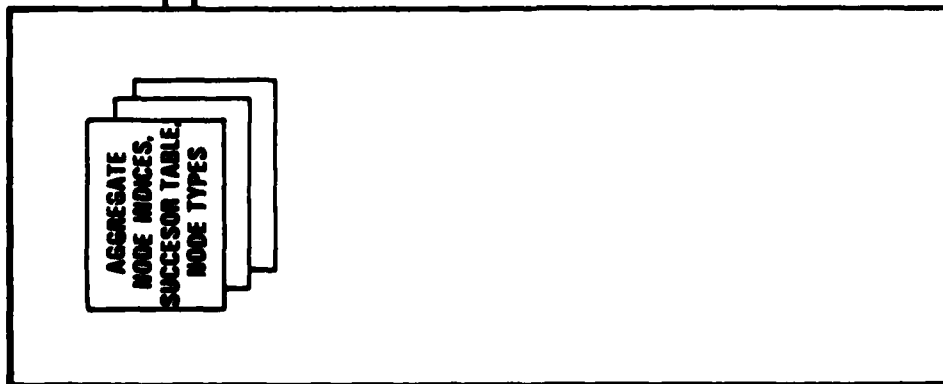
INPUT

PROCESS

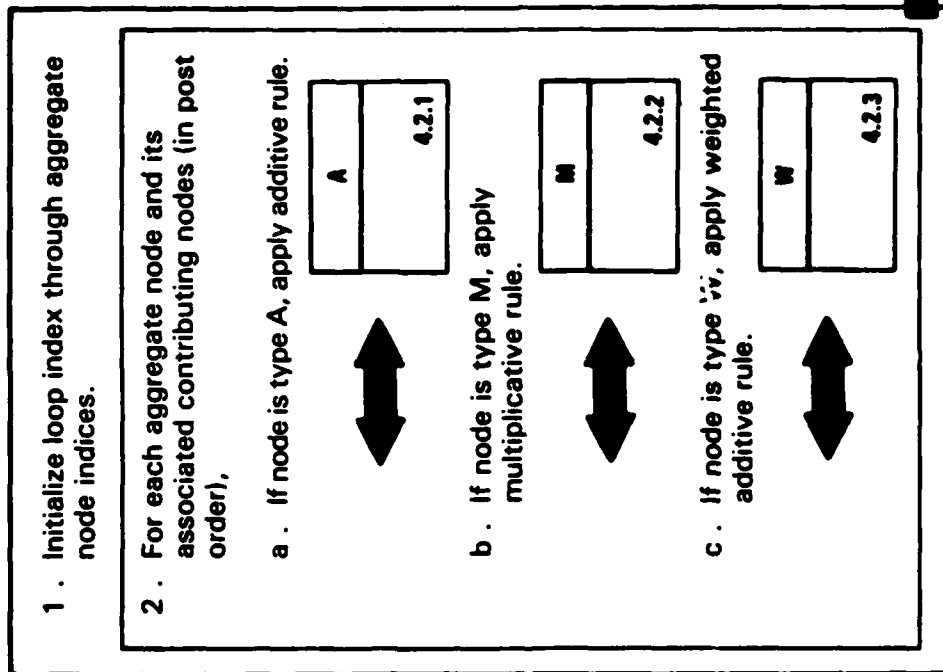
OUTPUT



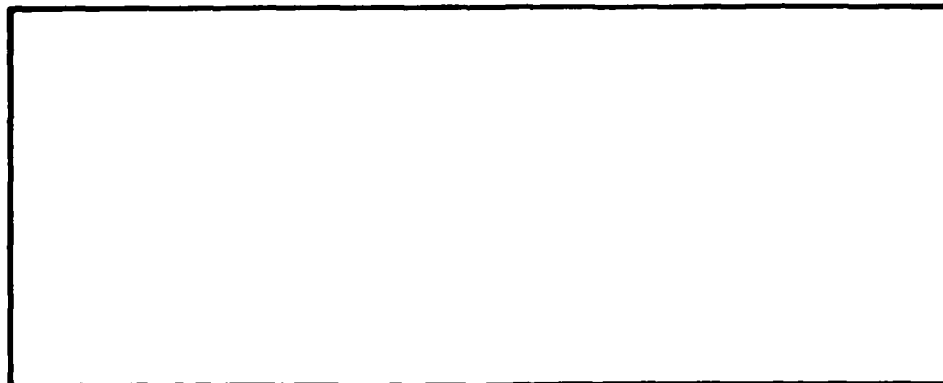
INPUT



PROCESS



OUTPUT



System/Program: RUN Name: ROLLBACK

Diagram ID: 4.2 Description: Calculate Model

Page: 2 of 2

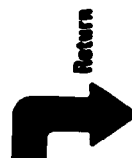
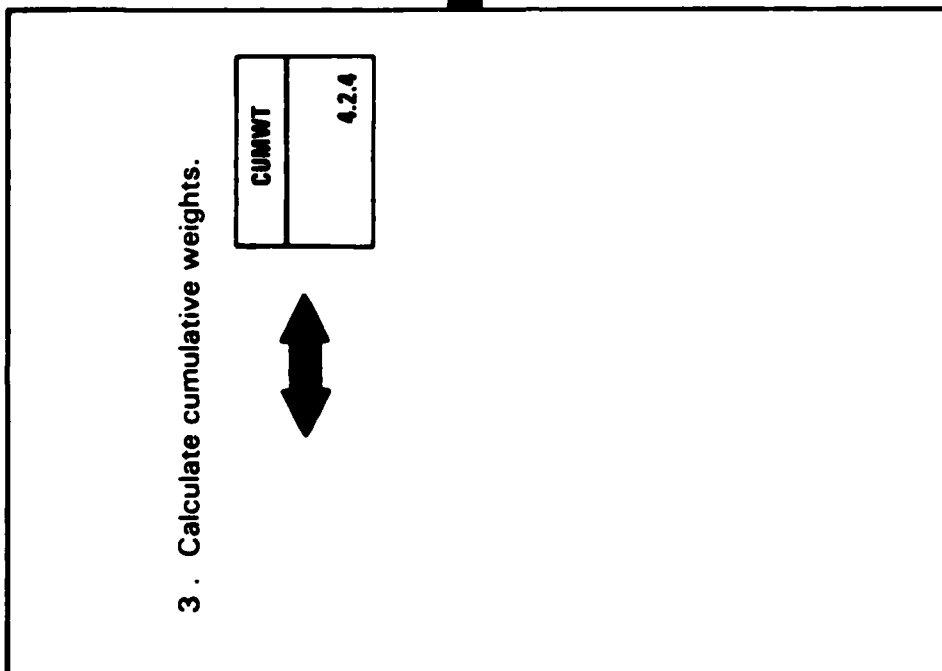
INPUT

--



Pg. 1
2.c.

PROCESS



Return

OUTPUT

--

System/Program: RUN

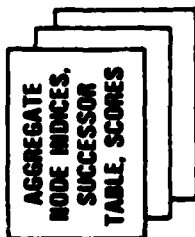
Name: A

Diagram ID: 4.2.1

Description: Calculate Type A

Page: of

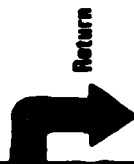
INPUT

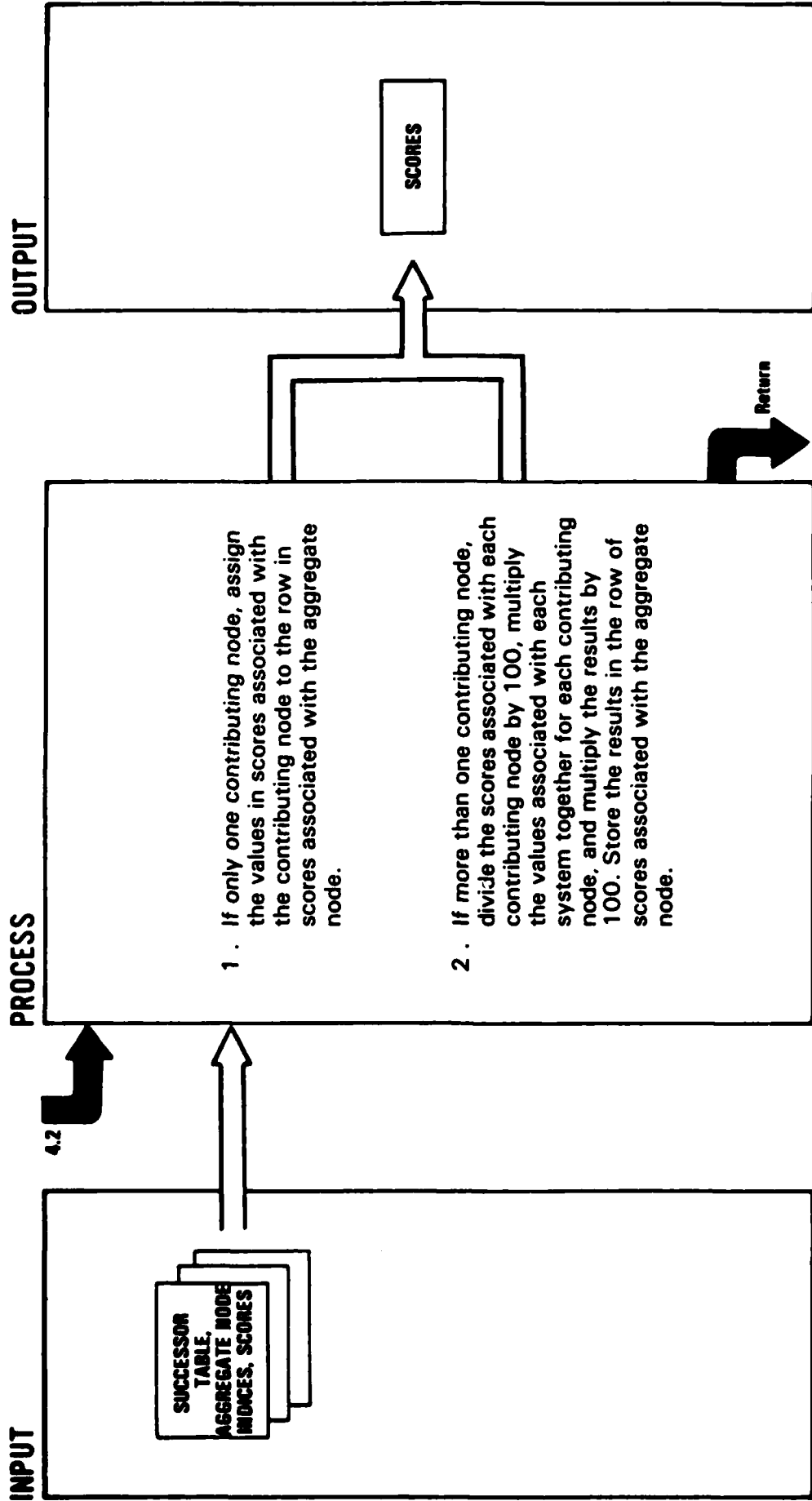


PROCESS

1. Set each value in scores associated with the aggregate branch to the total of the associated values for the contributing nodes.

OUTPUT





INPUT

SUCCESSOR
TABLE,
AGGREGATE NODE
INDICES, SCORES,
WEIGHTS

4.2

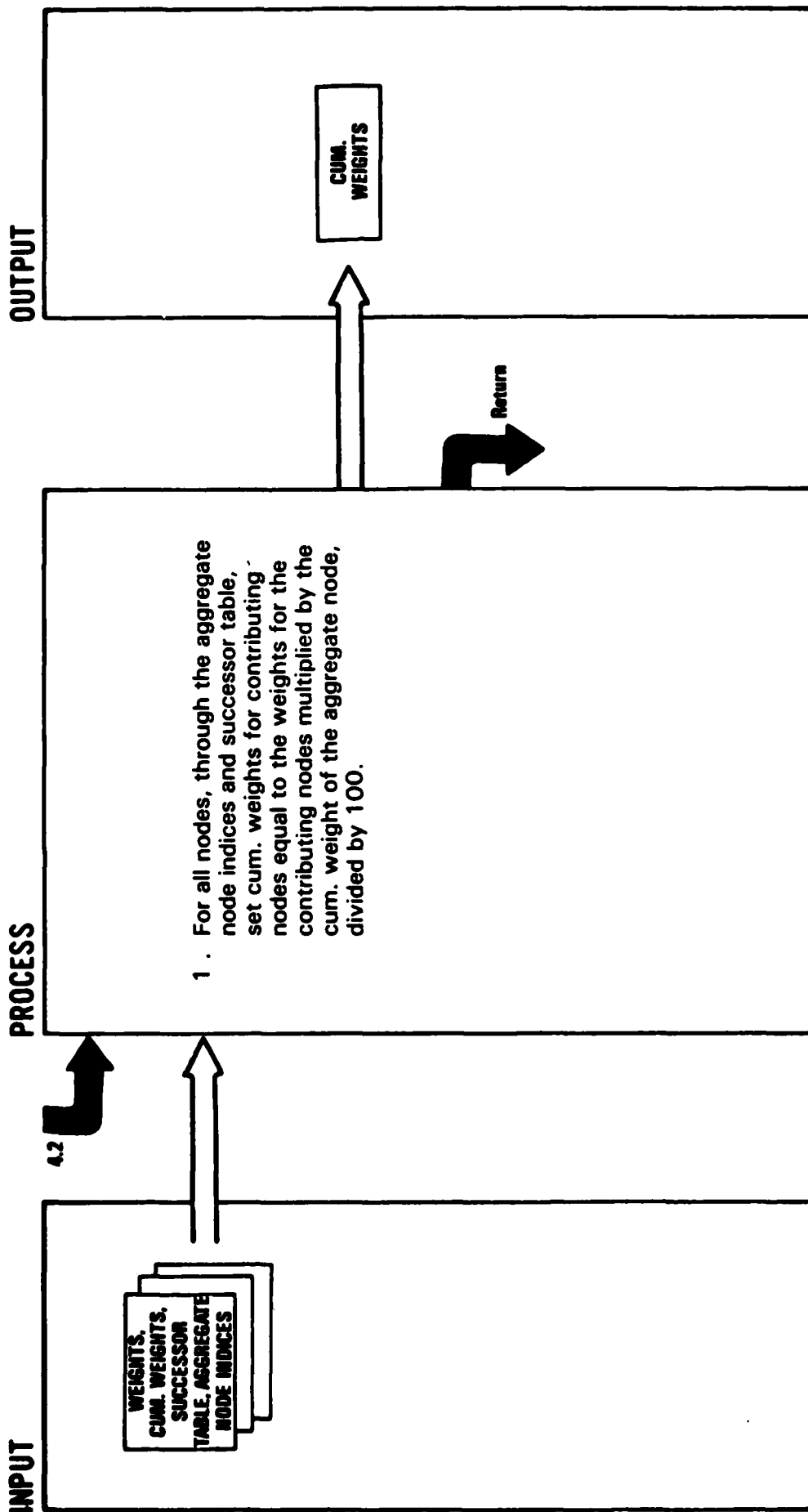
PROCESS

1. Divide the weights associated with the contributing nodes by 100, multiply the scores associated with the contributing node by the calculated weight, add together the results for each system within the contributing nodes, and save the results in the scores for the aggregate node.

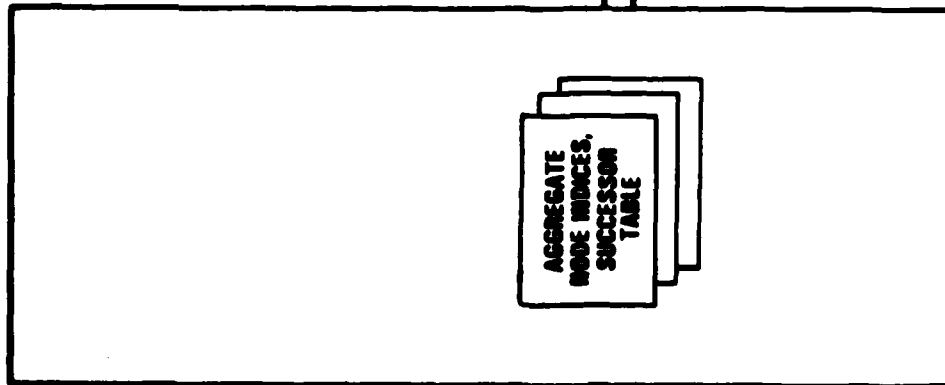
Return

OUTPUT

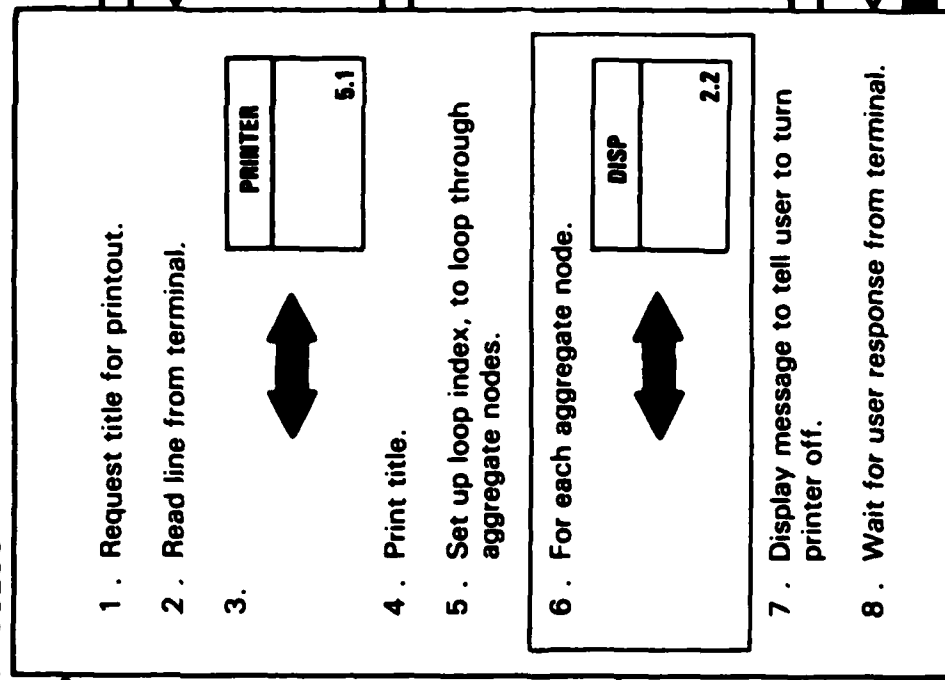
SCORES



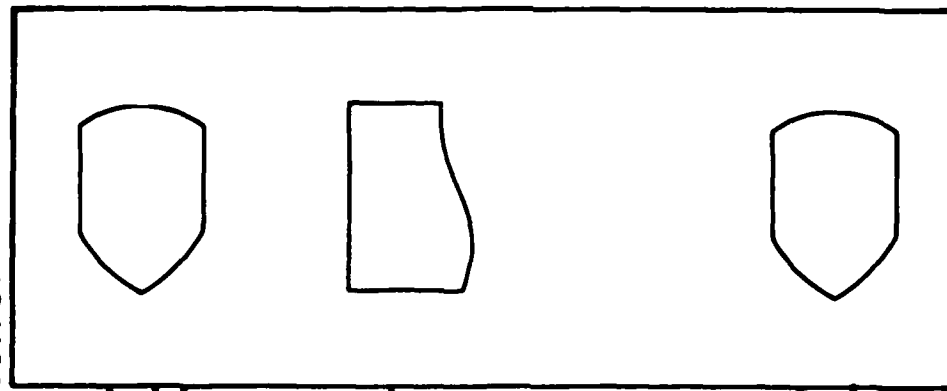
INPUT



PROCESS



OUTPUT



System/Program: RUN

Name: PRINTER

Diagram ID: 5.1

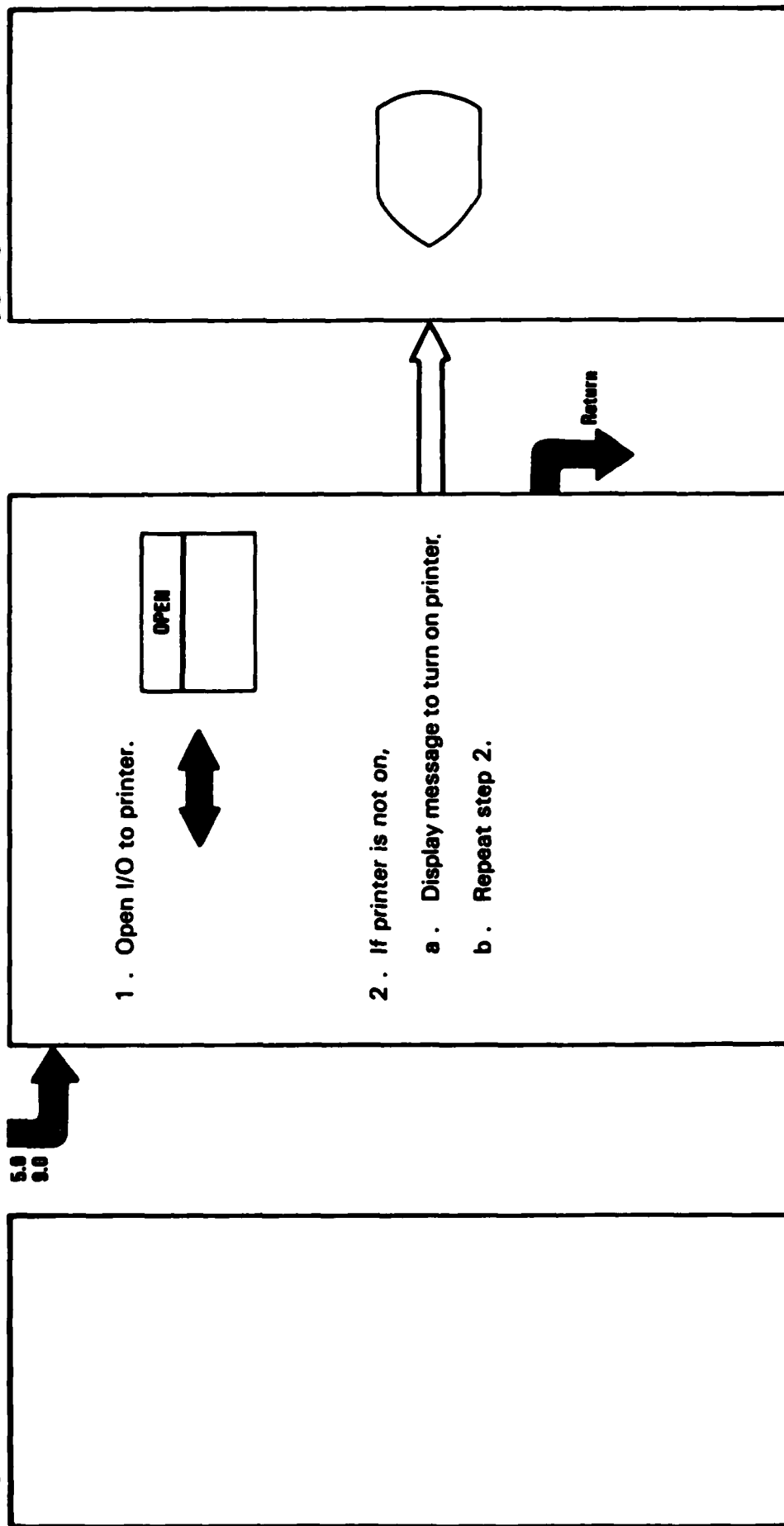
Description

Page: of

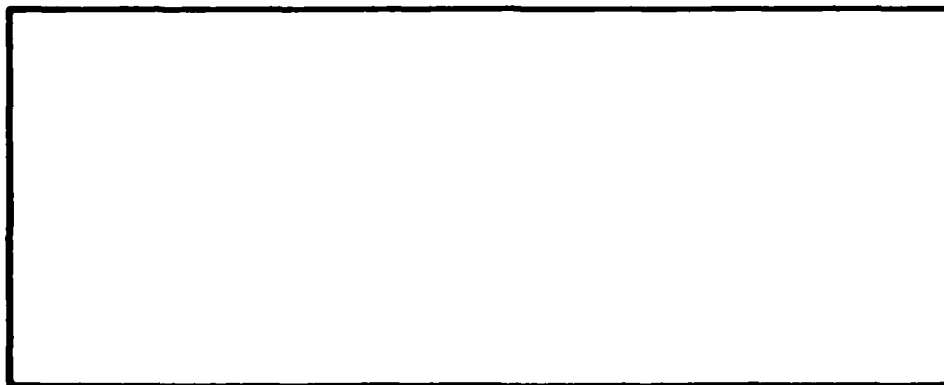
INPUT

PROCESS

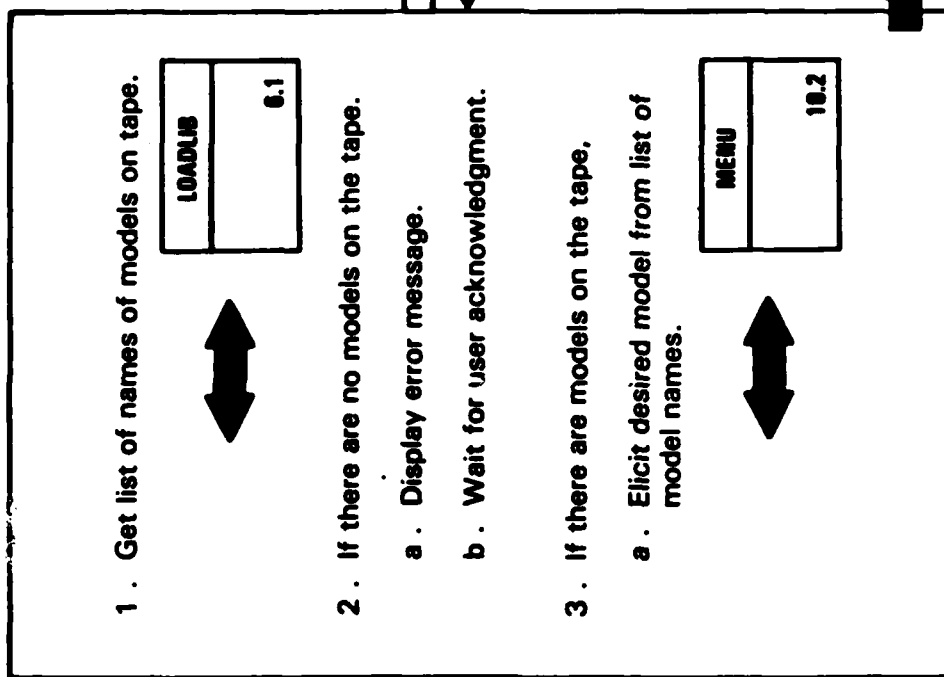
OUTPUT



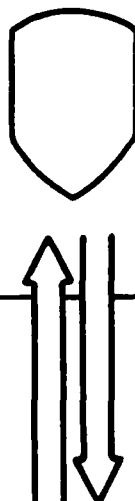
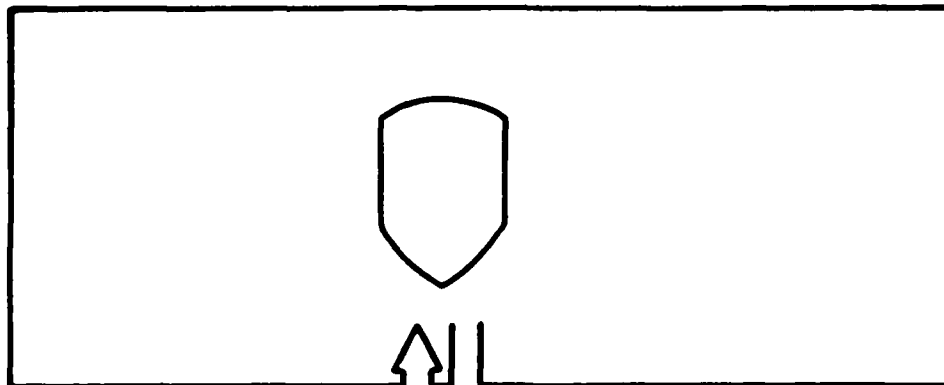
INPUT



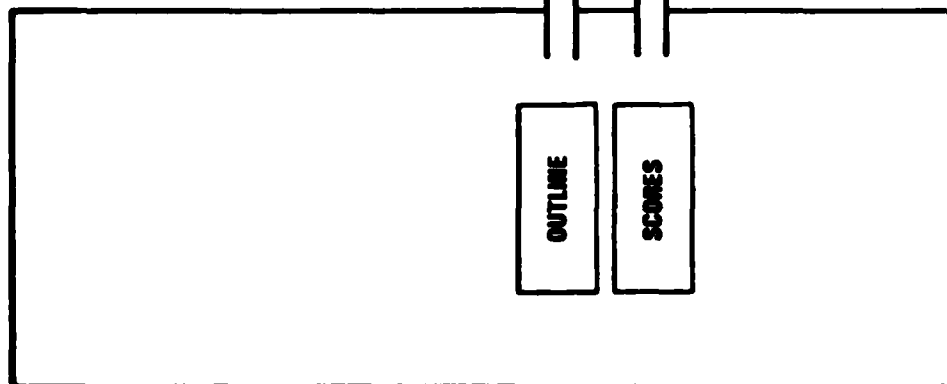
PROCESS



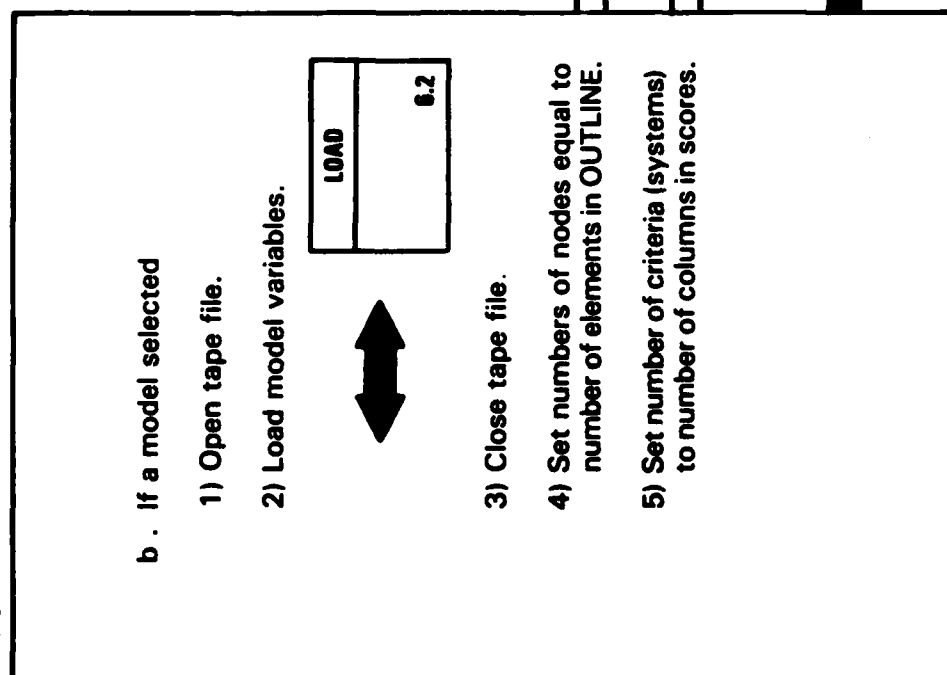
OUTPUT



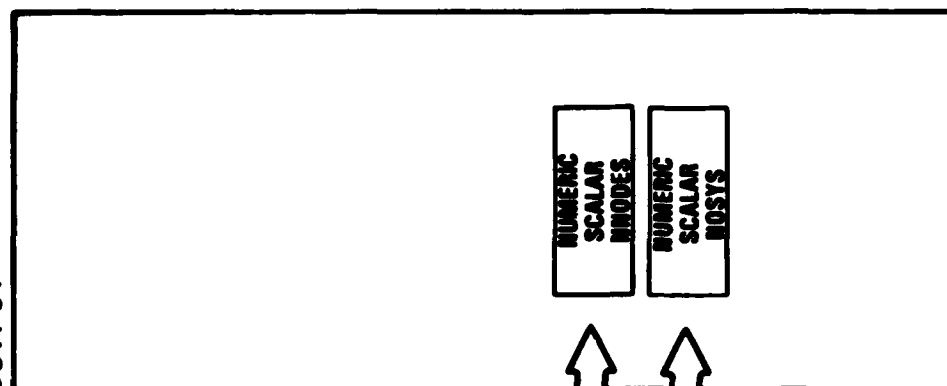
INPUT

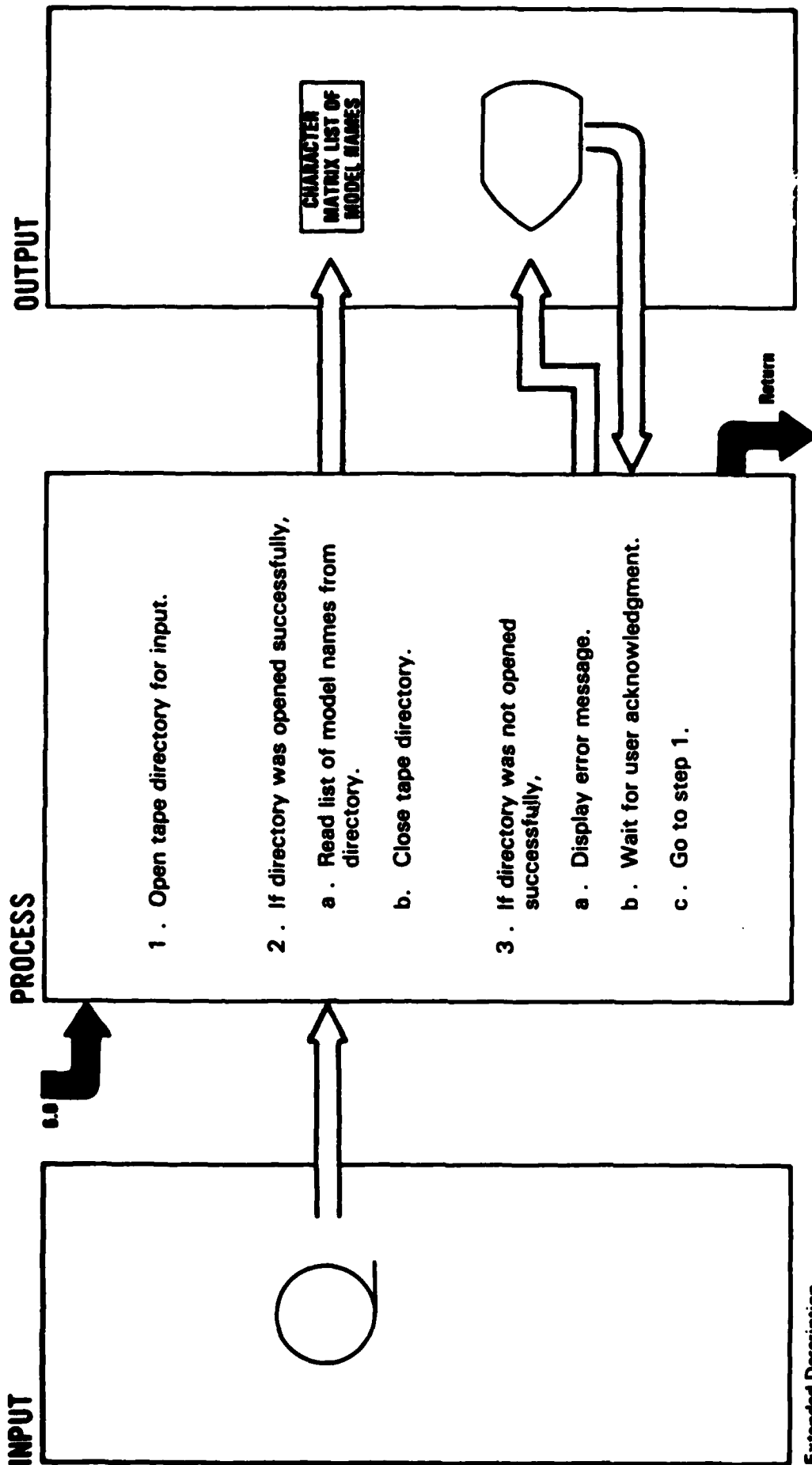


PROCESS



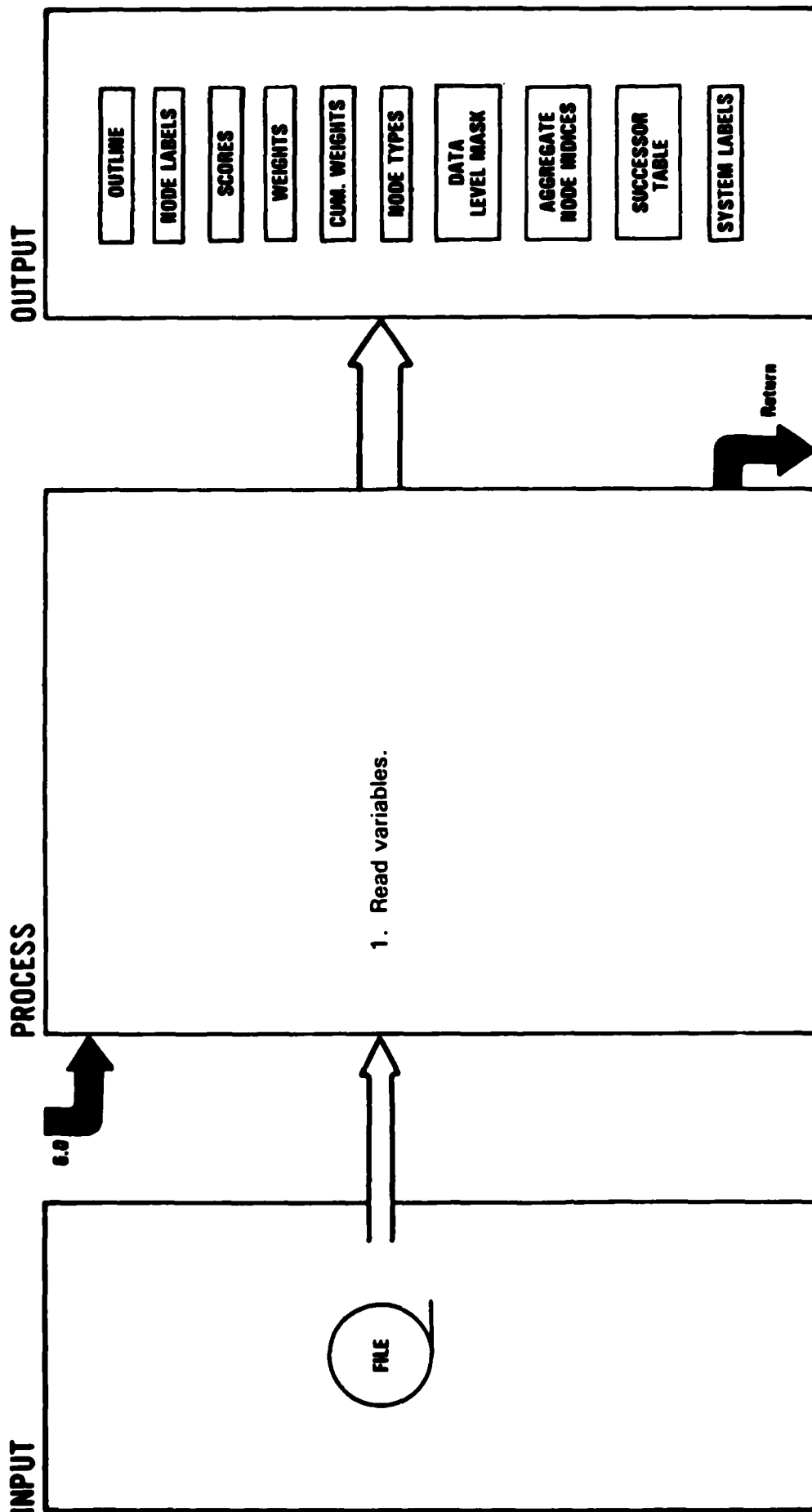
OUTPUT





Extended Description

Position of model names within list indicates where models are stored on tape.



Extended Description

1. The **OUTLINE TABLE** contains an element for each node in the model, sorted in increasing numerical sequence order. The value is an encoded representation of the node outline number supplied for a node when the model structure is created.
2. The **NODE LABELS** contain descriptions (one per node in the same order as the outline table) of nodes that are supplied when the model structure is created.
3. **SCORES** is a numeric array which contains a set of values for each node of the structure. Each set of values consists of one number per system defined in the model.
4. **WEIGHTS** is a numeric vector containing the relative-importance values assigned to each node in the model structure. The elements must appear in the same order as the associated outline numbers. When a model structure is created, the vector is null or contains zeros.
5. For each element in the node outline table, there is an associated element in the **CUMULATIVE WEIGHTS** vector. The vector will contain the percentage of importance with respect to the entire model when all **WEIGHTS** have been entered.
6. The **NODE TYPES** are indicators of the type of calculation that is to be used in assessing **SCORES** and **WEIGHTS**.

INPUT

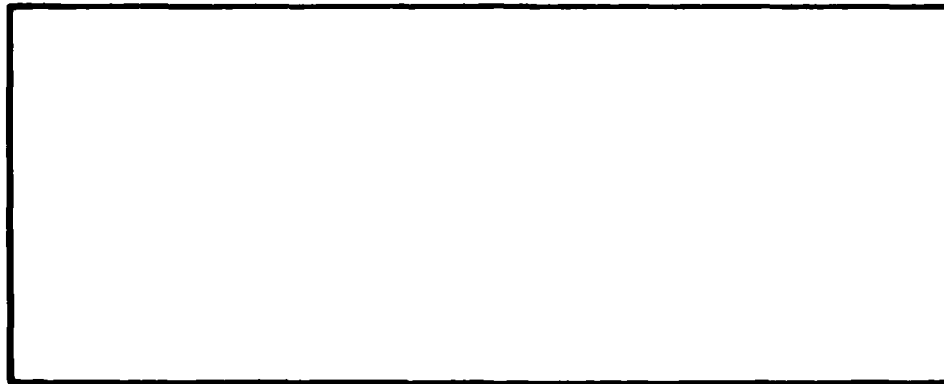
PROCESS

OUTPUT

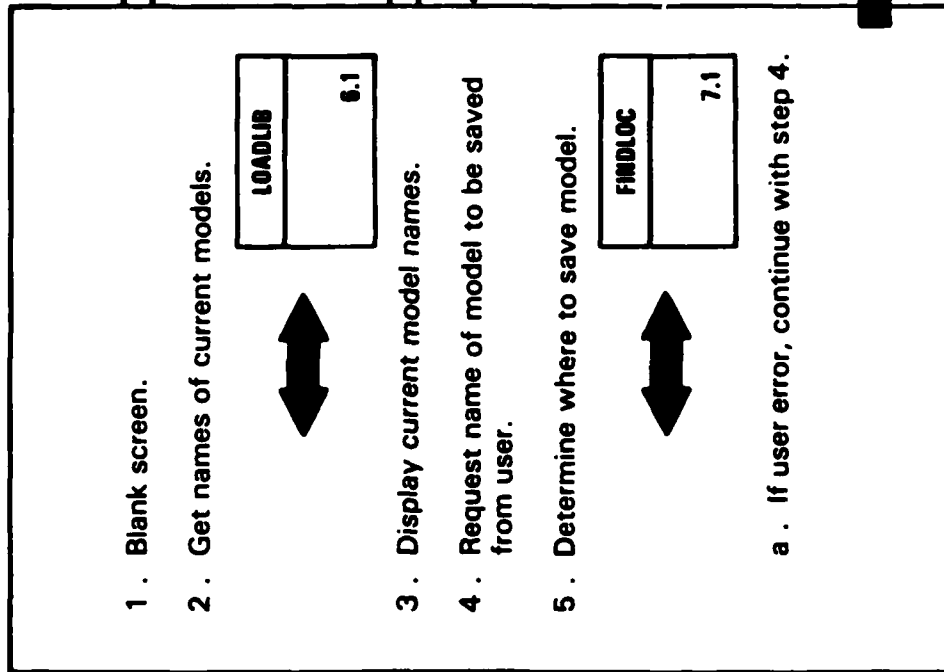
Extended Description

7. The DATA LEVEL MASK indicates which nodes are at the data level (bottom level) versus the nodes that are aggregate or non-bottom-level nodes.
8. The AGGREGATE NODE INDICES contain the sequence number of elements in the model variables which correspond to only the aggregate nodes. An Aggregate node is a node which has one or more subsequent nodes contributing to it.
9. The SUCCESSOR TABLE is an array which contains, for each aggregate node, the set of indices of nodes which contribute to a node.
10. The SYSTEMS LABELS contain the user-specified character descriptions of the systems being evaluated.

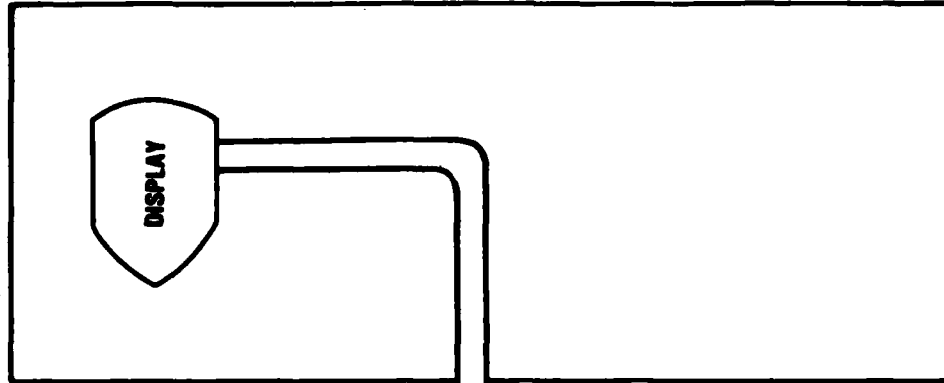
INPUT



PROCESS

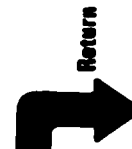
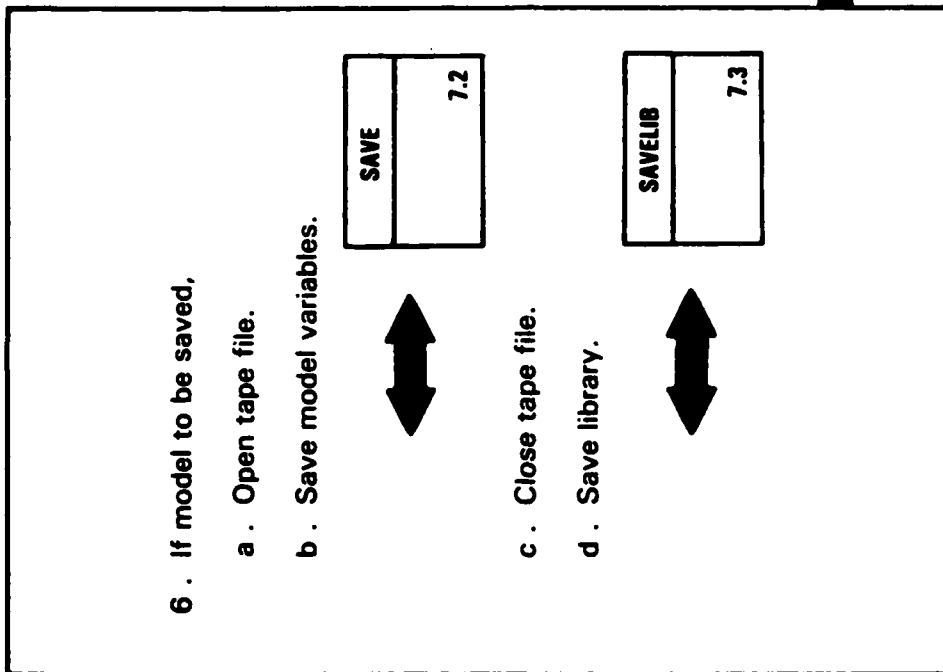


OUTPUT



INPUT

PROCESS

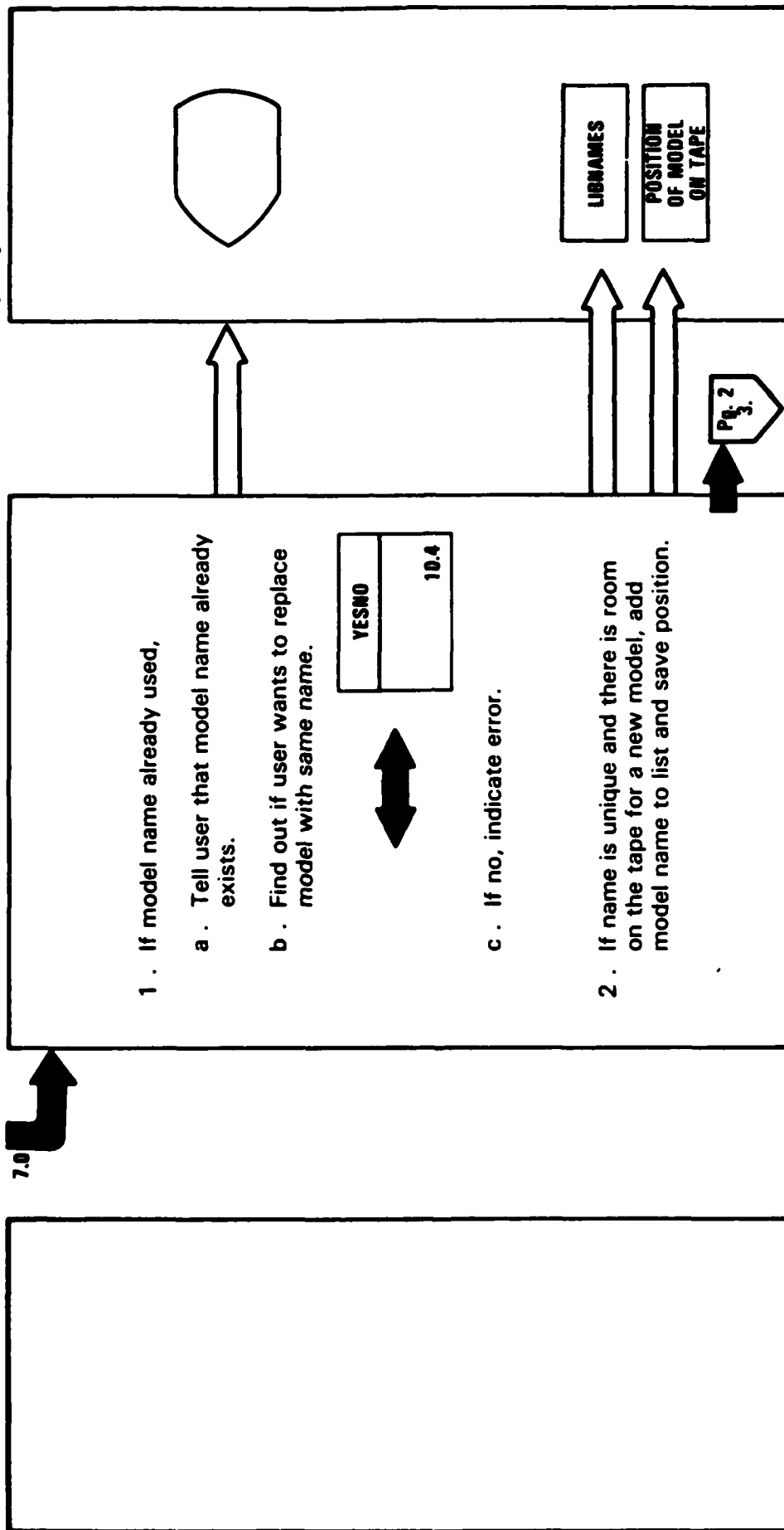


OUTPUT

INPUT

PROCESS

OUTPUT



INPUT

PROCESS



3. If name is unique but there is no room for another model, display current model names and ask user which model to replace.

MENU	
	10.2



- a. If model name selected, replace old model name with new model name in list, and save position.

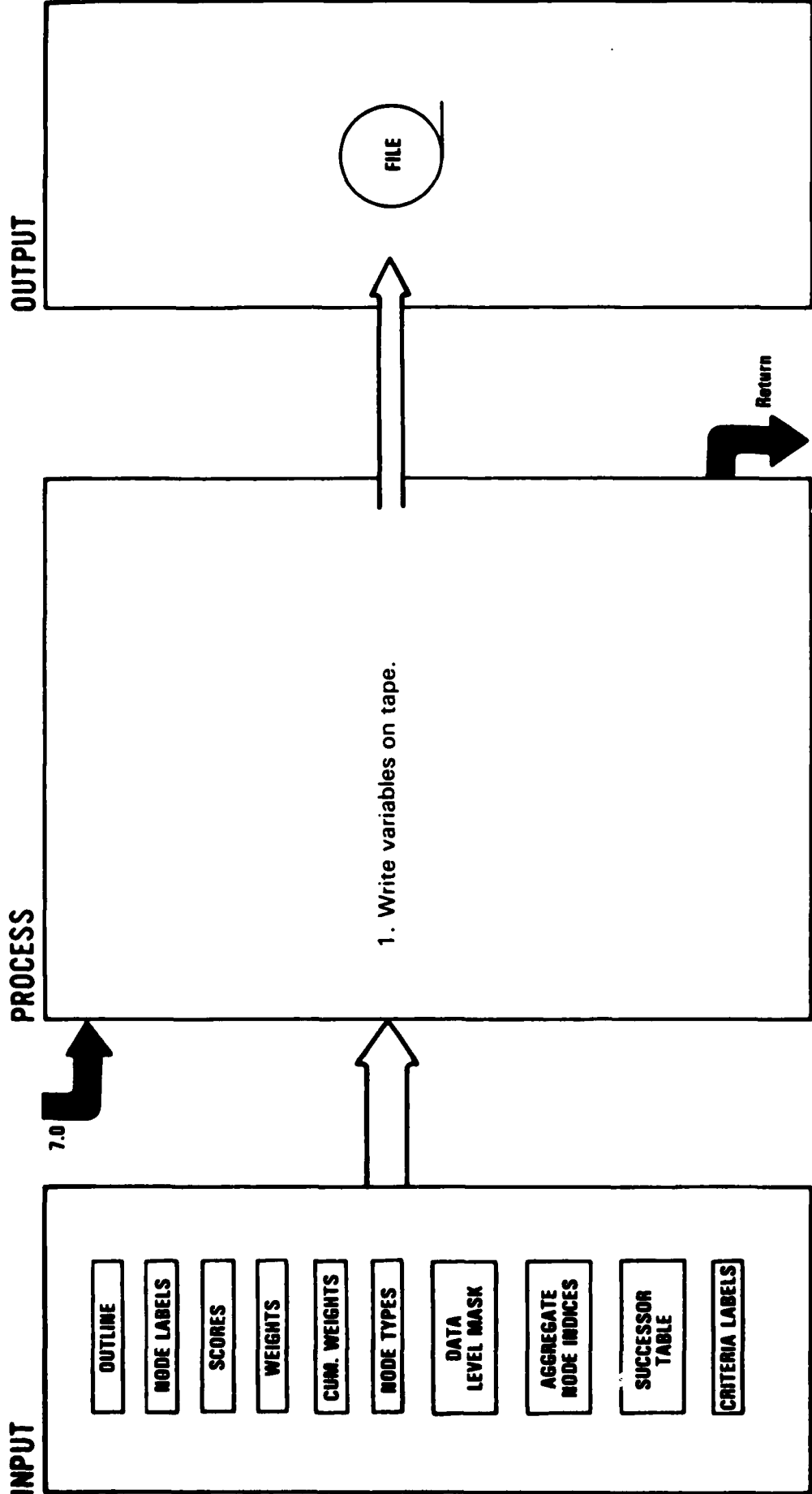
Return



OUTPUT

LIBNAMES

POSITION
OF MODEL
ON TAPE



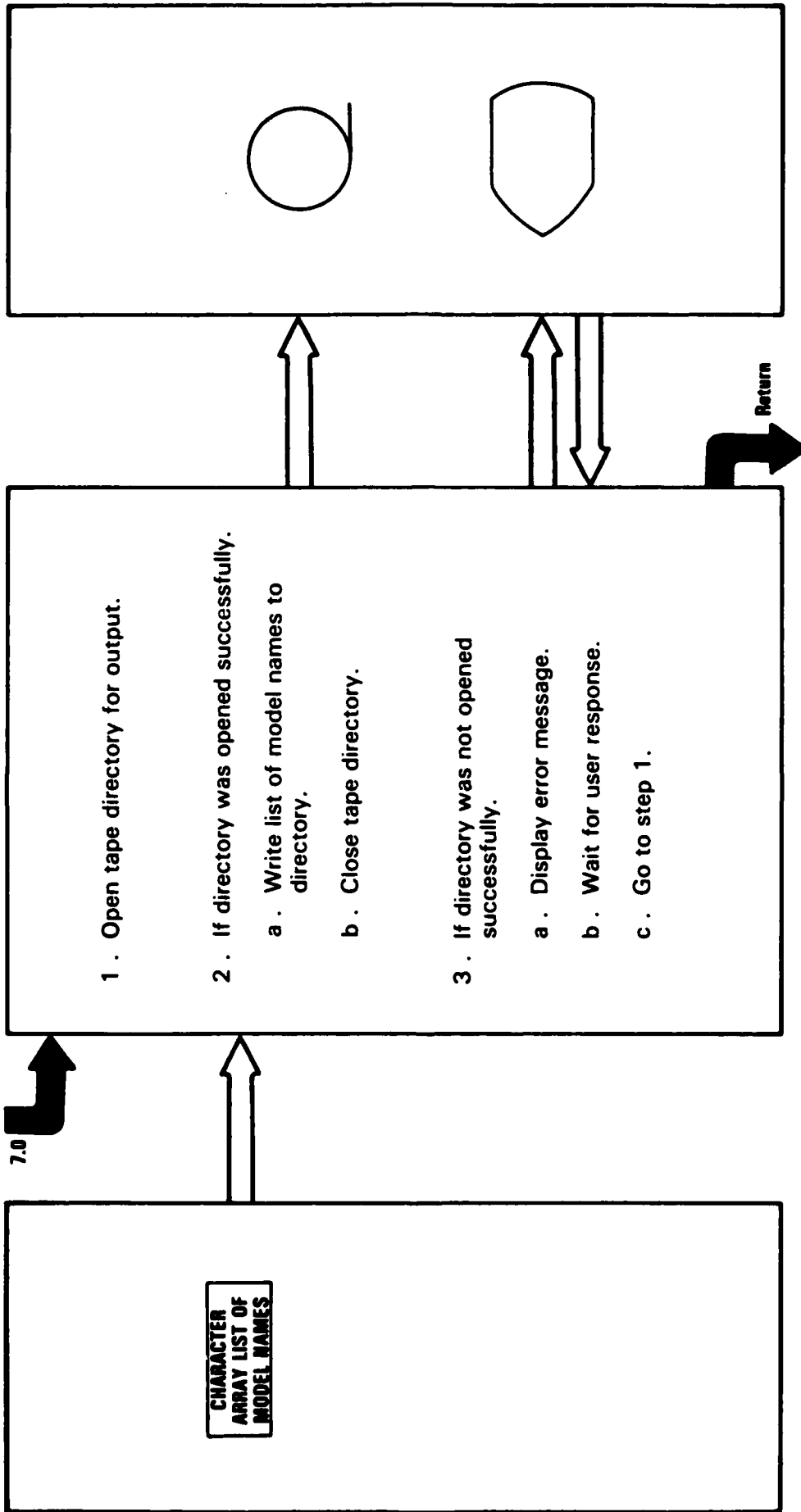
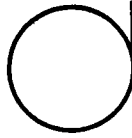
INPUT

CHARACTER
ARRAY LIST OF
MODEL NAMES

PROCESS

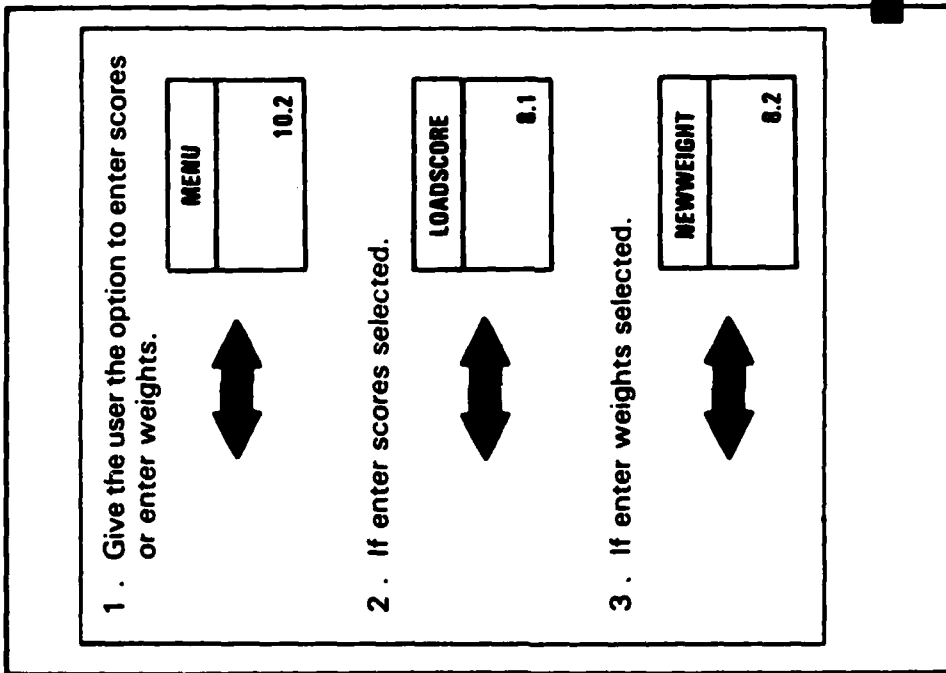
- 1 . Open tape directory for output.
- 2 . If directory was opened successfully.
 - a . Write list of model names to directory.
 - b . Close tape directory.
- 3 . If directory was not opened successfully.
 - a . Display error message.
 - b . Wait for user response.
 - c . Go to step 1.

OUTPUT



INPUT

PROCESS



OUTPUT

INPUT

PROCESS

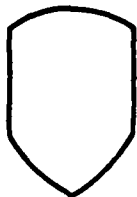
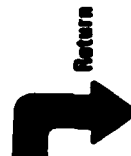
OUTPUT

Pg. 1
3.

4 . When no option selected display message that model is being recalculated.

5 . Calculate model.

ROLLBACK	4.2
----------	-----



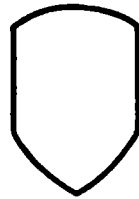
INPUT

INNODES, OUTLINE,
INOSYS, NODE
LABELS, DATA
LEVEL MASK

PROCESS

- 1 . Display character vector of instructions.
- 2 . Initialize loop index for scores and preset all scores to zero.
- 3 . For each node:
 - a . If data level node,
 - 1) Passing number of systems and character vector of node outline number and node label, request value for each system.
 - b . If not data level node
 - 1) Display node outline number and node label.
 - 2) Set all systems in scores to zero.

ENTERLINE
3.1.1

**OUTPUT****SCORES****LOOP
INDEX****SCORES****Return**

System/Program: RUN

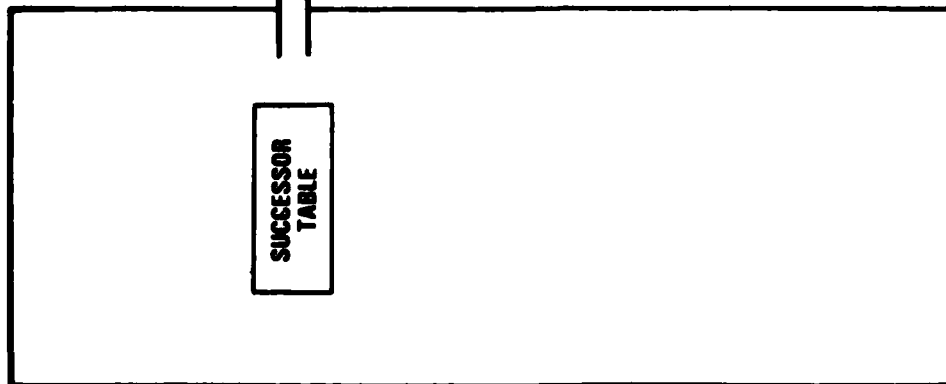
Diagram ID: 8.2

Name: NEWWEIGHT

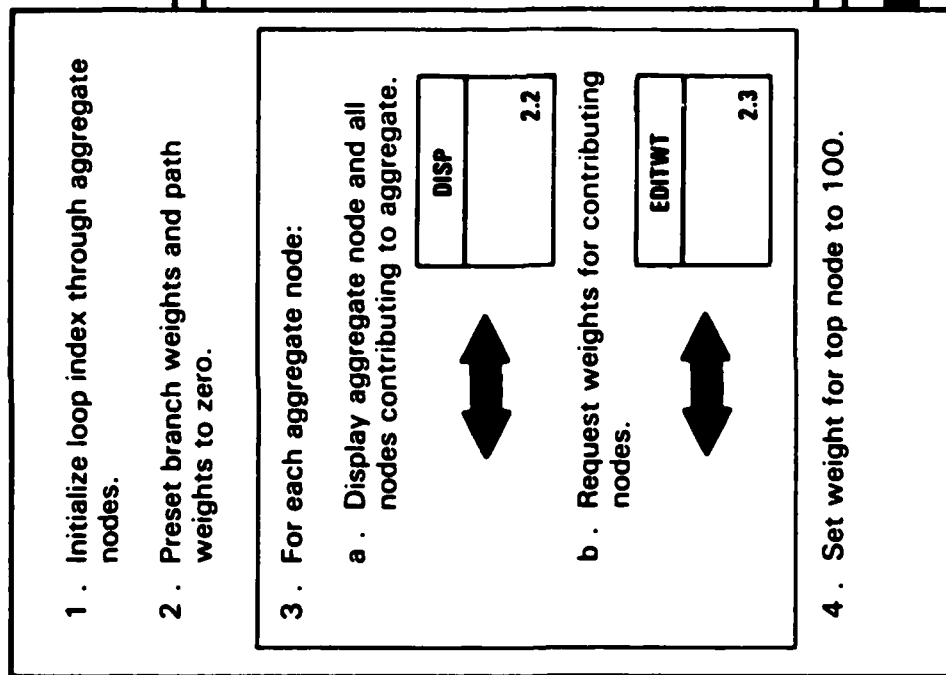
Description: Enter All Weights

Page: of

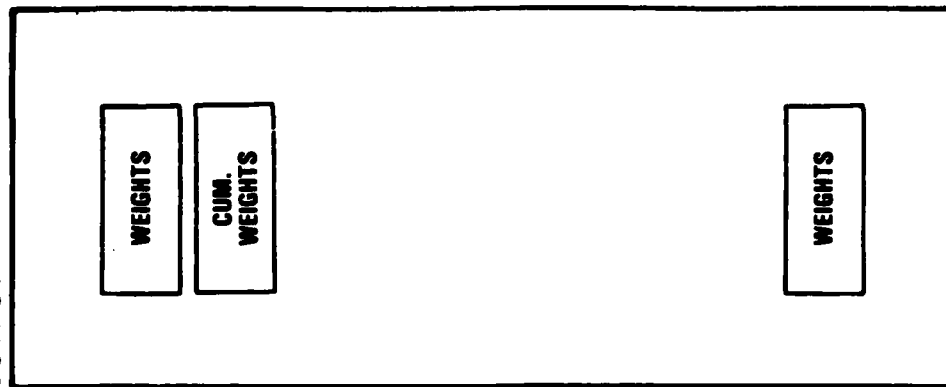
INPUT



PROCESS



OUTPUT

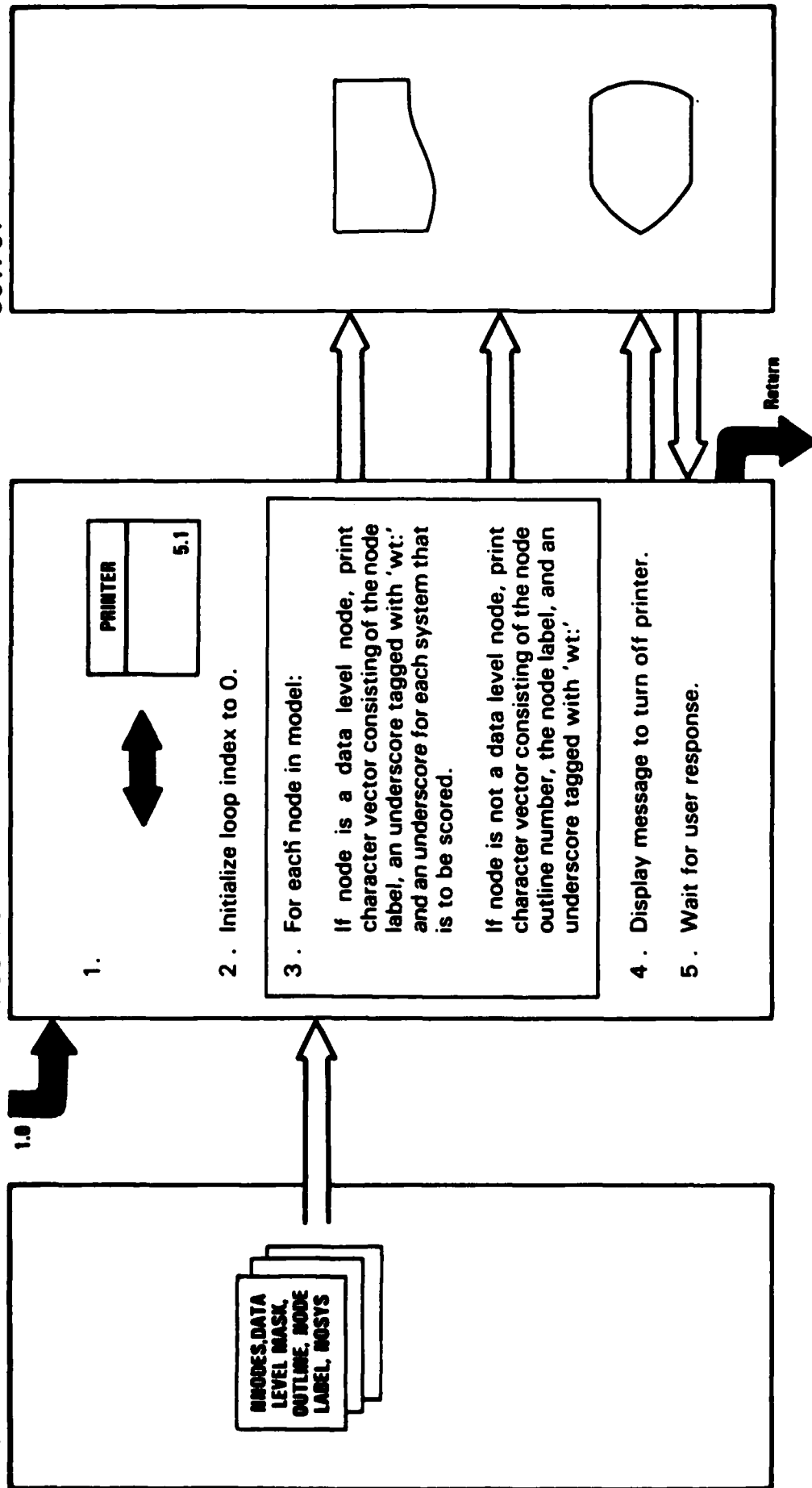


Return

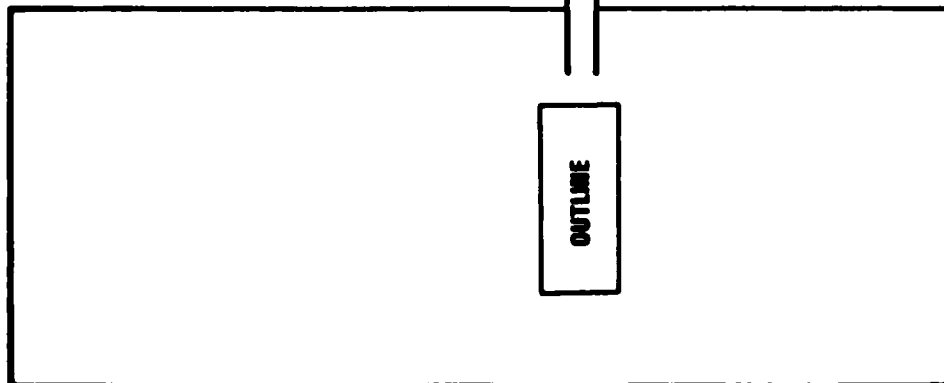
INPUT

PROCESS

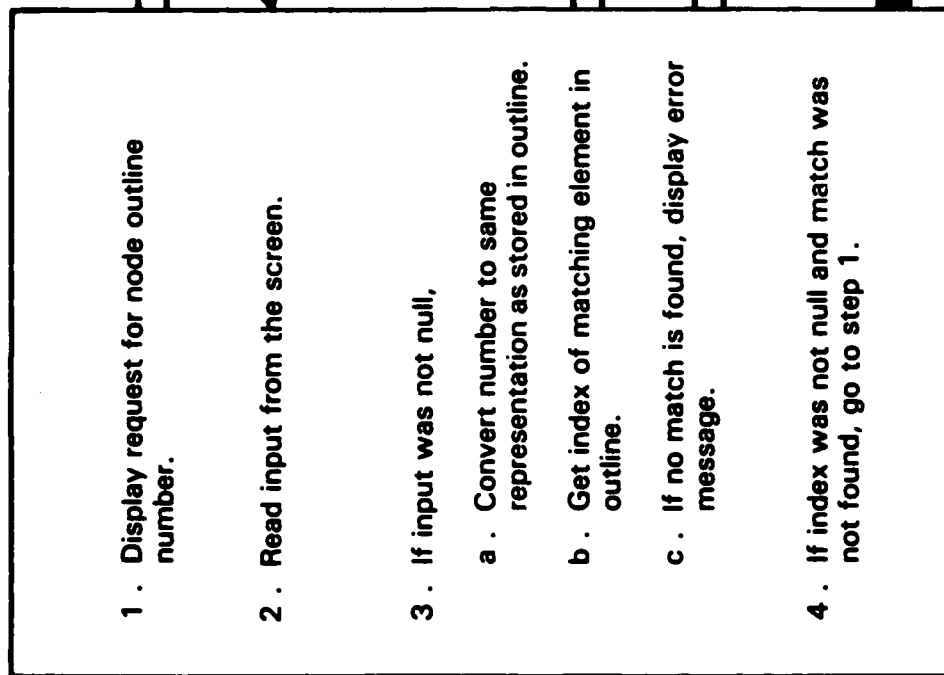
OUTPUT



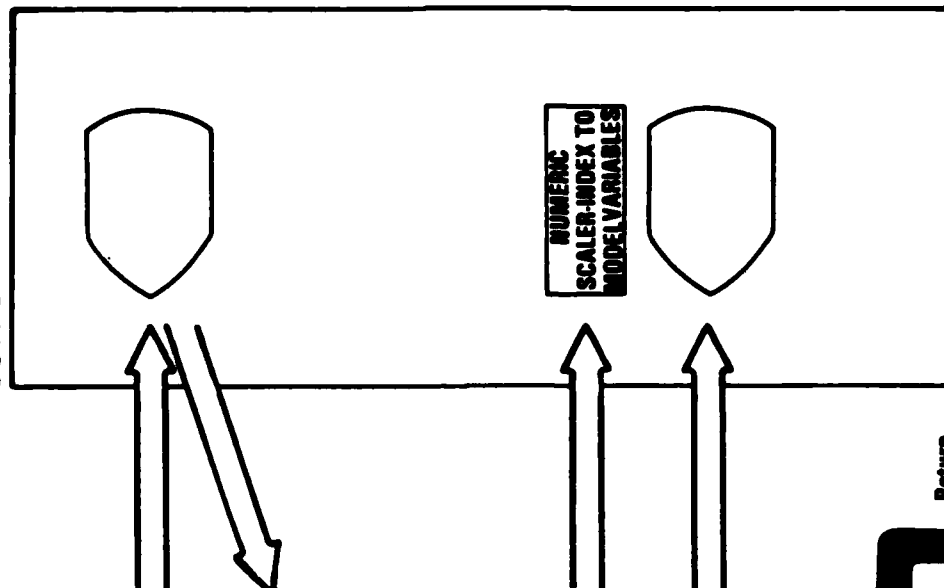
INPUT



PROCESS



OUTPUT



Return

PROCESS

1. Display character vector consisting of user instructions.
2. Display character array consisting of possible logic paths for user to select.
3. Accept input from terminal.
4. Set result to the number of the option selected by user, or to zero if no selection made.
5. If the result is greater than the number of options presented, change the result to contain the number of the last option.

INPUT

CHARACTER VECTOR
CONSISTING OF
INSTRUCTIONSCHARACTER
ARRAY
CONSISTING OF
OPTIONS AVAILABLE
FOR EXECUTION

OUTPUT

RESULT:
0 IF NO OPTION
SELECTED, OR
NUMBER OF
SELECTED OPTION

Return

