JASP:
A SIMULATION LANGUAGE
FOR A TIME-SHARED SYSTEM

A. Alan B. Pritsker

prepared for
UNITED STATES AIR FORCE PROJECT RAND
JASP:
A SIMULATION LANGUAGE
FOR A TIME-SHARED SYSTEM

A. Alan B. Pritsker

This research is supported by the United States Air Force under Project RAND — Contract No. AF4620-67-C-0045 — monitored by the Directorate of Operational Requirements and Development Plans, Deputy Chief of Staff, Research and Development, HQ USAF. Views or conclusions contained in this study should not to be interpreted as representing the official opinion or policy of Rand or of the United States Air Force.
PREFACE

Simulation has become a valuable Air Force tool for studying problems of inventory and maintenance management, scheduling, and manpower allocations. This Memorandum describes JASP, a language developed for writing simulation programs involving manual intervention and for facilitating the use of a time-sharing system to perform small-scale simulations. The need for such activity is evidenced both by RAND's own simulations of Air Force activities and by the assistance RAND has provided to Air-Force initiated efforts in this area. To illustrate areas for which JASP was designed, this study presents simulations of a simple queueing system and a man-machine price and inventory system.

A digital computer simulation program contains a model of the system being simulated and methods for exercising it. Complex data processing functions are required to manipulate input data, to maintain status information on the system and to collect statistical data for analysis. The JASP language provides the means for performing these functions so that the user is free to devote his energies to the model-building portion of a simulation program.

JASP uses as a host language, JOSS, the RAND-developed time-sharing language. Among the features that make JASP practical and usable in the JOSS environment are the overlaying of program segments and the reduction of storage requirements by dynamic storage allocation, and packing words in memory. While these features are normally used at the expense of computing time, JASP is designed for small problems or simulations involving man-machine interaction, and computing time is not as critical here as in other applications.
While this report describes JASP for the potential user, it was found necessary to provide information dealing with the language's structure and coding. The Memorandum, therefore, is a combination User's Manual and Programmer's Guide, and it is assumed throughout that the reader is familiar with JOSS and acquainted with SIMSCRIPT or GASP II.

Though JASP is designed for use as a simulation vehicle, parts of the language can be used directly. For instance, the information storage and retrieval methods can be used for list processing. To simplify the orientation of the report, JASP elements will be presented in terms of simulation applications, and allusions will be made to non-simulation uses where appropriate.

JASP is available for use without modification by anyone at RAND and in the Air Force having access to a JOSS console. The methodology has been presented in some detail to facilitate the translation to other languages for the benefit of potential Air Force users equipped with time-sharing systems other than JOSS.*

The author is a consultant to the Management Sciences Department of The RAND Corporation.

*JOSS is the trademark and service mark of The RAND Corporation for its computer program and services using that program.
SUMMARY

JASP, a simulation language for use on the JOSS time-shared system, is written in the JOSS language and provides standard routines for performing the functions common to many simulations. Possibly more important, JASP provides a conceptual framework which makes the writing of simulation programs on JOSS relatively easy. Because JASP is JOSS-based, simulations involving manual interaction can be written using the JOSS capability for such interaction. Translation of JASP for use on other time-sharing systems available to Air Force users should not be difficult.

JASP provides routines for performing the following functions:
1. Initialization;
2. Time and event control;
3. Information storage and retrieval;
4. Performance data collection;
5. Summary, monitoring and error reporting; and

The report describes each of the routines for performing the above functions. JASP variables are represented by JOSS capital letters, and their special definitions are described in detail. Simulations of a simple queueing system and a man-machine price and inventory system are used to illustrate areas for which JASP was designed. These demonstrate the JASP approach and language, and illustrate the relative ease with which an analyst can write a simulation program.

The Memorandum and its appendices provide complete documentation and ready-reference for JASP statement types, part interactions, definitions of variables, and the JOSS coding. All items are stored in file 704 (p8260), which can be recalled by any RAND or USAF user having access to a JOSS console.
ACKNOWLEDGMENTS

The concept of a simulation language for JOSS evolved during discussions with Louis W. Miller and Barbara C. Markowitz. Their research on VIMCOS II\(^4\) provided the launching pad for JASP. Their inputs and help are gratefully acknowledged. I would also like to thank Richard J. Kaplan for his participation in an early version of the price-inventory simulation and for making several useful suggestions for improving that simulation. And finally, I am grateful to Morton Berman for his many valuable suggestions.
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREFACE</td>
<td>iii</td>
</tr>
<tr>
<td>SUMMARY</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>vii</td>
</tr>
</tbody>
</table>

**Section**

I. JASP OVERVIEW ........................................... 1

II. JASP VARIABLES AND FORMULAS ............................... 6
   - Storage of Entities and Events ...................... 6
   - Attributes of Sets .................................. 10
   - Performance Variables ............................... 12
   - Get and Put Formulas ................................ 12
   - Random Deviates and Parameter Storage .............. 13
   - Definitions of W, M, and Q .......................... 14

III. JASP STORAGE AREAS .................................... 15
    - The File Storage Array, R .......................... 15
    - Storage of Attributes of Sets ...................... 19
    - Statistical Storage Arrays ....................... 20

IV. JASP STATEMENTS ......................................... 22
    - Initialization ..................................... 26
    - Time and Event Control ............................. 31
    - Information Storage and Retrieval ................ 32
    - Data Collection .................................... 37
    - Summary Reporting ................................ 40
    - Monitoring Events and the Status of Sets .......... 42
    - Error Reporting .................................... 43
    - Obtaining Random Numbers and Random Deviates ...... 44

V. USER PROCEDURES AND EXAMPLES ............................ 48

VI. CONCLUSIONS AND RECOMMENDATIONS ......................... 89

Appendix

A. GLOSSARY OF JASP VARIABLES .............................. 91
B. JASP STATEMENT TYPES ................................... 94
C. JASP PART DESCRIPTION AND INTERRELATIONS .............. 96
D. JASP ITEM--PART IDENTIFICATION ........................ 99
E. LISTING OF JOSS CODING OF JASP PARTS .................. 100

REFERENCES .................................................. 119
I. JASP OVERVIEW

The JASP simulation language is a collection of JOSS statements and specially defined JOSS variables. A complete simulation program is obtained when JASP is supplemented by problem-oriented statements.

The JASP framework for creating a simulation model follows that employed in SIMSCRIPT and GASP, and looks at systems in terms of entities, defined as any distinguishable object, being or processing unit of a system. Entities are characterized by their attributes. For example, a customer of a store is an entity. He may possess attributes like time of arrival, cash available, etc. Entities with common characteristics can be grouped together in sets. All customers waiting to be served form a set of waiting customers. Sets also have attributes like the number of entities in the set, the first entity in the set, etc.

The status of a system is described in terms of the attributes associated with entities, or sets of entities. The status of the system changes whenever the value of one of these attributes changes. When the value of an attribute changes, an event has occurred. In JASP, the converse is also assumed, i.e., values of attributes can only change when an event occurs. The simulation of a system involves establishing a set of potential events or event notices. A set of potential events contains all possible events that can occur as the next event. At the end of one event, time advances to that of the next event, and program control is transferred to the JOSS coding that represents the next event. Simulation of a system involves passing from one event to the next in the proper chronological order, where at
each event one or more of the following occurs: (1) decisions are made; (2) the values of attributes of entities or sets of entities are changed; (3) new possible next events are established; and/or (4) performance measures are updated. The premise of this viewpoint is the isolation of independent events. The event is the important element and a simulation program can be designed by modeling each event.

In JASP, this isolation of events is used as the basis for the definitions of JASP variables, parts and ítems (groups of parts). Definitions are made so as to assist the JASP user in coding the problem-oriented events that can occur during a system simulation. Figure 1 shows a conceptual framework for simulation programs written in JASP. The shaded portion of this figure shows the user's coding responsibilities. To create a complete simulation program, the user need only provide the JOSS code representing what takes place when an event occurs, initialize the non-JASP variables, and identify the part numbers associated with each event. JASP provides support routines for maintaining the set of next possible events, sets of entities, performance measure updating, intermediate and summary reporting, and the generation of random deviates.

The following functions are provided by JASP:

1. The executive routine that determines the next action to be performed during the simulation.

2. On-line initialization of JASP variables, sets and the parameters of random variables.

3. Operations on sets

   3.1 Filing an entity in a set.
Fig. 1 -- Conceptual framework for JASP Simulation Programs
3.2 Removing the first entity from a set.
3.3 Removing a specific entity from a set.
3.4 Retrieving attributes of an entity from a set, without removing the entity from the set.
3.5 Finding an entity in a set whose attributes meet specified conditions.
3.6 Calculating set characteristics (such as the sum of the values of Attribute 1) for all entities in a set.
3.7 Maintaining performance statistics on sets, such as the average number in a set.

4. Collection of values on performance variables.
5. Calculation of statistics for performance variables.
6. Presentation of statistics on performance variables and sets.
7. Generation of random numbers and random deviates from the uniform, normal, lognormal and Erlang distributions.
8. Monitoring events or set status during a simulation.
9. Reporting errors when illogical conditions occur.

The user must become familiar with the JASP variables, statements and operating procedures. Because JASP is JOSS-based, its operating procedures are similar to those of JOSS. They will be explained through examples in Section V. JASP variables are defined in Section II. One array, called the filing array, is used to store all entities and sets of entities. Because of its importance. Section III is devoted to it. JASP statements are described in Section IV, with a complete listing by function presented in Appendix B. The interrelations between JASP parts are listed in Appendix C. A list of the
JASP parts included in each item is presented in Appendix D. A complete listing of JASP is given in Appendix E.
II. JASP VARIABLES AND FORMULAS

Of the fifty-two symbols available for variable names in JOS, twenty-five are reserved for use by the JASP routines. These are referred to as JASP variables and formulas; they include all the capital letters except 0. The use of these twenty-five JASP symbols will be discussed in this section. A glossary of the symbols is presented in Appendix A, an abbreviated list is presented in Figure 2.

STORAGE OF ENTITIES AND EVENTS

A user writing a simulation program in JASP frequently creates events and entities and files them in their appropriate sets. He also performs other operations on sets, such as finding or removing an entity. To perform these operations in JASP, each set must be given a numeric code. The variable I is used to identify this code; and I must be defined prior to performing any operation on a set. As a convention, the set with the code number equal to 1 is defined as the event set. When filing events into the event set, the user must write the statement: "Set I=1." or equivalently, modify a statement with "for I=1". The assignment of numeric code values to other sets is left to the user. Caution: The variable I is also used as an index for other purposes.

The number of attributes associated with an entity is defined by the user during the initialization procedure. When performing set operations, the vector A(·) is used to transfer attribute values to and from a set. Thus A(·) can be used as a buffer storage area.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Referenced by JASP User</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(·)</td>
<td>Buffer storage for attribute i of an entity.</td>
<td>yes</td>
</tr>
<tr>
<td>B(·,I)</td>
<td>Parameter values for distribution I.</td>
<td>no</td>
</tr>
<tr>
<td>C(·,I)</td>
<td>Storage area for data concerning X(I).</td>
<td>no</td>
</tr>
<tr>
<td>D(·,I)</td>
<td>Storage area for data concerning Y(I).</td>
<td>no</td>
</tr>
<tr>
<td>E</td>
<td>Event code of current event.</td>
<td>yes</td>
</tr>
<tr>
<td>F(·,I)</td>
<td>Attributes of set I.</td>
<td>no</td>
</tr>
<tr>
<td>G(0,L,Q,W)</td>
<td>Formula to &quot;get&quot; the D digits, located L digits from right with Q digits in fractional part of word W.</td>
<td>possibly</td>
</tr>
<tr>
<td>H(I,·)</td>
<td>Storage area for histogram for data concerning X(I).</td>
<td>no</td>
</tr>
<tr>
<td>I</td>
<td>An index; primarily a set number or a variable number.</td>
<td>yes</td>
</tr>
<tr>
<td>J</td>
<td>An index; primarily used for the column number of an entity.</td>
<td>yes</td>
</tr>
<tr>
<td>K(I)</td>
<td>Column number of the first entity in set I.</td>
<td>yes</td>
</tr>
<tr>
<td>L(I)</td>
<td>Column number of the last entity in set I.</td>
<td>yes</td>
</tr>
<tr>
<td>M(·)</td>
<td>Miscellaneous and temporary storage array.</td>
<td>yes</td>
</tr>
<tr>
<td>N(I)</td>
<td>Current number of entities in set I.</td>
<td>yes</td>
</tr>
<tr>
<td>O</td>
<td>Not used.</td>
<td>---</td>
</tr>
<tr>
<td>P(J)</td>
<td>Column number of predecessor of entity in row J.</td>
<td>possibly</td>
</tr>
<tr>
<td>Q</td>
<td>A zero when time attribute is not packed; otherwise number of digits in fractional part of time.</td>
<td>possibly</td>
</tr>
<tr>
<td>R(·,J)</td>
<td>Storage area for entity in column J.</td>
<td>no</td>
</tr>
</tbody>
</table>

Fig. 2 -- Definition of JASP Variables and Formulas
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Referenced by JASP User</th>
</tr>
</thead>
<tbody>
<tr>
<td>S(J)</td>
<td>Column number of successor of entity in column J.</td>
<td>possibly</td>
</tr>
<tr>
<td>T</td>
<td>Current time.</td>
<td>yes</td>
</tr>
<tr>
<td>U(D,L,Q,W,V)</td>
<td>Formula to &quot;put&quot; the value V with Q digits in its fractional part in the D digits of word W, located L digits from the right.</td>
<td>possibly</td>
</tr>
<tr>
<td>V(I)</td>
<td>Value of random deviate obtained from distribution with parameter values as stored in B(·,I).</td>
<td>yes</td>
</tr>
<tr>
<td>W(·)</td>
<td>Storage area for initial values.</td>
<td>possibly</td>
</tr>
<tr>
<td>X(I)</td>
<td>Performance variable I (as defined by the user) for which statistics based on observations are desired.</td>
<td>yes</td>
</tr>
<tr>
<td>Y(I)</td>
<td>Performance variable I (as defined by the user) for which statistics over time are desired.</td>
<td>yes</td>
</tr>
<tr>
<td>Z</td>
<td>Current random number.</td>
<td>yes</td>
</tr>
</tbody>
</table>

Fig. 2 -- (Continued)
Whenever an entity is to be filed in set I, A(·) must be defined with the number of attributes specified for set I. When an entity is removed from set I, JASP automatically establishes A(·) to represent the attributes of the entity removed from set I. A(·) can then be used directly in JASP. With the exception of the event set, the definitions of all the attributes of a set are established by the user. For the event set, A(1) is the time the event is to occur and A(2) is an event code. An event code is specified by the user to give a numeric value to the different possible events of his simulation. Because event time and event code are important for the current event, the symbols T and E respectively are reserved for these quantities. The current time in a simulation is thus always accessible by referencing the variable T. Similarly the current event code is E.

Each entity or event is stored in one column of an array R(·,J) where J is the column number and the first subscript (denoted by the ·) can be considered as a "word" number. The number of words associated with a column of R is determined by the set number of the entity stored in the column. If storage becomes critical, the method discussed in Section III for changing the format for R should be investigated. The standard format used in JASP for array R allows as many attributes per entity as desired. All attributes may take on any value consistent with JOSS, except for attribute 2 which must have an integer value from 0 to 999. The number of words associated with each column is the maximum of (2 and the number of attributes); there must be at least two words associated with the entities of any set.
JASP provides the means required to file, remove and locate entities in the array, R. Thus, from the user's standpoint, the array R need never be accessed directly. The means for calculating characteristics of entities in sets is also provided.

The entity preceding the entity in column J is called the predecessor of J. The special formula P(J) specifies the column number in which the predecessor of J is stored. The entity that follows the entity in column J is called the successor of J. S(J) is the formula that identifies the column number in which the successor of J is stored. For instance S(3) would be the column number of the successor of the entity stored in column 3.

The column numbers of the predecessor and successor of an entity can be considered as attributes of the entity. They are automatically established and updated by JASP. If an entity has no predecessor, the code value 999 is assigned as its predecessor row-number. Thus if the entity in column 2 has no predecessor, it is the first entity in its set and P(2) = 999. An entity that is last in its set has a code value of zero assigned as its successor's column-number.

ATTRIBUTES OF SETS

For each set I used in a JASP program, the following attribute information is maintained:

1. Number of entities in the set, N(I);
2. Column number of R of last entity in the set, L(I);
3. Column number of R of first entity in the set, K(I);
4. Maximum number of entities ever in the set;
5. Number of attributes for an entity in the set;
6. Number of words of storage assigned to an entity of the set;

7. The procedure for ranking entities in the set;

8. The attribute to be used for ranking entities in the set;

9. Time integrated number of entities in the set;

10. Time of last change in the number of entities in the set.

This information is stored for set \( I \) in the \( i \)th column of the two-dimensional array, \( F \). (The array \( F \) is not normally accessed directly by the JASP user).

Formulas are provided in JASP to retrieve the number of entities, the column number of the first entity and the last entity of set \( I \). The symbols used for these formulas are \( N(I) \), \( K(I) \), and \( L(I) \), respectively. As an example of the use of these formulas and the \( S(J) \) formula, consider the determination of the column number of the second entity in set 2. This is obtained as \( S(K(2)) \).

The ranking method (attribute 7 of a set) specifies the priority procedure for the set. The following special codes are used:

<table>
<thead>
<tr>
<th>Code</th>
<th>Ranking Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A low value is given priority, or Low Value First (LVF). For entities with the same value, the entity that was in the set first is given priority (LVF-FIFO).</td>
</tr>
<tr>
<td>2</td>
<td>A high value is given priority, or High Value First (HVF). For entities with the same value, the entity that was in the set last is given priority (HVF-LIFO).</td>
</tr>
<tr>
<td>3</td>
<td>First-In, First-Out (FIFO)</td>
</tr>
<tr>
<td>4</td>
<td>Last-In, First-Out (LIFO)</td>
</tr>
<tr>
<td>5</td>
<td>LVF-LIFO</td>
</tr>
<tr>
<td>6</td>
<td>HVF-FIFO</td>
</tr>
</tbody>
</table>

These codes and the ranking attribute number are established by the user for each set (except the event set, which is LVF ranked on time) during the initialization procedure.

*For LVF and HVF priority procedures, the secondary ranking procedure is based on the length of time an entity has been in the set.*
PERFORMANCE VARIABLES

Variables used to measure performance for which JASP is requested to collect and compute estimates must be referred to either by the symbol \( X(I) \) or by \( Y(I) \). The index \( I \) is used here to identify the code number of the performance variable. Code numbers are established by the user. When the user wishes to collect statistics on his coded variable number \( I \), he must set \( X(I) \) to its appropriate value before calling the part that maintains statistics on performance variables.

The \( X \)-variables are characterized by a value at a specified observation point; for instance, \( X(i) \) could be the dollar value of a proposal. The \( Y \)-variables are characterized by their possession of a value that persists over time; thus, \( Y(i) \) could be the dollar value of all outstanding proposals. Although \( X \) and \( Y \) are JASP variables, they are defined in terms of the JASP user's problem and the user must change their value to conform to the changes that occur at event times. JASP collects data so that estimates of the statistical properties associated with the \( X \) and \( Y \) variables can be calculated. Subsequently, these are automatically printed in the JASP summary report.

The storage arrays for maintaining data on the \( I^{th} \) \( X \)-variable are \( C(\cdot, I) \) and \( H(I, \cdot) \) where \( H(I, \cdot) \) is only used if a histogram of the values observed for \( X(I) \), is requested. For \( Y \)-variables, the array \( D(\cdot, I) \) is used. The arrays \( C, D \) and \( H \) are not normally accessed by the JASP user.

GET AND PUT FORMULAS

As core storage is a critical problem with JOSS, packing of words is a common occurrence. To insert a variable into part of a word
(Put) or to access the value of a variable that is part of a word (Get), two formulas are provided in JASP. The Get formula is \( G(D,L,Q,W) \) and the Put formula is \( U(D,L,Q,W,V) \) where

- \( D \) = number of digits associated with the value of the variable
- \( L \) = number of digits in word \( W \) to the right of the value of the variable
- \( Q \) = number of digits in fractional part of value of the variable
- \( W \) = the symbol of the word being packed or unpacked
- \( V \) = new value to be assigned to the variable.

The \( G \) and \( U \) formulas assume the nine digit JOSS word. Thus, the statement \( g = G(3,2,1,R(3,2)) \) would obtain the digits shown below from word \( R(3,2) \):

\[
\begin{array}{cccc}
XXX & \boxed{xxxx} & XX
\end{array}
\]

and specify a value for \( g \) in the form xx.x. The value of \( R(3,2) \) is not changed by a Get operation. If \( R(3,2) \) is 999321678, \( g \) would be assigned the value 21.6. The statement \( R(3,2) = U(3,2,1,R(3,2),17.4) \) changes the value of \( R(3,2) \) to 999317478. When using the \( G \) and \( U \) formulas, the user must not exceed the number of digits specified in the packed format and must make certain that negative values are not packed. Also, only one Put operation for a given variable can be performed on a given line.

**RANDOM DEVIATES AND PARAMETER STORAGE**

The symbol \( Z \) is used as the value of the current random number. It is initialized as the random number seed during the initialization procedure.

*The arguments are dummy arguments and are not JASP variables.
The symbol \( V(I) \) is used to return to the JASP user a deviate (sample) from a distribution whose parameters are stored in \( B(\cdot,I) \). It is the user's responsibility to specify the value of \( I \) so that parameter set \( I \) is used. The distribution type is specified by the user by a number in the statement he uses in requesting a value for \( V(I) \). The specific parameter values stored in \( B \) will be described in the section on generating random deviates.

**DEFINITIONS OF W, M, AND Q**

The symbol \( W(\cdot) \) is used to store six initial values set by the JASP user at execution time. These are

\[
\begin{align*}
W(1) & \equiv \text{initial value of time (start time)} \\
W(2) & \equiv \text{maximum number of sets ever required} \\
W(3) & \equiv \text{maximum number of X-variables} \\
W(4) & \equiv \text{maximum number of Y-variables} \\
W(5) & \equiv \text{maximum number of parameter sets} \\
W(6) & \equiv \text{initial random number}
\end{align*}
\]

These values are used for problem definition and print-out control purposes. Gross overestimates of \( W(2) \) through \( W(5) \) affect only final reporting; they do not affect storage requirements.

The variable \( Q \) defines the number of fractional digits used in describing time when time is packed into a word of storage. If time is not packed, \( Q \) is set to zero.

The symbol \( M(\cdot) \) is an array for storing miscellaneous data, as an index or for temporary storage. Specific uses of \( M(\cdot) \) will be explained in Section IV.
III. JASP STORAGE AREAS

The three main storage areas for JASP variables are: (1) the file storage array, R; (2) the array for storing attributes of sets, F; and (3) the statistical storage arrays, C, D and H. The formats for these arrays will be discussed in this section. As previously described, a JASP model considers that a system is composed of entities that are characterized by attributes and that can be grouped together in sets. The filing mechanism performs the necessary housekeeping functions of storing, sorting, ranking, locating, updating and purging entities of sets. The filing mechanism consists of an independent group of parts that can be used for any application which includes filing operations, for instance, list processing.

THE FILE STORAGE ARRAY, R

A two dimensional array, R, is used to store the entities (entities and event notices) of all sets. Each column** of R corresponds to one entity and contains: (1) the attributes associated with the entity; and (2) pointers indicting the entity's predecessor and successor in the set of which it is a member. Each word of a column is used to store attributes and/or pointers. Because R is used to store many entities and events, it is critical that an efficient storage format for R be designed. This is a difficult task because the number of attributes associated with entities is problem oriented.

*This section involves a technical discussion of how JASP is coded and may be skipped.

**The first subscript will be referred to as the row- or word-number and the second subscript as the column-number, e.g., R(2,1) would be word 2 of column 1 of the array R.
To alleviate this, JASP was designed to permit the user to specify the format for the words of R. Also, a standard format is provided. If the standard format is not used, the user must specify the desired alternate format, and reprogram parts 308 and 309. The user can specify the number of attributes/entity and the number of words/entity for each set; but word 1 must in every case contain the predecessor and successor variables, as specified in the standard format.

We will now describe the programming of parts 308 and 309 for the standard format and two other formats. The standard format for a column of R is (recall A(·) is the buffer storage array that allows the user to communicate values to R):

| Word | 1 | 2 | 3 | 4 | ...
|------|---|---|---|---|---
| Format | xxx xxx xxx xxxxxxxxxx xxxxxxxxxx xxxxxxxxxx ...
| Array label | R(1,J) R(2,J) R(3,J) R(4,J) ...
| JASP label | P(J) S(J) A(2) A(1) A(3) A(4) ...

For this standard format, all sets have the same word specification, although each set can have a different number of attributes per entity. The number of attributes/entity is not limited;* the coding will allow for A(5) to be put in R(5,J), etc.

The procedure for filing an entity into R is as follows: JASP locates a column, J, that is available for storing A. JASP requests part 309 to insert the buffer vector, A, into column J of the filing

*Although the number of attributes/entity is not limited, when Forms 1 and 2 are used for printing attributes they are designed to accommodate at most six attributes/entity. They should be changed if more than 6 attributes/entity are used.
array $R$. JASP then automatically determines the values for $P(J)$ and $S(J)$ and stores them into $R(1,J)$ in the appropriate position.

When JASP requests that part 309 be executed for each attribute of the entity, the variable $M(7)$ is used to index the attribute number from 1 to the number of attributes associated with set 1. For the standard format, the coding for part 309 is:

309.1 Set $R(2,J) = A(1)$ if $M(7) = 1$.
309.2 Set $R(1,J) = A(2)$ if $M(7) = 2$.
309.3 Set $R(M(7),J) = A(M(7))$ if $M(7) > 2$.

Here it is assumed that $0 \leq A(2) < 1000$, so that only the three least significant digits of $R(1,J)$ are affected. $R(1,J)$ can be set to $A(2)$, as the pointers are inserted into $R(1,J)$ after part 309 is executed. $A(1)$ and $A(2)$ are out of order because it permits a one attribute entity which is not limited to 3 digits. Note that $M(7)$ is not restricted in value and, in particular, can be greater than 4.

Part 308 assigns to the buffer array $A$, the attribute values stored in column $J$ of set 1. For the standard format the coding for part 308 is:

308.1 Set $A(M(7)) = [M(7) > 2: R(M(7),J); M(7) = 1: R(2,J); 1000 - fp(R(1,J)/1000)].$

In step 308.1, unpacking word 1 was done directly since it will be faster than using the general Get formula, $G(3,0,0,R(1,J))$. As for part 309, part 308 is called for each appropriate value of $M(7)$.

*This is an attribute of a set and as such is stored in the array $P$. 
as specified by the attribute of the set (stored in array F) for which J is an entity.

Consider next a specification for R in which all files have the same format for the attributes, and a maximum of six attributes and three words of R are associated with any one entity. Suppose the format for a column of R is as shown below:

<table>
<thead>
<tr>
<th>Word number</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
</tr>
<tr>
<td></td>
<td>xxx</td>
<td>xxx</td>
<td>xxx</td>
</tr>
<tr>
<td></td>
<td>Q</td>
<td>Q</td>
<td>Q</td>
</tr>
<tr>
<td>Array label</td>
<td>R(1,J)</td>
<td>R(2,J)</td>
<td>R(3,J)</td>
</tr>
</tbody>
</table>

The Q associated with A(1) and A(4) indicates these attributes have fractional values and that there are Q digits in the fractional part of the value. Note that word 1 conforms to the required specification.

The coding for parts 308 and 309 for this format follow:

308.10 Do step 308.1 + M(7)/100.
308.105 Done.
308.11 Set A(1) = G(5,0,Q,R(2,J)).
308.12 Set A(2) = 1000 + fp(R(1,J)/1000).
308.13 Set A(3) = ip(R(2,J)/10^Q).
308.14 Set A(4) = G(5,0,Q,R(3,J)).
308.15 Set A(5) = G(2,7,0,R(3,J)).
308.16 Set A(6) = G(2,5,0,F(3,J)).

309.10 Do step 309.1 + M(7)/100.
309.105 Done.
309.11 Set R(2,J) = A(1) \cdot 10^Q.
309.12 Set R(1,J) = A(2).
309.13 Set R(2,J) = U(4,5,0,R(2,J),A(3)) if A(3) \neq 0.
309.14 Set R(3,J) = A(4) \cdot 10^Q.
309.15 Set R(3,J) = R(3,J) + A(5) \cdot 10^7.
309.16 Set R(3,J) = R(3,J) + A(6) \cdot 10^5.
The general get and put formulas are used when the packing or unpacking expression might be complicated. The specific formulas used are slightly more efficient in terms of computer speed. The specific formulas are much simpler than would normally be the case because the words of \( R(\cdot, J) \) have a zero value before an entity is stored in them. A maximum of 250 rows of \( R \) can be used at any one time. When an entity is removed from \( R \), the words of storage are made available.

In the next example, it is assumed that the standard format is used for all sets except set 2. Set 2 uses the format described in the immediately preceding example. In this case, the coding is:

308.01 To step \([I=2: 308.1; 308.02]\).
308.02 Set \( A(M(7)) = [M(7) > 2: R(M(7), J); M(7) = 1: R(2, J); 1000 \cdot fp(R(1, J)/1000)] \).
308.03 Done.

Steps 308.10 to 308.16 as above

309.00 To step \([I = 2: 309.1; 309.01]\).
309.01 Set \( R(2, J) = A(1) \) if \( M(7) = 1 \).
309.02 Set \( R(1, J) = A(2) \) if \( M(7) = 2 \).
309.03 Set \( R(M(7), J) = A(M(7)) \) if \( M(7) > 2 \).
309.04 Done.

Steps 309.10 to 309.16 as above.

Within the framework of parts 308 and 309, the format structure for the \( R \) array may be made as complex as desired. The degree of complexity to be used involves a trade-off analysis between: (1) the storage requirements for \( R \) and the coding required for parts 308 and 309; and (2) the increased computational times when packing is employed.

**STORAGE OF ATTRIBUTES OF SETS**

Ten attributes for each set of a JASP program are automatically maintained in array \( F \). To conserve storage, only four words are used
to store all the attributes associated with a set. A graphical portrayal of the location of these attributes for one set is shown below:

<table>
<thead>
<tr>
<th>Word number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attribute Number</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Packing</td>
<td>xxx xxx xxx xxx xx x xx xxxxxxxxxx xxxxxxxxxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Array label</td>
<td>F(1,I)</td>
<td>F(2,I)</td>
<td>F(3,I)</td>
<td>F(4,I)</td>
</tr>
</tbody>
</table>

The attribute code numbers associated with a set represent:
1. Number of entities in the set, N(I);
2. Column number of R of last entity in the set, L(I);
3. Column number of R of first entity in the set, K(I);
4. Maximum number of entities ever in the set;
5. Number of attributes for an entity in the set;
6. Number of words of storage assigned to an entity of the set;
7. The procedure (described below) for ranking entities in the set;
8. The attribute to be used for ranking entities in the set;
9. Time integrated number of entities in the set;
10. Time of last change in the number of entities in the set.

<table>
<thead>
<tr>
<th>Code</th>
<th>Ranking Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A low value is given priority, or Low Value First (LVF). For entities with the same value, the entity that was in the set first is given priority (LVF-FIFO).</td>
</tr>
<tr>
<td>2</td>
<td>A high value is given priority, or High Value First (HVF). For entities with the same value, the entity that was in the set last is given priority (HVF-LIFO).</td>
</tr>
<tr>
<td>3</td>
<td>First-In, First-Out, (FIFO).</td>
</tr>
<tr>
<td>4</td>
<td>Last-In, First-Out (LIPO).</td>
</tr>
<tr>
<td>5</td>
<td>LVF-LIFO</td>
</tr>
<tr>
<td>6</td>
<td>HVF-FIFO</td>
</tr>
</tbody>
</table>

As evidenced by the above, there are the following limitations on the attributes of sets:

- Number of entities/set ≤ 999
- Number of attributes/entity ≤ 99
- Number of words/entity ≤ 9
- Number of ranking procedures ≤ 9
- Attribute number for ranking ≤ 99

STATISTICAL STORAGE ARRAYS

The formats for the arrays C, H and D are shown on the following page.

The manner in which the arrays are packed, imposes the following limitations.

X(I) ≤ 9999 for all I
Number of observations on variable X(I) ≤ 9999 for all I
Number of observations on X(I) in a single cell of a histogram ≤ 999 for all I.
**Table 1**

FORMATS FOR ARRAYS C, H AND D

### ARRAY C

<table>
<thead>
<tr>
<th>Array element</th>
<th>C(1,I)</th>
<th>C(2,I)</th>
<th>C(3,I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity stored</td>
<td>$\Sigma X(I)$</td>
<td>$\Sigma (X(I) \cdot X(I))$</td>
<td>MAX X(I)</td>
</tr>
<tr>
<td>Packing</td>
<td>Full word</td>
<td>Full word</td>
<td>XXXX</td>
</tr>
</tbody>
</table>

### ARRAY H

<table>
<thead>
<tr>
<th>Array element</th>
<th>H(I,1)</th>
<th>H(I,2)</th>
<th>H(I,3)</th>
<th>H(I,4)</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity stored</td>
<td>Not used</td>
<td>LL(2)*</td>
<td>WID†</td>
<td>No. of cells</td>
<td>Obs./cell</td>
</tr>
<tr>
<td>Packing</td>
<td>X</td>
<td>XX.X</td>
<td>XX.X</td>
<td>XX</td>
<td>XXX XXX XXX</td>
</tr>
</tbody>
</table>

| Cell No. | (1) | (2) | (3) | (4) | (5) | (6) | (7) | (8) | (9) |

### ARRAY D

<table>
<thead>
<tr>
<th>Array element</th>
<th>D(1,I)</th>
<th>D(2,I)</th>
<th>D(3,I)</th>
<th>D(4,I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity stored</td>
<td>$\Sigma Y(I) \cdot \Delta t$</td>
<td>$\Sigma Y(I) \cdot Y(I) \cdot \Delta t$</td>
<td>MAX Y(I)</td>
<td>Last change time</td>
</tr>
<tr>
<td>Packing</td>
<td>Full word</td>
<td>Full word</td>
<td>Full word</td>
<td>Full word</td>
</tr>
</tbody>
</table>

*LL(2) is the lower limit of cell 2.
†WID is the width of each cell. Decimal point is implied.
§The value of $\Delta t$ is T-D(4,I).
IV. JASP STATEMENTS

The link between JASP and the user consists of statements that refer to those JASP parts which support the user in building a simulation program. It is JASP's objective to reduce the amount of coding required of the user by providing routines common to many problems. In JASP, parts have been coded to perform the following functions:

1. Initialization of JASP variables;
2. Time and event control;
3. Information storage and retrieval;
4. Performance data collection;
5. Summary, monitor and error reporting; and

All but number 2 can be considered support functions. Event and time control is more of a submain program than a support function. In most applications, event and time control are used to move the program from its start to its completion. This is illustrated in Figure 3, which shows the communication pattern between the user and JASP parts.

In terms of Figure 3, the user must write the main program, the user event selection part and the user event parts. In the main program, the user initializes those variables peculiar to his problem. The user employs all lower case letters for these problem-oriented variables.

The next step is for the user to begin the JASP initialization procedure (part 2). The code numbers for the sets, performance variables and parameter sets are established during this initialization procedure. After it is completed, the user calls the event and time control part (part 1) of JASP, which is the executive routine for the entire
Fig. 3 -- User and JASP Part Interaction
simulation. As shown in Figure 3, the executive routine calls part 30, which the user must write to relate event codes to the part numbers associated with the coding that simulate the happenings when events occur. These user event parts can have part numbers from 31 to 299, and can use any JASP statement to facilitate the simulation of events.

The main program and the user event selection part normally involve a small amount of coding that is similar for most applications. The large task for the user is the writing of the parts simulating the events that can occur. This puts the emphasis where it belongs—on the special characteristics of the system under study, and on how they can change at specific points in time.

Figure 4 further illustrates the supervisory role of the time and event control part. In this figure, it is seen that part 1, time and event control indicated by the shaded area, uses the other JASP parts to perform the functions required in time and event control.

The first step in time and event control is to remove the first, or next, event from the event set. The time variable, T, and event code, E, are assigned the values associated with this next event. A test is then made to determine if it is a user event (code < 300) or a control event (code ≥ 300). If it is the former, the user's event selection part is called, and the event is simulated by transferring to the user's event simulation code. Following simulation of the event, a check is made to determine whether a trace of events is being made. If it is, a printout of appropriate event information is made. Next the executive starts the above procedure by removing the next event.
Fig. 4 -- JASP Time and Event Control: Part Interaction
When a control event occurs, one of three things can happen:
(1) the simulation is completed; (2) a monitoring of the simulation is
desired; or (3) the standard JASP summary report is to be printed.
These tasks will be described in this section in detail at the time
when the JASP statements are presented which cause these functions to
occur.

Every JASP statement involves the JOSS statement, "Do part ____"
where the part number signifies a JASP function. JASP statements may
require the definition of one or more JASP variables. A listing
of the part numbers, JASP functions, and input requirements
is given in Figure 5. Each part will be referred to as a JASP
statement, and each will be described in the following
paragraphs.

INITIALIZATION

Initialization for a JASP program is accomplished through the
use of part 2 and supporting parts of JASP, as illustrated in Figure
6. The initialization is performed in an interactive mode; requests
for variable values are demanded of the user at the JOSS console.
Example 1 of the next section (Figure 8) illustrates the printout of the
interaction between JOSS and the user.

First, the heading "Initialization Procedure" is printed. Next,
the simulation start time, and the number of digits, Q, in the frac-
tional part (fp) of time are requested. If time is not packed as an
attribute of an event, the answer to the request "The number of digits in
fp of time (e.g., XXX.XX is 2) = " should be J. The next request is for the
<table>
<thead>
<tr>
<th>Part</th>
<th>Function</th>
<th>Input Variables Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Time and event control</td>
<td>none</td>
</tr>
<tr>
<td>2</td>
<td>Data initialization</td>
<td>as demanded</td>
</tr>
<tr>
<td>3</td>
<td>Filing an entry</td>
<td>$A(\cdot)$, $I$</td>
</tr>
<tr>
<td>4</td>
<td>Removing an entry</td>
<td>$I, J$</td>
</tr>
<tr>
<td>5</td>
<td>Search a set for an entry, $J$, that meets specified conditions</td>
<td>$I, M(1)$</td>
</tr>
<tr>
<td>6</td>
<td>Collection of statistics on static variables</td>
<td>$I, X(I)$</td>
</tr>
<tr>
<td>7</td>
<td>Collection of statistics on dynamic (time persistent) variables</td>
<td>$I, Y(I)$</td>
</tr>
<tr>
<td>8</td>
<td>Histogram</td>
<td>$I, X(I)$</td>
</tr>
<tr>
<td>9</td>
<td>Summary report</td>
<td>none</td>
</tr>
<tr>
<td>10</td>
<td>Error report</td>
<td>$M(1)$</td>
</tr>
<tr>
<td>11</td>
<td>Monitor</td>
<td>none</td>
</tr>
<tr>
<td>12</td>
<td>Print set I</td>
<td>$I$</td>
</tr>
<tr>
<td>13</td>
<td>Print all sets</td>
<td>none</td>
</tr>
<tr>
<td>14</td>
<td>Retrieve $A(\cdot)$</td>
<td>$I, J$</td>
</tr>
<tr>
<td>15</td>
<td>Obtaining random number, $Z$</td>
<td>none</td>
</tr>
<tr>
<td>16</td>
<td>Obtaining a uniform deviate, $V(I)$</td>
<td>$I$</td>
</tr>
<tr>
<td>17</td>
<td>Obtaining a normal deviate, $V(I)$</td>
<td>$I$</td>
</tr>
<tr>
<td>18</td>
<td>Obtaining a lognormal deviate, $V(I)$</td>
<td>$I$</td>
</tr>
<tr>
<td>19</td>
<td>Obtaining an Erlang deviate, $V(I)$</td>
<td>$I$</td>
</tr>
</tbody>
</table>

Fig. 5 -- Identification of Part Numbers for Performing Basic JASP Functions
Fig. 6 -- JASP Initialization Procedure: Part Interaction
maximum number of attributes and words associated with an event to be stored in the event set. Next, the maximum number of sets ever used, including the event set, is requested.

The next portion of the initialization procedure deals with attributes of sets. The attributes associated with set number 1, the event set, are automatically established by JASP as LVF ranked on time (attribute 1). For all other sets, the priority code number (1 = LVF-FIFO; 2 = HVF-LIFO; 3 = FIFO; 4 = LIFO; 5 = LVF-LIFO; 6 = HVF-FIFO) and the attribute number of the ranking attribute, and the number of attributes and words (of R) per entity must be defined. The priority code establishes the ranking procedure for the set. Any attribute can serve as the ranking attribute for a set. If ranking is done on a FIFO or LIFO basis, no ranking attribute is required. A message to this effect is typed during the initialization procedure.

The next section of the initialization procedure involves the initialization of the arrays for storing performance measure information. The maximum number of variables collected in part 6, is requested, i.e., the number of variables defined in terms of X(I). The JASP variable X(I) is defined by the user and used to collect performance data. A request is made to determine if a histogram for each of these variables is desired. If it is, "1" should be typed in response to the demand for C(3,1). The lower limit of cell 2, the width of each cell, and the number of cells including the end cells for the histogram are requested for each histogram. Sequential requests are then made for 1) the maximum number of variables collected in part 7 (those defined in terms of Y(I), a dynamic performance
variable); 2) the number of sets of parameters to describe distribution functions from which samples (deviates) are desired; and 3) parameter values for each parameter set. These last values are stored in the Array B. The definitions of the values of B depend on their use in the simulation. (Their definition when used as parameters for distribution functions will be described later.)

A random number seed is then requested. The program will set the random number seed equal to 0.47594118 if a 0 is inserted as the starting random number seed.

The next section of the initialization procedure deals with the insertion of set entities and events. The word entity is used to indicate a member of a set, whether it be an entity or an event. Set number and attribute values must be specified for each initial entity. To conclude the insertion of initial entities, a 0 is typed as the response to the demand "Entry is for set =". The number of attribute values for an entity demanded is equal to the number established for the set specified. If it is desired to stop inserting attribute values before reaching this number, a value less than or equal to \(-10^5\) should be given as an attribute value. This causes all remaining attribute values to be set equal to 0. When all attribute values have been inserted, a check is made to determine if the user is satisfied with his inputs. If he is, he should respond with a 1 to the request "OK to file the above".

When all initial entities are filed, the initialization procedure requests information concerning the use of the random deviate generators. Those not used are deleted from core storage. If the lognormal generator is used, the user is given the opportunity to convert the
parameters of the lognormal to those parameters of the normal required for the lognormal generator.

All statements and values inserted during the initialization procedure are filed as item 22 (initd). A line is typed to indicate this operation and the step number that performs the operation. If item 22 is currently being used, the user should make appropriate changes to step 334.3. Thus, item 22 (initd) contains the main portion of the user's program; those parts of JASP required by the user's program; and all initial values.

In some cases, only part of the initialization procedure will be appropriate. Then, the individual parts involved in the initialization procedure can be used separately and the initialization data updated using the available JASP parts. The functions associated with each part are listed in Appendix C.

In Appendix E under item 16 (datan), the JOSS steps involved in the data initialization function are listed. The statements required to perform the complete data initialization procedure are:

Recall item 16 (datan).
Do part 2.

**TIME AND EVENT CONTROL**

Part 1 accomplishes time and event control. The first event in the event set (the next event) is removed and the event code, E, and current time, T, are established. If the event code is less than 300, JASP assumes that the next event is user-written and transfers to the appropriate user-written part through a call to part 30 which must be user-written. Part 30 must identify the part number of the user-written event corresponding to the event code. (Part 30 is an events
list that relates event codes to the part numbers, where the user has
written the program to simulate the event.)

If the event code is greater or equal to 300, a monitoring event
or the end of "time and event control" has occurred; the latter takes
place if the event code is 999. (An event with code 999 is inserted into
the event set whenever the user calls directly for part 9, the
printing of the final summary report.) The JASP user can insert
an event coded 999 at will to end part 1 time and event control. The
section dealing with monitor routines describes the options available
for monitoring events.

After return to the time and event control routine, the next
event is removed and the process described above is repeated. Time
and event control are initiated by the statements:

Recall item 17 (JASP).
Do part 1.

The first statement is included in the previously described initialization
procedure. If initialization (part 2) has been performed,
the statement "Do part 1" is sufficient. Because part 2 deletes from
core storage those JASP parts not required for the particular simulation,
recalling item 17 (JASP) can result in the use of additional core storage.

To start a simulation using data previously inserted into item
22 (initd), the calling sequence is:

Recall item 22 (initd).
Do part 1.

INFORMATION STORAGE AND RETRIEVAL

Filing an Entity into Set I

The attributes of the entity must be established in the attribute
buffer A(·). To store the vector A in the first available column of
the file R use the statement:

Do part 3 for I = ___.

Part 3 locates a column of R for storage of the new entity and,
if part 309 has not been modified, inserts the attribute values in
the standard format. If part 309 has been modified, the attributes are
stored to comply with the user's format, as specified in part 309.
JASP establishes the pointers (predecessor and successor column numbers)
of the entity in set I in accordance with the priority ranking code and
the ranking attribute specified in the initialization procedure by the
user. After filing, the number of entities in set I is increased
by one; other attributes associated with set I, such as first or last
entity in set I, are updated if necessary.

Removing the First Entity from Set I

Part 4 is called with J = 0. The statements involved are:

Set J = 0.
Do part 4 for I = ___.

When an entity is removed from a set, JASP updates the set's remaining
entities to ensure that they are in the proper order. The attributes
of the entity are inserted into the attribute buffer A(·). The core
storage previously used for the entity is made available for general
use. The number of entities and the first entity for set I are then
updated.

Removing an Entity Stored in Column J from Set I

The following statements are required:
Set $J = ____$.  
Do part 4 for $I = ____$.

It is the user's responsibility to ensure that column $J$ is indeed a member of set $I$. As in removing the first entity, JASP updates the set attributes and inserts the attributes from column $J$ in the buffer vector, $A$. Because the column number is given it may appear that the set number is redundant for this operation. This is not the case because: (1) the number of attributes for the entity stored in the $1^{st}$ column of $F$ must be determined; and (2) the attributes of set $I$ must be updated.

Retrieval of the Attributes of an Entity without Removing it from a Set

The following statements are used:

Set $J = ____$.  
Do part 14 for $I = ____$.

The statements assign the attributes stored in column $J$ to the buffer array, $A$. The set number $I$ is also required (as it is when a specific entity is removed from a set) because a different number of attributes is allowed with each set. The filing system is not altered when retrieving the attributes of an entity; only the attribute buffer is changed by this operation.

Finding an Entity in a Set

To find an entity in set $I$ located in column $J$ that meets specified conditions, the user must establish the conditions in part $M(1)$, where $M(1)$ is a user part number. (User part numbers are from 31 to 299.) In part $M(1)$, the variable $M(2)$ should be assigned the value 0 when the desired conditions are met. When JASP calls part $M(1)$, the buffer array $A$ contains the attribute values of column $J$ to test against
the desired conditions. When the conditions are met (i.e., part
M(1) has set M(2) = 0), testing is stopped, and the variable J is
the column number of the set I entity that met the conditions. If
no entity meets the conditions, J is given a value of zero. Entities
in a set are always tested, beginning with the first entity in the
set. The statements are:

   Set M(1) = ___.
   Do part 5 for I = ____.

The following example illustrates JASP's "find" or "locate" feature.
Suppose it is desired to find the first entity in set 2 for which the
value of attribute 1 is greater than 10, and the value of attribute
2 is less than 5. First, the part number M(1) that specifies the
desired conditions is arbitrarily established, say as part 35. The
coding required to determine the column number is:

   Set M(1) = 35.
   Do part 5 for I = 2.

   35.1 To step [A(1) > 10:35.2; 35.4].
   35.2 Done if A(2) ≤ 5.
   35.3 Set M(2) = 0.
   35.4 Done.

Note that M(2) need not be initialized to a value other than 0,
as this is done in part 5.

Finding the Entity with the Largest (Smallest) Value

The method for finding the entity with the largest value of
attribute 3 in set 4 does not conform exactly to the specifications
for the find routine discussed in the last paragraph. If in part
M(1), M(2) is never set to 0, part 5 can be used to obtain all the
attribute values of successive entities in the set. Part M(1) can
then be used to identify the column number containing the largest
(smallest) value. Letting \( e \) be the column containing the largest value of attribute 2, and \( v \) the largest value of attribute 3, the coding is:

Set \( e = 0 \).
Set \( v = -10^5 \).
Set \( M(1) = 36 \).
Do part 5 for \( I = 4 \).

36.1 Done if \( A(3) \leq v \).
36.2 Set \( v = A(3) \).
36.3 Set \( e = J \).

In this example, the lower case symbols \( e \) and \( v \), being lower case JOSS variables, are used because non-JASP variables are involved. For this example, any non-JASP symbols could be used. Also, note that the column being tested as a candidate is assigned the symbol \( J \), and is accessible to the user. If the entity with the smallest value is desired, the initial value of \( v \) should be set to a large value, and a \( \geq \) sign used in statement 36.1.

Obtaining Computed Attributes of Sets

If a function of the attributes of a set's entities is desired, part 5 can be used again. If it is, the part number defining the calculation must be defined and made accessible to part 5, and the initial value of the variable to be computed must be assigned. For sums, it will normally be zero; for products, it will normally be one. The coding to obtain computed attributes of sets is:

Set \( v = \begin{cases} 0 \text{ if a sum is to be computed} \\ 1 \text{ if a product is to be computed} \end{cases} \)
Set \( M(1) = \) _____.
Do part 5 for \( I = \) ____.
The following examples illustrate the coding used to compute the attributes associated with all entities in a set.

**Example 1.** Calculate the sum of the values of attribute 3 of the entities in set 2.

Set \( v = 0 \).
Set \( M(1) = 37 \).
Do part 5 for \( I = 2 \).

37. Set \( v = v + A(3) \).

**Example 2.** Calculate the product of the values of attribute 2 of the entities in set 4.

Set \( v = 1 \).
Set \( M(1) = 38 \).
Do part 5 for \( I = 4 \).

38. Set \( v = v \cdot A(2) \).

**Example 3.** Each entity in set 3 represents an item in inventory at warehouse 3. Attribute 3 represents the number of units of inventory on-hand for one item and attribute 4 represents the unit cost of the item. Calculate the dollars tied up in inventory at warehouse 3.

Set \( v = 0 \).
Set \( M(1) = 39 \).
Do part 5 for \( I = 3 \).

39. Set \( v = v + A(3) \cdot A(4) \).

**Example 4.** Same as example 3 with the condition that only items with a value greater than or equal to $1000.00 should be considered. The only change required is in statement 39, which is modified to:

39. Set \( v = v + A(3) \cdot A(4) \) if \( A(4) \geq 1000 \).

**DATA COLLECTION**

Three parts are provided for collecting data during a simulation. Parts 6 and 7 are used to collect data for estimating the parameters of a distribution describing the performance measure variables associated with a given problem. The maximum value observed for each performance
measure is also maintained. Part 8 is used to classify values of a variable into given cells so that a histogram can be prepared to show the frequency with which the variable was within a given range.

Parts 6, 7 and 8 each collect data throughout the running of a JASP program at points specified by the user by his calls to parts 6 and 7. When a summary report is requested by a call to part 9, summary statistics are printed for each variable for which data is collected.

Part 6 is used to collect information on a variable that assumes a value at a particular observation point. Examples are the waiting time of a customer in a queueing situation or the test score of a student. These will be referred to as "static variables." The word "static" indicates only that the variable value is of interest at a point in time. Part 7 is used to collect statistics on variables that persist over a period of time. Examples are the number of customers in a system and the assets of a corporation throughout a year. These will be referred to as "dynamic variables."

Collection of Statistics on Static Variables

The user must assign a code number to each static variable for which statistics are desired from the JASP program. The code identifies the subscript of the variable, X, as well as the value of the static variable to be included in the statistical calculation. The calling statement for adding the value X(I) to the data previously collected on variable I is:

$$\text{Do part 6 for } I = \_\_\_. $$

As an illustration of the use of this statement, consider that the user identifies the time spent in a system as static variable number
2. If the time spent in the system was 20 hours then the statements

\[ X(2) = 20. \]
Do part 6 for \( I = 2. \)

would include this one value with all other values observed for this variable.

A histogram based on the values of \( X(I) \) will be constructed for variable \( I \) if the response during initialization indicated that a histogram was desired. The user needs no programming statement to obtain a histogram.

**Collection of Statistics on Dynamic Variables**

The collection of statistics on dynamic variables requires the time integration of the variable over the time period for which its value has not changed. The user must assign a code number, \( I, \) to each dynamic variable; and the value of the variable must be stored in the JASP variable \( Y(I) \). The values of the dynamic variable over the time for which it has maintained its current value may be integrated by calling:

Do part 7 for \( I = ____ \).

The user is responsible for the collection of data on dynamic variables immediately before he changes the variable's value.

For example, if the user defines dynamic variable number 2 as the number of customers in a system, then when a new customer arrives, statistics for this variable must be collected by using the statement "Do part 7 for \( I = 2 \)," before \( Y(2) \) is incremented. The value of \( Y(2) \) is assumed to be equal to the current number in the system when part 7 is called. A graphical illustration of a dynamic variable is shown below.
Because the code numbers for static variables and dynamic variables are independent, the same code number can be assigned to both.

**SUMMARY REPORTING**

The basic output from JASP is a summary report obtained from part 9. Part 9 performs a core storage management function, and calls the part that types a statistical summary for the variables included in the program. The time integration for dynamic variables collected in part 7 is updated to the point at which the summary report is requested.

In addition to the statistical estimates for the program's defined variables, the summary report provides a print-out of the statistical estimates associated with each set used in the program, including the average number and maximum number of entities in a set; and the number in the set at the time the summary report is requested. Also typed are the entities in the set when the summary report was requested; as well as information on the problem being solved, such as the parameter set values (if any), the random number seed; the
start time of the simulation, and time at which the summary report
was requested. No information on the name of the user nor his project
number is provided; these are provided at the top of each JOSS output
sheet.

In addition to the information regarding the problem being solved,
the JASP summary report prints the value of the timer, which is reset
after the initialization procedure. This represents the amount
of time involved in the simulation, not including initialization or
final reporting; and if the simulation time period is increased, the
user can estimate the computer time required for the simulation.
A sample of the JASP summary report is given in Example 1 (Figure 10)
of the next section.

Several summary report features regarding histogram print-out
should be noted. There will be twelve cells per line for a histogram.
If a histogram involves more than twelve cells, the additional cells
will be on separate lines. New heading information is not provided for
those cells greater than twelve. If a histogram involves a number of
cells which is not a multiple of twelve, the unused cells will have a
value of -1 inserted into them to so indicate.

The statement for obtaining a summary report is:

Do part 9.

To augment the summary report with additional output, statements can
be inserted directly into part 329. Because JOSS automatically integrates
all steps with the same part number, the user can insert the augmented
output wherever he chooses by referencing part 329 in the statements
inserted anywhere within the user's program. Whenever the user
calls for a summary report directly an end of simulation event (code 999)
is scheduled; this terminates the simulation run after the summary report is printed. Summary reports obtained to monitor the simulation do not schedule an end of a simulation event. These are described in the next paragraph.

**MONITORING EVENTS AND THE STATUS OF SETS**

JASP provides the capability to (1) monitor each event as it occurs between any two points in time; (2) type a set at a particular point in time; (3) type all sets at a particular point in time; and (4) obtain a summary report at a particular point in time. Part 11 is the basic monitoring routine of JASP, and is called whenever an event code with a value greater than or equal to 300 but not 999 occurs. By filing events in the event set with event codes greater than or equal to 300, monitoring can occur as specified by the user. Events can be established during the initialization procedure or during on-line operation. As with other events, the time for the monitor event is established as attribute 1, A(1), and the event code as attribute 2, A(2).

An event code of 300 initiates the typing of the current time and the current event, and the attributes of the next event stored in the event set. This print-out will continue until another event with code 300 occurs. Thus, the user should establish pair events of type 300; the first event would initiate the typing of events; the second would stop the typing of events.

An event code of 301 will cause the event set to be typed at the time the event occurs. The event set is typed by part 12. Part 12 can be used for typing any set I. The statements to obtain a set I print-out are:
Recall item 20 (rpts).
Do part 12 for I = ____.

An event coded 302 and scheduled to occur at a particular point in time will call part 13 and cause each set to be typed. If an event coded 303 is scheduled to occur at a particular point in time it will cause a summary report to be typed by performing a portion of part 9.

Events coded 300, 301, 302, and 303 do not affect the storage arrays associated with the JASP variables. They are reporting type events and do not change the status of the system or the variables describing the system.

JASP users may employ parts 12 and 13 without going through the event routine. These are particularly useful in an interactive mode, and when the user desires print-outs of particular sets. When accessed directly from the JOSS console, part 11 should be used to accomplish the desired function. The statements required are:

Recall item 20 (rpts).
Set E = ____.
Do part 11.

where E = 300, 301, 302 or 303. Part 11 will delete all parts except itself that are used for monitoring.

ERROR REPORTING

Whenever an unexpected condition occurs within JASP, part 10 is called and the step number where the error occurred is specified by M(l). Part 10 types the following information:
1. The step number where the error occurred.

2. The entities currently in all sets.

3. The array $F$ which contains the attributes of all sets.

4. The array $C$ which contains the statistical quantities collected for all variables defined as $X(I)$.

5. The array $D$ which contains the information regarding all variables defined as $Y(I)$.

After printing this information JASP prints the message "Error report completed" and then types "Your move" to demand that the user perform the next move or operation. The user also has access to the JASP error reporting mechanism. For example, if the variable $v$'s legitimate values were 1, 2, or 3 and in step 50.5, $v$ were tested for these values, the user could include an error check as follows:

50.5 To step \([v=1: 50.81; v=2: 50.82; v=3: 50.83; 50.51]\).
50.51 Do part 10 for $M(1) = 50.5$.

**OBTAINING RANDOM NUMBERS AND RANDOM DEVIATES**

The JASP variable $Z$ is used to store the current value of the random number. To obtain a new random number, the following statement is used:

```
Do part 15.
```

In part 15, the value of $Z$ is updated to a new random value. The random numbers are in the interval $(0,1)$ and have as many as seven digits. Currently, part 15 is written to obtain one random number stream. To obtain more than one random number stream, $Z$ must be a dimensioned variable, and the user would have to provide initial seed values in the initialization procedure for each random number stream. When part 15 is called it would be necessary to indicate the random number stream to be used.
JASP provides statements for obtaining random deviates from a uniform distribution; a normal distribution; a lognormal distribution; and an Erlang distribution. Deviates generated by the latter three distributions, have statements written to set the deviate value equal to an extreme value, provided the extreme value is exceeded by the deviate. This is not sampling from a truncated distribution; it corresponds to sampling from a mixed distribution, where there exists a finite probability that samples will be obtained at the extreme points specified by the user. (To sample from truncated distributions, part 327 should be rewritten to reject deviate values which do not fall within the specified limits.) In all four distributions, the user must specify the parameter set number, I, to indicate the parameters associated with the distribution for which he desires a deviate. In each case JASP returns V(I) as the deviate requested.

Obtaining a Deviate from a Uniform Distribution

The following statement will obtain a deviate from a uniform distribution with parameters as specified by parameter set I:

    Do part 16 for I ___.

When parameter set I is used to obtain a samples from the uniform distribution, the following are the definitions for the parameters stored in array B, column I:

    B(2,I) = lower limit of distribution; and
    B(3,I) = upper limit of distribution.

Obtaining a Deviate from a Normal Distribution

The following statement will obtain a deviate from a normal distribution with parameters as stored in parameter set I:
Do part 17 for $I = \ldots$.

The requirements for column $I$ of the array $B$ are:

- $B(1, I) =$ mean of the distribution;
- $B(2, I) =$ smallest value allowed for deviate;
- $B(3, I) =$ largest value permitted for deviate; and
- $B(4, I) =$ standard deviation associated with the distribution.

**Obtaining a Deviate from a Lognormal Distribution**

The following statement will obtain a deviate from a lognormal distribution with parameters as stored in parameter set $I$:

Do part 18 for $I = \ldots$.

The parameters stored in column $I$ of array $B$ are identical to those for the normal distribution; and the parameter values are for the normal distribution, which corresponds to the lognormal distribution for which a deviate is desired. If the parameters for the lognormal distribution are inserted in parameter set $I$ then, in the initialization procedure, a demand is made for the value of $I$ for which the transformation of the parameters should be made to the corresponding normal distribution.

**Obtaining a Deviate from an Erlang Distribution**

The following statement will obtain a deviate from an Erlang distribution with parameters as stored in parameter set $I$:

Do part 19 for $I = \ldots$.

A deviate from the $k$-Erlang distribution can be considered as the sum of $k$ deviates obtained from an exponential distribution. The parameter values associated with parameter set $I$, used for the Erlang distribution, are:
B(1,I) = the mean time associated with the corresponding exponential distribution
B(2,I) = smallest value of deviate allowable
B(3,I) = largest value of deviate allowable
B(4,I) = the number, k, of exponential deviates to be included in the sum that represents the Erlang deviate.

While deviates from the exponential distribution can be obtained by setting B(4,I) equal to 1, it is simpler to use the two statements:

Do part 15.
Set $y = -a \cdot \log (Z)$.

where $a$ is the mean time for the exponential distribution (the same as B(1,I)), and $y$ is a deviate drawn from the exponential distribution.
V. USER PROCEDURES AND EXAMPLES

JASP procedure is similar to that for using JOSS. Currently, all JASP parts are stored in file 714 (p8260) in items 16, 17, and 20-25 inclusive. Appendix D gives an item-part identification list.

The normal JASP procedural sequence is:

1. User's main program;
2. JASP initialization;
3. JASP time and event control.

The item containing the user's main program is recalled, and a direct command made to do the appropriate part. This part, in turn, has indirect calls to JASP initialization and time and event control. The examples will illustrate this coding.

In JASP initialization, the necessary root segments of JASP are added to the user's program and these, together with all initial data values are stored in item 22 (initd). By using item 22, initialization can be bypassed on future runs. JASP assumes that the user will maintain item control; and that item 22 is (or can be made) available during initialization.

Two examples are presented in this report. The first illustrates the use of JASP to solve a simple queueing problem, involving a single server processing one customer at a time. Customers who arrive while the server is busy, wait in a queue until they can be served.

In the simulation, statistics are collected on the time spent in the system by a customer and the idle time of the server.

*The initial coding for this problem took less than two hours.
The second example presents a simulation of a man-machine price-inventory situation that illustrates the interactive simulation capability available through JASP. Here, a decision maker is incorporated in the simulation; when a decision is to be made, JASP types a message requesting information from the decision maker. The latter in turn can request information from the computer by invoking specific options that cause the computer to display information. The following decisions are required in this particular simulation: (1) a price quote to a customer; (2) the time and amount of an order for additional items; and (3) the times at which status reviews should be made.

The examples are intended to illustrate the use of JASP and its associated procedures, not to solve a problem nor to present a man-machine simulation.

**Example 1 - A Single-Channel Queueing Situation**

A single channel queueing system consists of a server and customers seeking service. The server can process only one customer at a time; and other customers wait in a single queue for service. It is assumed that the processing time of the server and the time between the arrival of customers are exponentially distributed, with mean times of 6 and 10 respectively. At time zero, there is one customer in the system, who did not have to wait; the server will finish processing this customer at time 3.5. The first new customer to the system will arrive at time 0. It is desired to simulate this queueing situation for 400 time-units to obtain statistical estimates of the following variables:
1. The time a customer is in the system;
2. The time a customer is waiting in the queue;
3. The number of customers in the system; and
4. The busy time of the server.

Three events will be used to simulate the queueing system:

<table>
<thead>
<tr>
<th>Event Code</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The arrival of a customer;</td>
</tr>
<tr>
<td>2</td>
<td>The end of processing a customer; and</td>
</tr>
<tr>
<td>3</td>
<td>The end of the simulation.</td>
</tr>
</tbody>
</table>

As is typical with simulations, these events describe the system and the simulation's control points.

A queue set is needed in addition to the event set, to maintain the customers waiting in the server's queue. Set 2 will be the queue set; and it will be assumed that the customers are served on a first in, first out (FIFO) basis. The attributes required for each set are shown below:

Attributes of Entities

<table>
<thead>
<tr>
<th>Set</th>
<th>A(1)</th>
<th>A(2)</th>
<th>A(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Event</td>
<td>Event time*</td>
<td>Event code*</td>
<td>Customer's arrival time</td>
</tr>
<tr>
<td>2. Queue</td>
<td>Customer's arrival time</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

*Attribute definitions are fixed by JASP.*
Three attributes are associated with set 1, and one with set 2. The standard format for the filing array R will be used; parts 308 and 309 will not be rewritten. Three words are required for set 1 (recall A(2) is put in word 1); two words are required for set 2 (since A(1) is in word 2).

To obtain the performance measures specified in the problem statement, numeric code numbers for the JASP variables $X(\cdot)$ and $Y(\cdot)$ must be defined. The code, variable, and definition relationship selected is shown below.

<table>
<thead>
<tr>
<th>Code</th>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$X(1)$</td>
<td>Value of the time a customer spent in system</td>
</tr>
<tr>
<td>2</td>
<td>$X(2)$</td>
<td>Value of the time a customer spent in queue</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code</th>
<th>Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$Y(1)$</td>
<td>Number of customers in the system</td>
</tr>
<tr>
<td>2</td>
<td>$Y(2)$</td>
<td>Status of server: 0 if idle; 1 if busy.</td>
</tr>
</tbody>
</table>

These specifications are sufficient for the JASP user to write the non-JASP parts of the simulation program. (See Figure 7.) Flow charts of all events are found in Figure 8.

The JASP user must write a part 30 which transfers control to the proper part, based on the event code $E$. For this problem, parts 121, 131, and 141 were selected arbitrarily to represent the parts corresponding to event codes 1, 2 and 3. Thus, part 30 consists of the one step "Do part 111 + E * 10." as shown in Figure 7.

The user can select any part number between 30 and 300. Part 100 is selected for the main program and the coding is as shown in Figure 7.
The non-JASP variables, $a$ and $m$, are used respectively for the mean time between arrivals and mean processing time. The initial values for the performance variables are set in part 100. The initialization procedure is started by statements 100.5 and 100.6. Time and event control is initiated at step 100.9.

The coding of the event parts annotated in Figure 7 together with the flow charts of Figure 8, clearly explain the logic and statement types. Parts 121, 131 and 141 represent the arrival, end of service, and end of simulation events respectively. Parts 122, 123, 132, and 142 are used by these parts for convenience in isolating specific functions.

Figures 9 and 10 illustrate the initialization procedure and summary report for this example. The initialization procedure follows the discussion presented in Section IV. The summary report shows that it took 3.91 minutes to perform a simulation of 411.718 time units. The 11.718 time units above the 400 time units specified as the end of simulation time is required to process all customers who arrived prior to time 400. The statistics on $X(1)$ show that an average of over 15 time units was spent in the system for the 43 customers, even though the average service time was only 6.227 (15.175-8.948). One customer spent 46 time units in the system. The histogram for $X(1)$ shows that 14 customers spent less than 6 time units in the system and 9 spent more than 30 (6 + 12(2)) time units. Values for $Y(2)$ can be seen from the summary report.

From the line for variable $Y(1)$, we see that the average number in the system was 1.58, with a maximum of 7. The fraction of the time
the server is busy is 0.650 which is the average value for \( Y(2) \). Note that \( Y(2) = 1 \) when the server was busy, hence the time when \( Y(2) = 1 \) divided by the total time is the fraction of time that the server is busy. For this example only the printout for set 2 is of interest. Since set 2 was used for the queue of customers, the values shown represent the statistics for the server's queue.
Recall item 11 (sque).
Done.
Type all.

30.1 Do part 111+E*10.

100.1 Set a=10.
100.2 Set m=6.
100.3 Set Y(2)=1.
100.4 Set Y(1)=1.
100.5 Recall item 16 (datan).
100.6 Do part 2.
100.7 Set X(2)=0.
100.8 Do part 6 for I=2.
100.9 Do part 1.
100.95 Stop.

121.1 Do part 15.
121.3 Set A(1)=T-a*log(Z).
121.35 Do part 3 for I=1.
121.36 Do part 7 for I=1.
121.4 Set Y(1)=Y(1)+1.
121.6 To step 121.92 if Y(2)=0.
121.7 Delete A.
121.8 Set A(1)=T.
121.85 Let A be sparse.
121.9 Do part 3 for I=2.
121.91 Done.
121.92 Do part 122.

122.1 Do part 7 for I=2.
122.2 Set Y(2)=1.
122.3 Set A(3)=T.
122.4 Set X(2)=0.
122.5 Do part 123.

123.1 Do part 6 for I=2.
123.15 Do part 15.
123.4 Set A(1)=T-m*log(Z).
123.5 Set A(2)=2.
123.6 Do part 3 for I=1.

131.1 Set X(1)=T-A(3).
131.2 Do part 6 for I=1.
131.3 Do part 7 for I=1.
131.4 Set Y(1)=Y(1)-1.
131.5 To step 131.82 if N(2)=0.
131.6 Do part 7 for I=2.
131.7 Set Y(2)=0.
131.8 Done.
131.82 Do part 132.

132.1 Set J=0.
132.2 Do part 4 for I=2.

Events List

Mean interarrival time.
Mean service time.
Server status is busy.
1 customer in system.
Recall initialization segments.
Perform initialization.
Waiting time of customer in system at T = 0.
Collect statistics on X(2).
Begin time and event control.
Simulation ended.

Obtain a random number, Z.
Time of next arrival.
File next arrival event in set 1.
Integrate Y(1) to present time.
Increase no. in system for current arrival.
Process new arrival if server idle.
Delete buffer storage.
Time of arrival of customer is attribute 1.
Clear buffer except for A(1).
File new arrival in queue.
Arrival event completed.
Update system status since status of server is to change.

Integrate server status to present.
Server is now busy.
Arrival time of customer is now.
Waiting time of customer is zero.
Establish end of service event.

Collect statistics on customer waiting time.
Obtain a random number.
Time for end of service event.
End of service event code.
File in event set.

Compute time in system.
Collect statistics on time in system.
Integrate number in system to present.
Reduce number in system by 1.
Process next customer if one is in the queue.
Integrate status of server to present.
Change status of server to idle.
End of service event completed.
Process customer waiting.

Set up to remove first entry.
Move entry J from set 2.

Fig. 7 -- Program Listing for Queueing Simulation
132.3 Set X(2)=T-A(1).
132.35 Set A(3)=A(1).
132.4 Do part 123.

141.1 Do part 142.
141.2 To step 141.1 if N(1)=0.
141.3 Do part 9.

142.1 Set J=0.
142.2 Do part 4 for I=1.
142.3 Set T=A(1).
142.4 Do part 131 if A(2)=2.

Compute waiting time (statistics collected at 123.1).
Reset arrival time for storage in set 1.
Establish end of service event.

Clear system of next event.
Repeat above if events still exist.
Request for summary report.

Set up to remove first entry.
Remove entry J for set 1.
Update time.
Perform end of service if event is for end of service.

Type size.
size = 225

Fig. 7 — Program Listing for Queueing Simulation (con't.)
Fig. 8 -- Flow Charts of Events for Example 1
Use file 714 (p8260)
Roger.
Recall item 11 (sque).
Done
Do part 100.

Initialization Procedure

Simulation start time = 0
The no. of digits in fp of time(e.g. xxx.xx is 2) = 0
Max. No. of attributes for events = 3
No. of words of R to store these attributes = 3
Max no. of sets ever used = 2

Codes for sets are: 1=LVF, 2=HVF, 3=FIFO, 4=LIFO.

****** For Set 2
The max. no. of attributes for this set = 1
Max. no. of words to store these attributes = 2
Code for establishing priority = 3
No ranking attribute required for FIFO or LIFO.

Max. No. of variables collected in part 6 = 2
Set C(1,3)=0 or 1. A 1 implies that a histogram is desired for var. I.

\[ C(1,3) = 1 \]
Lower limit of cell 2 in interval [1,99.9] = 6
Width of each cell in interval [1,99.9] = 2
Number of cells (including end cells) = 14

\[ C(2,3) = 1 \]
Lower limit of cell 2 in interval [1,99.9] = .1
Width of each cell in interval [1,99.9] = .1
Number of cells (including end cells) = 2

Max. No. of variables collected in part 7 = 2
Max. no. of parameter sets used = 0
Random number seed (a 0 sets seed to .47594118) = 0

Initial Set Entries (Entities). A 0 indicates no more sets.
A value ≤ 10^5 indicates no more attributes for this entry.

Entry is for set = 1
\[ A(1) = 0 \]
\[ A(2) = 1 \]
\[ A(3) = 0 \]
Respond with a 1 if OK to file the above = 1

Entry is for set = 1
\[ A(1) = 3.5 \]
\[ A(2) = 2 \]
\[ A(3) = 0 \]
Respond with a 1 if OK to file the above = 1

Entry is for set = 1
\[ A(1) = 400 \]
\[ A(2) = 3 \]
\[ A(3) = 0 \]

*Note: Underlining indicates the user's inputs which is normally green.

Fig. 9 -- Initialization Procedure for Queueing Simulation
Respond with a 1 if OK to file the above = 1*

Entry is for set = 0
Type 1 if JASP deviate generators are used = 0

Step 334.3 files all as item 22 (initd).

Error at step 334.3: Please discard the item or use a new item number.
Discard item 22 (initd).
Done.
Go.

* Underlining indicates the user's inputs which are normally green.

Fig. 9 -- Initialization Procedure for Queueing Simulation (con't.)
** SUMMARY REPORT **

timer = 3.91

RANDOM NUMBER SEED = .47594118
START TIME = .000
SUMMARY REPORT PRINTED AT 411.718

** STATISTICS COLLECTED IN PART 5 **

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>NO. OBS.</th>
<th>AVERAGE</th>
<th>STD. DEV.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X( 1)</td>
<td>43</td>
<td>1.517508</td>
<td>1.2537860</td>
<td>4.600000</td>
</tr>
<tr>
<td>X( 2)</td>
<td>43</td>
<td>8.947781</td>
<td>1.1006705</td>
<td>3.200000</td>
</tr>
</tbody>
</table>

** HISTOGRAMS **

<table>
<thead>
<tr>
<th>VAR.</th>
<th>LL(2) WID</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>.8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>X( 1)</td>
<td>6.0 2.0</td>
<td>14</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>9</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>X( 2)</td>
<td>.1 1.0</td>
<td>13</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

** STATISTICS COLLECTED IN PART 7 **

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>END TIME</th>
<th>AVERAGE</th>
<th>STD. DEV.</th>
<th>MAX VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y( 1)</td>
<td>4.117 02</td>
<td>1.584891</td>
<td>1.7020481</td>
<td>7.000000</td>
</tr>
<tr>
<td>Y( 2)</td>
<td>4.117 02</td>
<td>6.503812</td>
<td>4.7684957</td>
<td>1.000000</td>
</tr>
</tbody>
</table>

** PRINTOUT FOR SET 1 **

AVERAGE NUMBER IN SET WAS 1.6219
MAXIMUM NUMBER IN SET WAS 3
CURRENT NUMBER IN SET IS 0

** PRINTOUT FOR SET 2 **

AVERAGE NUMBER IN SET WAS .9345
MAXIMUM NUMBER IN SET WAS 6
CURRENT NUMBER IN SET IS 0

timer = 6.02

Fig. 10 -- Summary Report for Queueing Simulation
Example 2 - A Man-Machine Price-Inventory Simulation

The description for this example is contained within the simulation program. The following dialogue will ensure that the program is ready to run:

```
Use file 714 (p8260).*
Roger.
Recall item 4 (inall).
Done.
File all as item 22 (initd).
Done.
Delete all.
Recall item 3 (invmm).
Done.
Do part 50.
```

The output from the JASP simulation is as follows:

**MAN-MACHINE INVENTORY SYSTEM**

At this time the following alternatives are available:

1. Obtain a description of the inventory situation;
2. Start the clock to begin the processing of customers;
3. Insert new parameter values and then start clock.

Your choice = 1**

**A MAN-MACHINE PRICE-INVENTORY SIMULATION**

In this simulation you are the distributor of a commodity. Customers request either 1, 2, 3, or 4 units of your commodity. The price of a unit is set by you. You quote only one price to each customer. He will accept or reject your quote. Inventory is adjusted accordingly and you can place an order for more units. You also can schedule review times at which orders can be placed or cancelled. Analysis of the inventory can be made at review, demand and order receipt times.

*Underlining indicates the user's inputs which are normally green.
**Ibid.
Your performance is measured in terms of profit per day. Your revenues are from sales, and your costs are for purchase, carrying charges, ordering and transportation and review. The specific cost characteristics are:

Purchase price of a unit to you 20.00
Inventory carrying charge for 1 unit for 1 day .020
Cost of ordering and transporting a unit 3.00
Cost for making an order (incurred even if order is cancelled) 7.00
Cost of performing a review 2.00
Cost if a review is cancelled 1.00

There are no explicit costs for lost sales. However as the number of customers rejecting your price quote increases the demand for your commodity decreases.

The average time between customers 1.00
The ave. number of units requested/cust. 1.8
The average time between the placement and receipt of an order 10.00

An analysis can be performed at each event time.

Analysis can be suspended until a time you specify by responding with a 0 to the demand for analysis decision.
The analysis options available to you are:

Option List

0. For listing of options.
1. Data values desired.
2. No action really desired.
3. Schedule a review or list of review times.
4. Place an order for units or list of incoming orders.
5. Cancel a review.
6. Cancel an order.
7. Obtain number of units due-in in k days.
8. Desire to go out of business.
MAN-MACHINE INVENTORY SYSTEM

At this time the following alternatives are available:

1. Obtain a description of the inventory situation;
2. Start the clock to begin the processing of customers;
3. Insert new parameter values and then start clock.

Your choice = 3
Mean time between arrivals of customers = 1
Purchase price of an item for the distributor = 20
Inventory carrying charge for 1 unit for 1 day = 0.02
Cost of ordering and transporting a unit = 3
Cost for making an order (indep. of size) = 7
Cost of making a review = 2
Cost of cancelling a review = 1
Are analysis requests to be included (y or n)? = y
Smallest time to perform a review = 0.5
1/2 random interval about due-in time = 1
Inventory on-hand = 15
No. of units due-in = 15

Make sure you insert events to receive due-ins
Ave. price for 1 units that is to be rejected is w(i).

\[ w(1) = 41 \]
\[ w(2) = 40 \]
\[ w(3) = 39 \]
\[ w(4) = 38 \]
Initialization Procedure

Simulation start time = 0
The no. of digits in fp of time(e.g., xxx.xx is 2) = 0
Max. No. of attributes for events = 3
No. of words of R to store these attributes = 3
Max no. of sets ever used = 1

Max. No. of variables collected in part 6 = 4
Set C(I,3)=0 or 1. A 1 implies that a histogram is desired for var. I.

\[
\begin{align*}
C(1,3) &= 0 \\
C(2,3) &= 0 \\
C(3,3) &= 0 \\
C(4,3) &= 0
\end{align*}
\]

Max. No. of variables collected in part 7 = 2
Max. no. of parameter sets used = 5

****** For Parameter Set 1

Parameter 1 = 5
Parameter 2 = 8
Parameter 3 = 12
Parameter 4 = 2

****** For Parameter Set 2

Parameter 1 = 0
Parameter 2 = -2
Parameter 3 = 2
Parameter 4 = 0

****** For Parameter Set 3

Parameter 1 = 0
Parameter 2 = -2
Parameter 3 = 2
Parameter 4 = 0

****** For Parameter Set 4

Parameter 1 = 0
Parameter 2 = -3
Parameter 3 = 3
Parameter 4 = 0

****** For Parameter Set 5

Parameter 1 = 0
Parameter 2 = -3
Parameter 3 = \frac{7}{2}
Parameter 4 = \overline{0}

Random number seed (a 0 sets seed to .47594118) = .52349371

Initial Set Entries (Entities). A 0 indicates no more sets. A value \leq -10^{4}5 indicates no more attributes for this entry

Entry is for set = 1
A(1) = 0
A(2) = 59
A(3) = \overline{0}
Respond with a 1 if OK to file the above = 1

Entry is for set = 1
A(1) = 3
A(2) = 69
A(3) = 15
Respond with a 1 if OK to file the above = 0
Entry not filed

Entry is for set = 1
A(1) = 8
A(2) = 69
A(3) = 15
Respond with a 1 if OK to file the above = 1

Entry is for set = 1
A(1) = 5
A(2) = 79
A(3) = 0
Respond with a 1 if OK to file the above = 1

Entry is for set = 1
A(1) = 100
A(2) = 89
A(3) = 0
Respond with a 1 if OK to file the above = 1

Entry is for set = 0
Type 1 if JASP deviate generators are used = 1
Type 1 if Uniform generator is used = 1
Type 1 if Lognormal generator is used = 0
Type 1 if Normal generator is used = 0
Type 1 if Erlang generator is used = 1

Step 334.3 files all as item 22 (initd).

Error at step 334.3: Please discard the item or use a new item number.
Discard item 22 (initd).
Done.
Go.
MAN-MACHINE INVENTORY SYSTEM

At this time the following alternatives are available:

2. Start the clock to begin the processing of customers;
3. Insert new parameter values and then start clock.

Your choice = 2

**** Customer requests 4 units at .00
Analysis (y or n)? = n

Price quote to customer = 38
Price too high. 4 units not sold

**** Customer requests 4 units at 3.21
Analysis (y or n)? = n

Price quote to customer = 37
Customer buys 4 units for 37.00

**** Customer requests 3 units at 4.25
Analysis (y or n)? = n

Price quote to customer = 37
Customer buys 3 units for 37.00

**** Customer requests 4 units at 4.82
Analysis (y or n)? = y

option = 4

Do you want a list of orders(y or n)? = y

Current list of orders

An order is due at time 8.00 for 15 units.
No. of units to order(0 for no order) = 10
Scheduled time for arrival of order is 13.00
Analysis (y or n)? = y

option = 6

To cancel order with desired property, set k equal to:
1. Largest no. of units; 2. Smallest no. of units;
3. Last due.in; 4. First due in.

k = 4

Order cancelled was to arrive at 8.00 with 15 units.
Analysis (y or n)? = n
Price quote to customer = 38
   Price too high. 4 units not sold 8 4 38.00 REJ.

**** Customer requests 1 units at 4.92
Analysis (y or n)? = n

Price quote to customer = 39
   Customer buys 1 units for 39.00 7 1 39.00 ACC.

   This is a scheduled review.
Analysis (y or n)? = y

   option = 1
   Set k= 1, for price data; 2, pct lost; 3, cost picture; 4, done.

   k = 1
   Ave. price pd. = 37.67  Ave. price rejected = 38.00  Ave. price quote = 37.80

   k = 2
   Percent customers that did not buy = 40.00

   k = 3

Sales completed = 8. Sales lost = 8. Purch.+Tran. cost = 184.00
Orders made = 1. Orders canc. = 1. Tot. Order Cost = 7.00
Reviews completed = 1. Reviews canc. = 0. Tot. Review cost = 2.00
Average inventory (units) = 13.10  Inventory Carrying Cost = 1.31
   Total Cost = 194.31
Total revenue from sales = 298.00
Average profit = 20.74 after 5.00 days.

   k = n
Analysis (y or n)? = n

**** Customer requests 3 units at 6.39
Analysis (y or n)? = n

Price quote to customer = 38
   Customer buys 3 units for 38.00 4 3 38.00 ACC.

**** Customer requests 1 units at 7.12
Analysis (y or n)? = n

Price quote to customer = 40
   Customer buys 1 units for 40.00 3 1 40.00 ACC.

**** Customer requests 3 units at 7.89
Analysis (y or n)? = y
option = 7
Set k = no. of days. Set k=0 when done.
k = 6
It is expected that 18 units will arrive in the next 6.00 days.
k = -1
Analysis (y or n)? = y
option = 4

Do you want a list of orders(y or n)? = n
No. of units to order(0 for no order) = 20
Scheduled time for arrival of order is 20.00
Analysis (y or n)? = 0
Suppress analysis until time = 15

Price quote to customer = 38
Customer buys 3 units for 38.00 0 3 38.00 ACC.

**** Customer requests 2 units at 8.31
Inventory not sufficient to meet demand

**** Customer requests 2 units at 12.74
Inventory not sufficient to meet demand

At time 13.15 an order of 18 units was received.

Current inventory on-hand = 18. Units on order = 20.

**** Customer requests 1 units at 13.69

Price quote to customer = 40
Customer buys 1 units for 40.00 17 1 40.00 ACC.

**** Customer requests 2 units at 14.68

Price quote to customer = 39.95
Customer buys 2 units for 39.95 15 2 39.95 ACC.

**** Customer requests 3 units at 15.45
Analysis (y or n)? = y

option = 8
Are you sure you're through(y or n)? = y
** SUMMARY REPORT **

Sales completed = 18. Sales lost = 12. Purch.+Trans. cost = 414.00
Orders made = 2. Orders canc. = 1. Tot. Order Cost = 14.00
Reviews completed = 1. Reviews canc. = 0. Tot. Review cost = 2.00
Average inventory (units) = 7.68 Inventory Carrying Cost = 2.37
Total revenue from sales = 685.90
Average profit = 16.41 after 15.45 days.

timer = 7.96

RANDOM NUMBER SEED = .52349871
START TIME = .000
SUMMARY REPORT PRINTED AT 15.448

<table>
<thead>
<tr>
<th>PARAMETER SET</th>
<th>VALUE 1</th>
<th>VALUE 2</th>
<th>VALUE 3</th>
<th>VALUE 4</th>
<th>VALUE 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.00</td>
<td>8.00</td>
<td>12.00</td>
<td>2.00</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>.00</td>
<td>-2.00</td>
<td>2.00</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>.00</td>
<td>-2.00</td>
<td>2.00</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>.00</td>
<td>-3.00</td>
<td>3.00</td>
<td>.00</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>.00</td>
<td>-3.00</td>
<td>2.00</td>
<td>.00</td>
<td></td>
</tr>
</tbody>
</table>

** STATISTICS COLLECTED IN PART 6 **

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>NO. OBS.</th>
<th>AVERAGE</th>
<th>STD. DEV.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X( 1)</td>
<td>8</td>
<td>3.861875 01</td>
<td>1.2950290 00</td>
<td>3.900000 01</td>
</tr>
<tr>
<td>X( 2)</td>
<td>2</td>
<td>3.800000 01</td>
<td>0</td>
<td>3.800000 01</td>
</tr>
<tr>
<td>X( 3)</td>
<td>10</td>
<td>3.849500 01</td>
<td>1.1715232 00</td>
<td>3.900000 01</td>
</tr>
<tr>
<td>X( 4)</td>
<td>2</td>
<td>1.900000 01</td>
<td>1.4142136 00</td>
<td>1.900000 01</td>
</tr>
</tbody>
</table>

** STATISTICS COLLECTED IN PART 7 **

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>END TIME</th>
<th>AVERAGE</th>
<th>STD. DEV.</th>
<th>MAX VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y( 1)</td>
<td>1.545 01</td>
<td>7.676999 00</td>
<td>6.7590540 00</td>
<td>1.800000 01</td>
</tr>
<tr>
<td>Y( 2)</td>
<td>1.545 01</td>
<td>2.416148 01</td>
<td>1.0071075 01</td>
<td>3.800000 01</td>
</tr>
</tbody>
</table>

** PRINTOUT FOR SET 1 **

AVERAGE NUMBER IN SET WAS 2.6636
MAXIMUM NUMBER IN SET WAS 5
CURRENT NUMBER IN SET IS 3
Event Time is 1.54 01  Event Code is 59

Attributes of Entries in Set 1

<table>
<thead>
<tr>
<th>J</th>
<th>P</th>
<th>S</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>999</td>
<td>2</td>
<td>1.6642 01</td>
<td>59</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>2.0445 01</td>
<td>69</td>
<td>2.0000 01</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>1.0000 02</td>
<td>89</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Timer = 10.27
Description of JASP Program for Example 2.

The man-machine price-inventory simulation involves four events which occur when (1) a demand is made; (2) an order is received; (3) a review takes place; and (4) the simulation is ended. Before starting the simulation descriptive information must be provided to the decision maker for the system, or to provide initialization routines to the analyst for changing the simulation's script (parameters). Both activities require display vehicles; JOSS provides this capability.

Figure 11 shows the part sequencing for Example 2. The division between the initial displays and the simulation is indicated by the dotted horizontal line to show core storage requirements can be reduced by overlaying the event simulation portions of the program on the initial displays.

During the simulation, interaction between the program and the decision maker is achieved through displays and demands for decisions. The displays can be grouped by decision type, and recalled from disk storage when necessary. The grouping of displays into parts, and of parts into items, is a design problem that involves the trade-off between storage requirements and computing time. When using JASP (or just JOSS), core storage management can become an involved and challenging problem. Only a small amount of design work was done for Example 2. (Because Example 2 is intended to demonstrate JASP, speed of programming was of prime concern to illustrate the facility with which the language could be used. Example 2 was programmed and running within two days. Frills were added after this.)

Only one set is required for the simulation. The attributes for each event type are shown on the following page.
Use file 714 (p8260)
Recall item 3 (invmm)
Do part 50

Fig. 11 -- Part Sequencing for Example 2
<table>
<thead>
<tr>
<th>Event Type</th>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Demand</td>
<td>Time</td>
</tr>
<tr>
<td>Receipt</td>
<td>Time</td>
</tr>
<tr>
<td>Review</td>
<td>Time</td>
</tr>
<tr>
<td>End</td>
<td>Time</td>
</tr>
</tbody>
</table>

The time between demands was assumed to be distributed according to the exponential, with mean time of \( m \cdot a \), where \( a \) is set during initialization and \( m \) is a function of the number of lost sales (see step 60.1).

The number of units requested by a customer was given the following discrete distribution:

<table>
<thead>
<tr>
<th>Number Requested, ( d )</th>
<th>Probability or Requesting ( d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.4</td>
</tr>
<tr>
<td>2</td>
<td>.3</td>
</tr>
<tr>
<td>3</td>
<td>.2</td>
</tr>
<tr>
<td>4</td>
<td>.1</td>
</tr>
</tbody>
</table>

The sampling from discrete distributions is rather simple in JOSS and requires only one step (see step 60.42), assuming that all possibilities can be typed on one line.

Other random characteristics associated with the problem involve the acceptance of a price by a customer which was assumed to be

\[
w(d) + V(d+1) \\
\text{for } d = 1, 2, 3, 4,
\]

where \( w(d) \) is the average price accepted if \( d \) units are demanded, and \( V(d+1) \) is a deviate from a uniform distribution with
parameters as stored in column $d+1$ of array $B$.

The values for $w(d)$ and the parameters of $B$ are set in the initialization routine. The time between the placement of an order and its receipt (lead time) was assumed to be distributed according to the Erlang distribution, with parameters as stored in $B(\cdot,1)$. A uniform deviate between $-v$ and $v$ was superimposed on this lead time to represent random variation not known to the decision maker.

The role of the decision maker and profit and cost figures are described in the description of the system. The data values are set during initialization. Through the initial data values, the analyst can inhibit the demand for the analysis displays and force reviews to be set by the decision maker. Other variations can easily be incorporated. One could be an automatic mode in which the decision maker specifies routines for setting prices, the reorder point and order quantity for a predetermined time period. These, however, are digressions from the objective of presenting JASP.

A listing of parts by item for Example 2 is presented in Figure 12. Item 3 (invmm) is the root segment for Example 2. Principally, it contains parts which: (1) initiate other parts; (2) provide core storage management functions; and (3) might be required by several other parts. Parts 61, 62, 64 and 91 are used with part 5 to perform operations on sets, and to illustrate the use of the set-processing capability. These parts need not be in the root segment. They were put there because it was anticipated that several parts might use them. Also their storage requirements were small. Parts 65, 66 and 67 represent efficient core storage management steps. It might be
appropriate to include them directly in JASP.

<table>
<thead>
<tr>
<th>Item</th>
<th>Number</th>
<th>Part Number</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 (INVMM)</td>
<td>30</td>
<td>Transfer to appropriate event</td>
<td></td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>Option control and option storage management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>Demand event storage management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>50*</td>
<td>Starting part number for entire simulation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>51*</td>
<td>Initiate simulation description</td>
<td></td>
</tr>
<tr>
<td></td>
<td>52*</td>
<td>Starting part number to begin clock</td>
<td></td>
</tr>
<tr>
<td></td>
<td>53*</td>
<td>Starting part number for reinitialization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>59</td>
<td>Demand event call</td>
<td></td>
</tr>
<tr>
<td></td>
<td>61</td>
<td>Sum the number of orders due-in within k days</td>
<td></td>
</tr>
<tr>
<td></td>
<td>62</td>
<td>Type list of reviews or list of orders outstanding</td>
<td></td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>Find review event scheduling for time t</td>
<td></td>
</tr>
<tr>
<td></td>
<td>65</td>
<td>Delete form k</td>
<td></td>
</tr>
<tr>
<td></td>
<td>66</td>
<td>Delete part k</td>
<td></td>
</tr>
<tr>
<td></td>
<td>67</td>
<td>Delete step k</td>
<td></td>
</tr>
<tr>
<td></td>
<td>69</td>
<td>Receipt of order event</td>
<td></td>
</tr>
<tr>
<td></td>
<td>79</td>
<td>Start of review event</td>
<td></td>
</tr>
<tr>
<td></td>
<td>89</td>
<td>End of simulation event</td>
<td></td>
</tr>
<tr>
<td></td>
<td>91</td>
<td>Determine order with specified characteristics</td>
<td></td>
</tr>
<tr>
<td></td>
<td>98*</td>
<td>JASP initialization and time event control start</td>
<td></td>
</tr>
<tr>
<td></td>
<td>99*</td>
<td>Demand average acceptance price for i units</td>
<td></td>
</tr>
<tr>
<td></td>
<td>100*</td>
<td>Non-JASP initialization</td>
<td></td>
</tr>
<tr>
<td></td>
<td>102</td>
<td>Analysis request</td>
<td></td>
</tr>
<tr>
<td></td>
<td>329</td>
<td>Initiate add-on steps for summary report</td>
<td></td>
</tr>
<tr>
<td>6 (demop)</td>
<td>33</td>
<td>Option list</td>
<td></td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>Demand event</td>
<td></td>
</tr>
<tr>
<td></td>
<td>63</td>
<td>Delete demand event part and forms and self</td>
<td></td>
</tr>
<tr>
<td>7 (opts1)</td>
<td>101</td>
<td>Data value displays</td>
<td></td>
</tr>
<tr>
<td></td>
<td>103</td>
<td>Review scheduling or list of reviews</td>
<td></td>
</tr>
<tr>
<td></td>
<td>104</td>
<td>Order placement or list of orders</td>
<td></td>
</tr>
<tr>
<td>8 (opts2)</td>
<td>105</td>
<td>Cancel a review</td>
<td></td>
</tr>
<tr>
<td></td>
<td>106</td>
<td>Cancel an order</td>
<td></td>
</tr>
<tr>
<td></td>
<td>107</td>
<td>Calculate number of orders due-in in k days</td>
<td></td>
</tr>
<tr>
<td></td>
<td>108</td>
<td>Trap to insure termination is desired</td>
<td></td>
</tr>
<tr>
<td>9 (dinv)</td>
<td>151</td>
<td>Situation description</td>
<td></td>
</tr>
<tr>
<td></td>
<td>152</td>
<td>Option list printing and storage management</td>
<td></td>
</tr>
<tr>
<td>10 (addon)</td>
<td>34</td>
<td>Cost picture display</td>
<td></td>
</tr>
</tbody>
</table>

*These parts are deleted once the clock starts.

Fig. 12 -- Listing of Parts by Item for Example 2
Parts other than those contained in the root segment are not highly interdependent. With a small amount of core storage management, a procedure was developed so that they can be recalled from disk storage as required. From the decision maker's standpoint the delays introduced by recalling these items was not excessive as they provided time for data analysis and strategic planning.

Figure 13 represents the non-JASP symbols used in Example 2. The JASP coding for Example 2 is presented in Figure 14.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Unmodified mean time between demands</td>
</tr>
<tr>
<td>b</td>
<td>Number of customers that have arrived</td>
</tr>
<tr>
<td>c(.)</td>
<td>Cost data and temporary storage</td>
</tr>
<tr>
<td>d</td>
<td>Number of units demanded by a customer</td>
</tr>
<tr>
<td>e</td>
<td>Minimum time required for a review</td>
</tr>
<tr>
<td>f</td>
<td>Amount of time required to cancel an order</td>
</tr>
<tr>
<td>g</td>
<td>Number of customers not sold</td>
</tr>
<tr>
<td>h,i,j,k</td>
<td>Indices and temporary storage</td>
</tr>
<tr>
<td>l</td>
<td>Not used</td>
</tr>
<tr>
<td>m</td>
<td>Modification &quot;a&quot; based on number of customers not sold</td>
</tr>
<tr>
<td>n</td>
<td>No, false</td>
</tr>
<tr>
<td>o</td>
<td>Not used</td>
</tr>
<tr>
<td>p</td>
<td>Price quoted</td>
</tr>
<tr>
<td>q(1)</td>
<td>Number of orders made</td>
</tr>
<tr>
<td>q(2)</td>
<td>Number of orders cancelled</td>
</tr>
<tr>
<td>r(1)</td>
<td>Number of reviews completed</td>
</tr>
<tr>
<td>r(2)</td>
<td>Number of review cancelled</td>
</tr>
<tr>
<td>s(1)</td>
<td>Number of sales completed</td>
</tr>
<tr>
<td>s(2)</td>
<td>Number of sales lost</td>
</tr>
<tr>
<td>t,u</td>
<td>Temporary storage</td>
</tr>
<tr>
<td>v</td>
<td>Actual lead time is within ± v of specified lead time</td>
</tr>
<tr>
<td>w(i)</td>
<td>Average price acceptance level when i units are requested</td>
</tr>
<tr>
<td>x</td>
<td>Revenue to date</td>
</tr>
<tr>
<td>y</td>
<td>Yes, true</td>
</tr>
<tr>
<td>z</td>
<td>True, if option request is from part 60; false otherwise</td>
</tr>
</tbody>
</table>

Fig. 13 -- List of non-JASP Symbols for Example 2
30.0 Do part E.
31.1 Do part 102.
31.12 Set z=n if k=n.
31.2 Done if k=n.
31.3 Line.
31.32 Do part 63 if z=v.
31.4 Demand j as "option".
31.5 To step [j=0; 31.9; j=2; 31.9; 31.6].
31.6 Recall item 7 (opts1) if j<4.
31.61 Recall item 8 (opts2) if j>4.
31.63 Do part 100 +j.
31.64 Do part 65 for k=43,44,49 if j<4.
31.65 Do part 65 for k=42,48 if j>4.
31.66 Do part 66 for k=101,103,104 if j<4.
31.67 Do part 66 for k=105(1)108 if j>4.
31.7 Do part 89 if t<0.
31.75 Do part 102.
31.755 Line.
31.76 To step 31.4 if k=v.
31.9 Do part 59 if z=y.
31.93 Set z=n if j=0.
31.94 Done if j=0.
31.95 Do part 32.
31.96 To step 31.3.
32.1 Recall item 6 (demop).
32.2 Do part 63.
32.3 Do part 33.
50.1 Page.
50.11 Type "MAN-MACHINE INVENTORY SYSTEM".
50.2 Type " "
50.3 Type "At this time the following alternatives are available:".
50.4 Line.
50.5 Type "1. Obtain a description of the inventory situation;".
50.6 Type "2. Start the clock to begin the processing of customers;".
50.7 Type "3. Insert new parameter values and then start clock;".
50.75 Line.
50.8 Demand k as "Your choice".
50.9 Do part 50 +k.
51.05 Do part 66 for k=32,50,52,69,91,98,100.
51.06 Recall item 22 (inltd).
51.1 Recall item 9 (invmx).
51.2 Do part 151.
51.3 Recall item 3 (invmx).
51.4 Delete step 50.5.
51.5 Do part 50.

Fig. 14 -- JASP Coding of Example 1
52.1 Do part 66 for k=50,51,53,98,99,100.
52.2 Recall item 22 (initd).
52.4 Do part 67 for k=52.1,52.2,52.4.
52.5 Do part 1.

53.1 Do part 100.

59.1 Recall item 6 (demop).
59.2 Do part 66 for k=33.
59.3 Do part 60 if z≤y.

61.1 Done if A(2)≤69.
61.2 Set u=u+A(3) if A(1)≤T+k.

62.1 Done if A(2)=k.
62.2 Set i=1.
62.3 Type A(1) in form 79 if A(2)=79.
62.4 Type ip(A(1)+.5),A(3) in form 73 if A(2)=69.

64.0 Set M(2)=0 if A(1)=u.
64.1 Done if A(2)=79.
64.2 Set M(2)=0 if A(1)=u.

65. Delete form k.
66. Delete part k.
67. Delete step k.

69.2 Do part 7 for i=1(1)2.
69.3 Set Y(1)=Y(1)+A(3).
69.4 Set Y(2)=Y(2)-A(3).
69.5 Line.
69.6 Type A(1),A(3) in form 61.
69.7 Line.
69.8 Type Y(1),Y(2) in form 62.
69.9 Do part 31.

79.1 Set r(1)=r(1)+1.
79.2 Do part 7 for i=1(1)2.
79.24 Line.
79.25 Type "This is a scheduled review."
79.3 Set c(-2)=c(-1).
79.4 Set c(-1)=-1.
79.5 Do part 31.
79.6 Set c(-1)=c(-2) if c(-1)=-1.

89.1 Do part 66 for k=31,32,59,61,62,69,79,91.
89.2 Do part 65 for k=61,62,73,79.
89.3 Do part 9.

91.03 Done if A(2)=69.
91.05 To step [k=1; 91.1; k=3; 91.1; 91.2].

Fig. 14 -- JASP Coding of Example 2 (con't.)
91.1 Done if A(1)=u.
91.15 To step 91.3.
91.2 Done if A(1)=u.
91.3 Set h=J.
91.4 Set u=A(1).

98.1 Do part 66 for k=50, 51, 52, 53, 99, 100.
98.2 Recall item 16 (datan).
98.3 Do part 2.
98.4 Do part 67 for k=98.1, 98.2, 98.3, 98.4.
98.5 Do part 1.

99.2 Demand w(i).

100.11 Set v=true.
100.12 Set n=false.

100.13 Demand a as "Mean time between arrivals of customers".
100.14 Set z=n.

100.2 Demand c(0) as "Purchase price of an item for the distributor".
100.21 Demand c(1) as "Inventory carrying charge for 1 unit for 1 day".
100.22 Demand c(2) as "Cost of ordering and transporting a unit".
100.23 Demand c(3) as "Cost for making an order (indep. of size)".
100.24 Demand c(4) as "Cost of making a review".
100.25 Demand c(5) as "Cost of cancelling a review".
100.26 Demand k as "Are analysis requests to be included (y or n)?".
100.27 Set c(-1)=[k=n:10#8; -1].
100.275 Let c be sparse.
100.28 Set m=1.
100.3 Set b=0.
100.31 Set t=1.
100.32 Set u=1.
100.33 Set i=1.
100.4 Set g=0.
100.5 Demand e as "Smallest time to perform a review".
100.55 Demand v as "1/2 random interval about due-in time".
100.6 Set q(1)=0.
100.62 Set q(2)=0.
100.64 Set r(1)=0.
100.65 Set r(2)=0.
100.66 Set s(1)=0.
100.67 Set s(2)=0.
100.671 Set x=0.
100.68 Demand Y(1) as "Inventory on-hand".
100.69 Demand Y(2) as "No. of units due-in".
100.7 Type _-
100.71 Type "Complete failure if Y(2)<1."
100.74 Type "At most 1 of i units that is to be rejected is w(i).".
100.72 Do part 99 for i=1(1)4.
100.8 Do part 98.

102.1 Set k=n if T<c(-1).
102.12 Done if T<c(-1).
102.2 Demand k as "Analysis (y or n)?".
102.3 To step [k=y:102.4; k=n: 102.4; 102.5].

Fig. 14 -- JASP Coding of Example 2 (con't.)
102.4 Done.
102.5 Demand c(-1) as "Suppress analysis until time".
102.6 To step 102.1.

329.021 Recall item 10 (addon).
329.022 Do part 34.

Form 61:  
At time ___._. an order of ___ units was received.

Form 62:  
Current inventory on-hand = ___. Units on order = ___.

Form 73:  
An order is due at time ___._. for ___ units.

Form 79:  
A review is scheduled at time ___._.

Type size.
  size = 797

Fig. 14 -- JASP Coding of Example 2 (con't.)
Delete all.
Recall item 6 (demop).
Done.
Type all.

33.1 Page if in(S/42)=0.
33.15 Line.
33.2 Type "        Option List".
33.22 Line.
33.25 Type "         0. For listing of options.".
33.31 Type "         1. Data values desired.".
33.32 Type "         2. No action really desired.".
33.33 Type "         3. Schedule a review or list of review times.".
33.34 Type "         4. Place an order for units or list of incoming orders.".
33.35 Type "         5. Cancel a review.".
33.36 Type "         6. Cancel an order.".
33.37 Type "         7. Obtain number of units due-in in k days.".
33.38 Type "         8. Desire to go out of business.".
33.4 Delete part 33.

60.1 Set m=[g/b<.01; 1; g/b<.05; 1.05; 1.1] if b<0.
60.12 Do part 15.
60.15 Set A(1)=T-m*a*log(Z).
60.2 Do part 3 for I=1.
60.3 Line.
60.35 Type form 41 if T<.02.
60.4 Do part 15. 
60.42 Set d=[Z<.4:1; Z<.7:2; Z<.9:3; 4].
60.44 Type d in form 40.
60.455 Type "Inventory not sufficient to meet demand" if d>Y(1).
60.466 To step 60.984 if d>Y(1).
60.47 Do part 31 for z=y.
60.48 Line.
60.5 Type form 41 if ip(1-S/14)=1.
60.70 Demand p as "Price quote to customer".
60.705 Set X(3)=p.
60.706 Do part 6 for I=3.
60.71 Set k=v.
60.72 Do part 16 for I=d+1.
60.73 Set k=n if p>w(d)+V(I).
60.74 To step [k=y; 60.75; 60.98].
60.75 Do part 7 for I=1.
60.755 Set Y(1)=Y(1)-d.
60.76 Type d,p,Y(1),d in form 45.
60.77 Set X(1)=p.
60.78 Do part 5 for I=1.
60.79 Set s(1)=s(1)+d.
60.791 Set x=x+p*d.
60.8 Set b=b+1.
60.92 Do part 63.
60.93 Done.
60.98 Type d,Y(1),d in form 46.
60.982 Set X(2)=p.
60.983 Do part 6 for I=2.

Fig. 14 -- JASP Coding of Example 2 (con't.)
60.984 Set s(2)=s(2)+d.
60.985 Set g=g+1.
60.99 To step 60.8.

63.1 Delete part 60.
63.2 Do part 65 for k=40,41,45,46.
63.3 Delete part 63.

Storage management

Form 40:
*** Customer requests ___ units at ___.

Form 41:

Form 45:
Customer buys ___ units for ___.

Form 46:
Price too high. ___ units not sold

Type size,
size = 318

INV DEM PRICE

____ ___ ___ ACC.

____ ___ ___ REJ.

Fig. 14 -- JASP Coding of Example 2 (con't.)
Delete all.
Recall item 7 (opts1).
Done.
Type all.

101.3 Type "Set k= 1, for price data; 2, pct lost; 3, cost picture; 4, done."
101.4 Line.
101.45 Demand k.
101.5 To step [k=1:101.6; k=2:101.73; k=3:101.82; 101.9].
101.6 Set u=G(4,1,0,C(1,3)).
101.62 Set i=G(4,1,0,C(2,3)).
101.63 Set t=G(4,1,0,C(3,3)).
101.64 Set u=1 if u>0.
101.65 Set i=1 if i>0.
101.66 Set t=0 if t>0.
101.7 Type C(1,1)/u , C(2,1)/i , C(3,1)/t in form 43. Option 1
101.71 Line.
101.72 To step 101.4.
101.73 Type 100*g/b in form 44.
101.75 Line.
101.8 To step 101.4.
101.82 Do part 66 for k=103,104.
101.825 Do part 65 for k=43,44,49.
101.83 Recall item 10 (addon).
101.84 Do part 34.
101.85 Recall item 7 (opts1).
101.86 To step 101.4.
101.9 Done.

103.05 Type "Current list of scheduled reviews".
103.1 Line.
103.15 Set i=0.
103.17 Set k=79. Option 3
103.2 Set M(1)=62.
103.3 Do part 5 for I=1.
103.35 Type " None" if i=0.
103.4 Demand k as "Do you want to schedule an inventory review(y or n)?".
103.5 To step [k=103.6; k=n: 103.9; 103.4].
103.6 Demand u as "Desired time for the review".
103.7 Type "Review cannot be started so quickly" if u<T+h.
103.74 Set A(1)=max(T+h,u).
103.76 Set A(2)=79.
103.78 Do part 3 for I=1.
103.8 Set r(1)=r(1)+1.
103.9 Done.

104.1 Line.
104.2 Demand k as "Do you want a list of orders(y or n)?". Option 4
104.3 To step [k=y:104.31; k=n: 104.5; 104.2].
104.31 Line.
104.32 Type "Current list of orders".
104.33 Set k=69.
104.34 Do step 104.95 for h=.1,.15,.2,.3,.35.
104.5 Demand X(4) as "No. of units to order(0 for no order)".

Fig. 14 -- JASP Coding of Example 2 (con't.)
Done if X(4)=0.
Set q(1)=q(1)+1.
Do part 6 for I=4.
Set A(3)=X(4).
Set A(2)=69.
Do part 19 for I=1.
Do part 15.
Set A(1)=T+V(1)-v+2*V+Z.
Do part 3 for I=1.
Do part 7 for I=2.
Set Y(2)=Y(2)+X(4).
Type ip(A(1)+.5) in form 49.
Done.
Do step 103+h.

Form 43:
Ave. price pd.= ___  Ave. price rejected= ___  Ave. price quote= ___

Form 44:
Percent customers that did not buy = ___

Form 49:
Scheduled time for arrival of order is ___

Type size,
size = 348

Fig. 14 -- JASP Coding of Example 2 (con't.)
Delete all.
Recall item 8 (opts2).
Done.
Type all.

105.1 Line.
105.4 Demand u as "Time of review to be cancelled".  
105.5 Set M(1)=64.
105.6 Do part 5 for I=1.
105.62 Type "No review scheduled at that time" if J=0.
105.63 Done if J=0.
105.66 Set r(2)=r(2)+1.
105.68 Set r(1)=r(1)-1.
105.7 Do part 4 for I=1.
105.8 To step 105.1.
105.9 Done.

106.1 Line.
106.2 Type "To cancel order with desired property, set k equal to:".
106.22 Type "1. Largest no. of units; 2. Smallest no. of units;".
106.3 Type "3. Last due in; 4. First due in.".
106.31 Demand k.
106.315 To step 106.31 if k>4.
106.32 Set u[k=1:0;k=3:0;10#9].
106.34 Set h=0.
106.36 Set j[k=2;3;1].
106.4 Set M(1)=91.
106.5 Do part 5 for I=1.
106.54 Set j=6.
106.55 Line.
106.6 Type "No such order or too late to cancel" if h=0.
106.7 Done if h=0.
106.75 Set J=h.
106.8 Do part 4 for I=1.
106.83 Do part 7 for I=2.
106.85 Set Y(2)=Y(2)-A(3).
106.87 Type A(1),A(3) in form 48.
106.88 Set q(2)=q(2)+1.
106.9 Done.

107.1 Type "Set k = no. of days. Set k<0 when done.".
107.3 Demand k.
107.35 Done if k<0.
107.4 Set u=0.
107.5 Set M(1)=61.
107.6 Do part 5 for I=1.
107.7 Type u,k in form 42.
107.8 To step 107.3.
107.9 Done.

108.1 Demand k as "Are you sure you're through(y or n)?".  
108.2 To step [key:108.3;k=n:108.9;108.1].
108.3 Set t=1.
108.9 Done.

Fig. 14 -- JASP Coding of Example 2 (con't.)
Form 42:
It is expected that ___ units will arrive in the next ___ days.

Form 48:
Order cancelled was to arrive at ____ with ____ units.

Type size.
size = 257

Fig. 14 -- JASP Coding of Example 2 (con't.)
Delete all.
Recall item 9 (dinvm).
Done.
Type all.

151.08 Type **_** A MAN-MACHINE PRICE-INVENTORY SIMULATION**_.**
151.11 Type **.**
151.12 Type **"In this simulation you are the distributor of a commodity."**
151.13 Type **"Customers request either 1, 2, 3, or 4 units of your commodity."**
151.14 Type **"The price of a unit is set by you. You quote only one price."**
151.15 Type **"to each customer. He will accept or reject your quote. "**
151.16 Type **"Inventory is adjusted accordingly and you can place an order."**
151.17 Type **"for more units. You also can schedule review times at which"**
151.18 Type **"orders can be placed or cancelled. Analysis of the inventory"**
151.19 Type **"can be made at review, demand and order receipt times."**
151.20 Type **.**
151.21 Type **"Your performance is measured in terms of profit per day."**
151.22 Type **"Your revenues are from sales, and your costs are for"**
151.23 Type **"purchase, carrying charges, ordering and transportation"**
151.24 Type **"and review. The specific cost characteristics are:"**
151.25 Line.
151.26 Type c(0) in form 80.
151.27 Type c(1) in form 81.
151.28 Type c(2) in form 82.
151.29 Type c(3) in form 83.
151.30 Type c(4) in form 84.
151.31 Type c(5) in form 85.
151.315 Line.
151.32 Do part 65 for k=80, 81, 82, 83, 84, 85.
151.33 Type **"There are no explicit costs for lost sales. However as the"**
151.34 Type **"number of customers rejecting your price quote increases "**
151.35 Type **"the demand for your commodity decreases."**
151.36 Type **.**
151.37 Type a i in form 86.
151.375 Type **"The ave. number of units requested/cust. 1,8"**
151.38 Type B(1,1)*B(1,4) in form 87.
151.39 Line.
151.4 Type **"An analysis can be performed at each event time."**
151.5 Line.
151.6 Type **"Analysis can be suspended until a time you specify by"**
151.7 Type **"responding with a 0 to the demand for analysis decision."**
151.8 Type **"The analysis options available to you are:"**
151.91 Do part 152.

152.1 Delete part 151.
152.2 Do part 65 for k=80, 87.
152.3 Recall item 6 (demop).
152.4 Do part 63.
152.5 Do part 33.
152.6 Delete part 152.

Form 80:
Purchased price of a unit to you ___.

Fig. 14 -- JASP Coding of Example 2 (con't.)
Form 81:
Inventory carrying charge for 1 unit for 1 day ___

Form 82:
Cost of ordering and transporting a unit ___

Form 83:
Cost for making an order (incurred even if order is cancelled) ___

Form 84:
Cost of performing a review ___

Form 85:
Cost if a review is cancelled ___

Form 86:
The average time between customers ___

Form 87:
The average time between the placement and receipt of an order ___

Type size,
  size = 419

Fig. 14 -- JASP Coding of Example 2 (con't.)
Delete all.
Recall item 10 (addon).
Done.
Type all.

34.1 Type ___.
34.15 Type \( s(1), s(2), s(1) \times (c(0) + c(2)) \) in form 69. 
34.2 Set \( c(7) = q(1) \times c(3) \).
34.3 Type \( q(1), q(2), c(7) \) in form 63.
34.4 Type \( r(1), r(2), r(1) \times c(4) + r(2) \times c(5) \) in form 64.
34.5 Type \( D(1,1)/T, D(1,1) \times c(1) \) in form 65 if \( T \neq 0 \).
34.6 Set \( c(6) = c(7) + s(1) \times (c(0) + c(2)) \times c(4) \times r(1) + c(5) \times r(2) + D(1,1) \times c(1) \).
34.7 Type \( c(6) \) in form 66.
34.8 Type \( x \) in form 67.
34.85 Type \( (x-c(6))/T, T \) in form 68 if \( T \neq 0 \).
34.86 Type ___.
34.9 Do part 65 for \( k=63(1)69 \).
34.91 Delete part 34.

**Form 63:**
Orders made = ____. Orders canc. = _____. Tot. Order Cost = _____.

**Form 64:**
Reviews completed = ___. Reviews canc. = _____. Tot. Review cost = _____.

**Form 65:**
Average inventory (units) = _____. Inventory Carrying Cost = _____.

**Form 66:**
Total Cost = _____.

**Form 67:**
Total revenue from sales = ______.

**Form 68:**
Average profit = _______ after _______ days.

**Form 69:**
Sales completed = ___. Sales lost = ___. Purch.+Tran. cost=_____

**Typ. size:**
size = 156
VI. CONCLUSIONS AND RECOMMENDATIONS

A simulation language is needed that facilitates the inclusion of man as part of a simulation; JASP fills this need. By imbedding JASP in the JOSS system, the inherent advantages of JOSS are obtained in JASP. These include the accessibility of the JOSS consoles, and the man-machine interaction available through the JOSS system.

JASP can significantly decrease the time required for writing simulation programs on JOSS. Because JASP is JOSS-based, the user need learn no new operating procedures; additions, deletions and modifications to the language are easily made.

The basic requirement for using JASP involves the identification of 19, or fewer part numbers, with their corresponding functions. Actually, only part numbers 3, 4, 5, 6, 7 are used frequently during the writing of a simulation program. In addition to part identification, the JASP user must learn the JASP symbols. Of the 25 JASP symbols, seven represent formulas, and eight are not normally accessed directly. The following are the ones to remember: \( A(\cdot) \), \( E \), \( I \), \( J \), \( T \), \( V(I) \), \( X(I) \), \( Y(I) \), \( Z \), \( M(1) \) and \( M(2) \) when \( M(1) \) and \( M(2) \) are used with the FIND routines (part 5).

JASP can be improved by the use of more robust symbols and functional names. This is not recommended at this time, as it would involve a large-scale effort. Experience with JASP should establish the need for such development.

It is anticipated that the JASP user's most urgent requirement is a core-storage management system; this should relieve the user of the necessity for chaining and overlaying, and represents an area for future research and development.
Appendix A

GLOSSARY OF JASP VARIABLES

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A(·)</td>
<td>Buffer storage for attribute i of an entity.</td>
</tr>
<tr>
<td>B(·,I)</td>
<td>Parameter values for distribution I.</td>
</tr>
<tr>
<td>B(1,I)</td>
<td>Parameter 1 (normally the mean).</td>
</tr>
<tr>
<td>B(2,I)</td>
<td>Minimum acceptable sample value.</td>
</tr>
<tr>
<td>B(3,I)</td>
<td>Maximum acceptable sample value.</td>
</tr>
<tr>
<td>B(4,I)</td>
<td>Parameter 2 (normally the standard deviation).</td>
</tr>
<tr>
<td>C(·,I)</td>
<td>Storage area for data concerning X(I).</td>
</tr>
<tr>
<td>C(1,I)</td>
<td>ΣX(I) for all values observed.</td>
</tr>
<tr>
<td>C(2,I)</td>
<td>ΣX²(I) for all values observed.</td>
</tr>
<tr>
<td>C(3,I)</td>
<td>A packed word in the following format:* xxxx--max X(I); x--1 if histogram for X(I) desired.</td>
</tr>
<tr>
<td>D(·,I)</td>
<td>Storage area for data concerning Y(I).</td>
</tr>
<tr>
<td>D(1,I)</td>
<td>ΣY(I) · Δt where Δt is increment of time that variable I had the value Y(I).</td>
</tr>
<tr>
<td>D(2,I)</td>
<td>ΣY²(I) · Δt.</td>
</tr>
<tr>
<td>D(3,I)</td>
<td>Max Y(I) observed at any time.</td>
</tr>
<tr>
<td>D(4,I)</td>
<td>Last time a change in variable I occurred.</td>
</tr>
<tr>
<td>E</td>
<td>Event code of current event.</td>
</tr>
<tr>
<td>F(·,I)</td>
<td>Attributes of set I.</td>
</tr>
<tr>
<td>F(1,I)</td>
<td>xxx Number in set I.</td>
</tr>
<tr>
<td></td>
<td>xxx Column number of last entity in set I.</td>
</tr>
<tr>
<td></td>
<td>xxx Column number of first entity in set I.</td>
</tr>
<tr>
<td>F(2,I)</td>
<td>xxx Maximum number of entities in set I at any one time.</td>
</tr>
<tr>
<td></td>
<td>xx Number of attributes of entities in set I,1.</td>
</tr>
<tr>
<td></td>
<td>x Number of words per entity in set I.</td>
</tr>
<tr>
<td></td>
<td>x Method of ranking entities in set I: 1=LVF; 2=HVF; 3=FIFO, 4=LILO.</td>
</tr>
<tr>
<td></td>
<td>xx Attribute used for ranking entities in set I.</td>
</tr>
<tr>
<td>F(3,I)</td>
<td>Time integrated number in set I.</td>
</tr>
<tr>
<td>F(4,I)</td>
<td>Last time a change in the number of entities in set I occurred.</td>
</tr>
<tr>
<td>G(D,L,Q,W)</td>
<td>Formula to &quot;get&quot; the D digits, located L digits from right with Q digits in fractional part of word W.</td>
</tr>
</tbody>
</table>

*Formats will be specified from left to right.*
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>H(I,·)</td>
<td>Storage area for histogram for data concerning X(I).</td>
</tr>
<tr>
<td>H(I,1)</td>
<td>x Not used.</td>
</tr>
<tr>
<td></td>
<td>xxx Lower limit of cell 2 of histogram.</td>
</tr>
<tr>
<td></td>
<td>xxx Width of each intermediate cell of the histogram.</td>
</tr>
<tr>
<td></td>
<td>xx Number of cells in histogram including end cells.</td>
</tr>
<tr>
<td>H(I,2)</td>
<td>xxx Number of times X(I) &lt; lower limit.</td>
</tr>
<tr>
<td></td>
<td>xxx Number of times X(I) ≥ lower limit and &lt; lower limit + width.</td>
</tr>
<tr>
<td></td>
<td>xxx Number of times X(I) ≥ lower limit + width and &lt; lower limit + 2·width.</td>
</tr>
<tr>
<td>H(I,j)</td>
<td>Analogous to H(I,2) where number of words j depends on number of cells requested by user.</td>
</tr>
<tr>
<td>I</td>
<td>An index; primarily used to indicate a set number or a variable number.</td>
</tr>
<tr>
<td>J</td>
<td>An index; primarily used to indicate a column number of an entity.</td>
</tr>
<tr>
<td>K(I)</td>
<td>Formula to obtain the column number of the first entity in set I.</td>
</tr>
<tr>
<td>L(I)</td>
<td>Formula to obtain the column number of the last entity in set I.</td>
</tr>
<tr>
<td>M(·)</td>
<td>Miscellaneous and temporary storage array.</td>
</tr>
<tr>
<td>N(I)</td>
<td>Formula to obtain number of entities in set I.</td>
</tr>
<tr>
<td>O</td>
<td>Not used.</td>
</tr>
<tr>
<td>P(J)</td>
<td>Formula to obtain column number of predecessor of entity in row J of array R.</td>
</tr>
<tr>
<td>Q</td>
<td>A zero when time attribute is not packed; otherwise number of digits in fractional part of time (used when time attribute is packed in a word).</td>
</tr>
<tr>
<td>R(·,J)</td>
<td>Storage area for entity in column J.</td>
</tr>
<tr>
<td>R(1,J)</td>
<td>xxx Column number of predecessor of entity in column J; if none 999.</td>
</tr>
<tr>
<td></td>
<td>xxx Column number of successor of entity in column J; if none 0.</td>
</tr>
<tr>
<td></td>
<td>xxx Attribute 2 of entity in column I.</td>
</tr>
<tr>
<td>R(j,J)</td>
<td>Attribute values associated with entity in column J as defined by user.</td>
</tr>
<tr>
<td>S(J)</td>
<td>Formula to obtain column number of successor of entity in column J of array R.</td>
</tr>
<tr>
<td>Symbol</td>
<td>Definition</td>
</tr>
<tr>
<td>--------</td>
<td>------------</td>
</tr>
<tr>
<td>T</td>
<td>Current time.</td>
</tr>
<tr>
<td>U(D,L,Q,W,V)</td>
<td>Formula to &quot;put&quot; the value V with Q digits in its fractional part in the D digits of word W located L digits from the right.</td>
</tr>
<tr>
<td>V(I)</td>
<td>Value of random deviate obtained from distribution with parameter values as stored in B(:,I).</td>
</tr>
<tr>
<td>W(·)</td>
<td>Storage area for initial values.</td>
</tr>
<tr>
<td>X(I)</td>
<td>Performance variable I (as defined by the user) for which statistics based on observations are desired.</td>
</tr>
<tr>
<td>Y(I)</td>
<td>Performance variable I (as defined by the user) for which statistics over time are desired.</td>
</tr>
<tr>
<td>Z</td>
<td>Current random number.</td>
</tr>
</tbody>
</table>
## Appendix B

### JASP Statement Types

<table>
<thead>
<tr>
<th>Function</th>
<th>Required JASP Inputs</th>
<th>Required statements</th>
<th>Next Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data initialization</td>
<td>None</td>
<td>Recall item 16</td>
<td>Return</td>
</tr>
<tr>
<td>Time and event control</td>
<td>None</td>
<td>Recall item 17</td>
<td>Part 30</td>
</tr>
<tr>
<td>Filing and entity in set I</td>
<td>A(·),I</td>
<td>Do part 3 for I = _</td>
<td>Return</td>
</tr>
<tr>
<td>Removing first entity from set I</td>
<td>I,J = 0</td>
<td>Set J = 0. Do part 4 for I = _</td>
<td>Return</td>
</tr>
<tr>
<td>Remove entity in column J of set I</td>
<td>I,J</td>
<td>Set J = _. Do part 4 for I = _</td>
<td>Return</td>
</tr>
<tr>
<td>Retrieve attribute values from column J and set I</td>
<td>I,J</td>
<td>Set J = _. Do part 14 for I = _</td>
<td>Return</td>
</tr>
<tr>
<td>Finding an entity that meets condition as specified by part M(I).</td>
<td>I,M(1),M(2)</td>
<td>Set M(I) = _. Do part 5 for I = _</td>
<td>Return</td>
</tr>
<tr>
<td>Part M(I) should set M(2)=0 when conditions are met. J is returned as the column number of the entity that meets the conditions. If none, J = 0.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obtaining a value, v, of attributes or combination of attributes of a set [v must be set by user]</td>
<td>I,M(1)</td>
<td>Set M(1)=part number containing combination and conditions. Do part 5 for I = _</td>
<td>Return</td>
</tr>
<tr>
<td>Finding an entity, e, with largest (smallest) attribute value, v [e and v must be set by user]</td>
<td>I,M(1)</td>
<td>Set M(1)=part number containing conditions. Do part 5 for I = _</td>
<td>Return</td>
</tr>
<tr>
<td>Collection of statistics on static variables</td>
<td>I,X(I)</td>
<td>Do part 6 for I = _ (A histogram is automatically collected if requested in data initialization.)</td>
<td>Return</td>
</tr>
</tbody>
</table>

---

*Note: The order for specifying two variables when performing a part is not important. Thus the following statements are equivalent to removing the first entity from set I:

Set I = _.
Do part 4 for I = 0.

If I was previously defined, then in the above only the second statement is required.*
<table>
<thead>
<tr>
<th>Function</th>
<th>Required JASP Inputs</th>
<th>Statement Types</th>
<th>Next Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection of statistics on dynamic (time persistent) variables</td>
<td>I, Y(I)</td>
<td>Do part 7 for I = ___.</td>
<td>Return</td>
</tr>
<tr>
<td>Error reporting when error occurs at step M(I)</td>
<td>M(I)</td>
<td>Do part 10 for M(I) = ___.</td>
<td>Console</td>
</tr>
<tr>
<td>Summary report</td>
<td>---</td>
<td>Do part 9.</td>
<td>Main program</td>
</tr>
<tr>
<td>Generate a random number</td>
<td>Z</td>
<td>Do part 15.</td>
<td>Return</td>
</tr>
<tr>
<td>Generate a deviate from a uniform distribution with parameters stored in B(·,I)</td>
<td>I</td>
<td>Do part 16 for I = ___. (Returns V(I) as the deviate.)</td>
<td>Return</td>
</tr>
<tr>
<td>Generate a deviate from a normal distribution with parameters stored in B(·,I)</td>
<td>I</td>
<td>Do part 17 for I = ___. (Returns V(I) as the deviate.)</td>
<td>Return</td>
</tr>
<tr>
<td>Generate a deviate from a lognormal distribution with parameters stored in B(·,I)</td>
<td>I</td>
<td>Do part 18 for I = ___. (Returns V(I) as the deviate.)</td>
<td>Return</td>
</tr>
<tr>
<td>Generate a deviate from an Erlang distribution with parameters stored in B(·,I)</td>
<td>I</td>
<td>Do part 19 for I = ___. (Returns V(I) as the deviate.)</td>
<td>Return</td>
</tr>
</tbody>
</table>

By inserting entities in the event set (set 1) the following monitoring conditions can be obtained:

1. To initiate the printing of each event at time t₁, set A(1) = t₁, and A(2) = 300 and file.
2. To end the printing of each event at time t₂ (t₂ > t₁), set A(1) = t₂, and A(2) = 300 and file.
3. To print event set at t, set A(1) = t, and A(2) = 301 and file.
4. To print all sets at t, set A(1) = t, and A(2) = 302 and file.
5. To print a summary report at t, set A(1) = t, and A(2) = 303 and file.
### Appendix C

**JASP Part Description and Interrelations**

<table>
<thead>
<tr>
<th>Part</th>
<th>Function</th>
<th>Parts It Uses</th>
<th>Used by JASP Parts</th>
<th>Forms Required</th>
<th>Formulas Required</th>
<th>Variables Required of User</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Time and event control</td>
<td>4,30,11</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>2</td>
<td>Data initialization</td>
<td>312,314,315,316,317</td>
<td>none</td>
<td>none</td>
<td>24</td>
<td>none</td>
</tr>
<tr>
<td>3</td>
<td>Filing an entry</td>
<td>310,309,10,311</td>
<td>4,9,14,311,332</td>
<td>none</td>
<td>G,N,L,K,P,S</td>
<td>A(·),I</td>
</tr>
<tr>
<td>4</td>
<td>Removing an entry</td>
<td>14,3,10</td>
<td>1</td>
<td>none</td>
<td>K,N,L,S,P</td>
<td>I,J</td>
</tr>
<tr>
<td>5</td>
<td>Search a set for an entity that meets specified conditions</td>
<td>14,M(1)</td>
<td>none</td>
<td>none</td>
<td>K,S</td>
<td>I,M(1)</td>
</tr>
<tr>
<td>6</td>
<td>Collection of statistics on static variables</td>
<td>8</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>I,X(I)</td>
</tr>
<tr>
<td>7</td>
<td>Collection of statistics on dynamic (time persistent) variables</td>
<td>none</td>
<td>9</td>
<td>none</td>
<td>none</td>
<td>I,Y(I)</td>
</tr>
<tr>
<td>8</td>
<td>Histogram</td>
<td>none</td>
<td>6</td>
<td>none</td>
<td>G</td>
<td>I,X(I)</td>
</tr>
<tr>
<td>9</td>
<td>Summary report</td>
<td>7,329,3</td>
<td>11</td>
<td>none</td>
<td>none</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>Error report</td>
<td>330</td>
<td>3,4</td>
<td>none</td>
<td>none</td>
<td>M(1)</td>
</tr>
<tr>
<td>11</td>
<td>Monitor</td>
<td>14,12,13,9</td>
<td>1,330</td>
<td>27,1</td>
<td>K</td>
<td>none</td>
</tr>
<tr>
<td>12</td>
<td>Print set I</td>
<td>14</td>
<td>11,13,325</td>
<td>28,1,29,2</td>
<td>K,P,S</td>
<td>I</td>
</tr>
<tr>
<td>13</td>
<td>Print all sets</td>
<td>12</td>
<td>11,330</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>14</td>
<td>Retrieve A(·)</td>
<td>3,308</td>
<td>4,5,11,12</td>
<td>none</td>
<td>none</td>
<td>I,J</td>
</tr>
<tr>
<td>15</td>
<td>Obtaining random number, Z</td>
<td>none</td>
<td>16,17,328</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>

*In this table, only the specific parts, forms, and formulas required by a part are given. Complete identification of parts, forms and formulas required when using a part requires tracing the sequential requirements of parts. For example, part 9 uses no forms but uses part 329 that requires many forms. The sequential part, form and formula requirements are taken care of by the overlay management provided in JASP. However, if JASP is modified, the user must take cognizance of the sequential requirements.*
<table>
<thead>
<tr>
<th>Part</th>
<th>Function</th>
<th>Parts It Uses</th>
<th>Used By</th>
<th>Forms Required</th>
<th>Formulas Required</th>
<th>Variables Required of User</th>
</tr>
</thead>
<tbody>
<tr>
<td>320</td>
<td>Print B(·,I)</td>
<td>none</td>
<td>329</td>
<td>11</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>321</td>
<td>Print statistics for X(I)</td>
<td>none</td>
<td>329</td>
<td>26,10</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>322</td>
<td>Print histogram headings for X(I)</td>
<td>323,326</td>
<td>329</td>
<td>14</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>323</td>
<td>Unpack histogram</td>
<td>none</td>
<td>322</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>324</td>
<td>Print Statistics for Y(I)</td>
<td>none</td>
<td>329</td>
<td>18</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>325</td>
<td>Print set statistics and entries</td>
<td>12</td>
<td>329</td>
<td>19,20,21,22</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>326</td>
<td>Print histogram for X(I)</td>
<td>none</td>
<td>322</td>
<td>15</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>327</td>
<td>Restrict range of random deviates</td>
<td>none</td>
<td>17,18,19</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>328</td>
<td>Obtain the product of random numbers</td>
<td>15</td>
<td>19</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>329</td>
<td>Printing of summary report</td>
<td>320, 321, 322, 324, 325</td>
<td>9</td>
<td>4,5,6,7,8,9,12,13,16,17</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>330</td>
<td>Printing of error report</td>
<td>13,11</td>
<td>10</td>
<td>30,3</td>
<td>none</td>
<td>M(1)</td>
</tr>
<tr>
<td>331</td>
<td>Obtain the sum of random numbers</td>
<td>15</td>
<td>17</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>332</td>
<td>Initializes R with initial entries. Files all values as item 22 (initd)</td>
<td>333,3</td>
<td>317</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>333</td>
<td>Demands A(·)</td>
<td>none</td>
<td>332</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>334</td>
<td>Starts simulation</td>
<td>none</td>
<td>317</td>
<td>none</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>Part</td>
<td>Function</td>
<td>Parts It Uses</td>
<td>Used by JASP Parts</td>
<td>Forms Required</td>
<td>Formulas Required</td>
<td>Variables Required of User</td>
</tr>
<tr>
<td>-------</td>
<td>---------------------------------------</td>
<td>---------------</td>
<td>--------------------</td>
<td>----------------</td>
<td>-------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>16</td>
<td>Obtaining a uniform deviate, V(I)</td>
<td>15</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>I</td>
</tr>
<tr>
<td>17</td>
<td>Obtaining a normal deviate, V(I)</td>
<td>15, 327</td>
<td>18</td>
<td>none</td>
<td>none</td>
<td>I</td>
</tr>
<tr>
<td>18</td>
<td>Obtaining a lognormal deviate, V(I)</td>
<td>17</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>I</td>
</tr>
<tr>
<td>19</td>
<td>Obtaining an Erlang deviate, V(I)</td>
<td>328, 327</td>
<td>none</td>
<td>none</td>
<td>none</td>
<td>I</td>
</tr>
</tbody>
</table>

The following parts are used by JASP parts 1-19 and although available to the JASP user are not normally required by him. (Thus there is little need to become familiar with these parts.) Parts 308 and 309 are used to pack and unpack the array R and can be rewritten by the user.

<table>
<thead>
<tr>
<th>Part</th>
<th>Function</th>
<th>Used by JASP Parts</th>
<th>Forms Required</th>
<th>Formulas Required</th>
<th>Variables Required of User</th>
</tr>
</thead>
<tbody>
<tr>
<td>308</td>
<td>Unpack attributes from array R</td>
<td>none</td>
<td>14, 311</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>309</td>
<td>Pack attributes into array R</td>
<td>none</td>
<td>3</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>310</td>
<td>Locate an available column, J, in R</td>
<td>none</td>
<td>3</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>311</td>
<td>Stores an entry, J, in proper sequence in set I</td>
<td>3</td>
<td>none</td>
<td>P, S</td>
<td></td>
</tr>
<tr>
<td>312</td>
<td>Initialize priority, ranking value, and number of attributes in a set</td>
<td>2, 23</td>
<td>none</td>
<td>As demanded</td>
<td></td>
</tr>
<tr>
<td>313</td>
<td>Not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>314</td>
<td>Initialize last time variable changed value</td>
<td>2</td>
<td>none</td>
<td>none</td>
<td></td>
</tr>
<tr>
<td>315</td>
<td>Initialize histogram information</td>
<td>2</td>
<td>none</td>
<td>None</td>
<td>As demanded</td>
</tr>
<tr>
<td>316</td>
<td>Initialize parameter sets</td>
<td>2</td>
<td>25</td>
<td>none</td>
<td>As demanded</td>
</tr>
<tr>
<td>317</td>
<td>Deletion and recall parts and forms</td>
<td>332, 334</td>
<td>2</td>
<td>none</td>
<td>none</td>
</tr>
<tr>
<td>318</td>
<td>Not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>319</td>
<td>Not used</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Appendix D

**JASP ITEM--PART IDENTIFICATION**

<table>
<thead>
<tr>
<th>Item</th>
<th>Parts</th>
<th>Forms</th>
<th>Formulas</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 (DATAN)</td>
<td>2, 312-317, 332-334</td>
<td>23, 24, 25</td>
<td>none</td>
</tr>
<tr>
<td>17 (JASP)</td>
<td>1, 3-10, 14, 15, 310, 311</td>
<td>none</td>
<td>G, K, L, N, P, S, U</td>
</tr>
<tr>
<td>20 (rpts)</td>
<td>11-13</td>
<td>1, 2, 27-29</td>
<td>none</td>
</tr>
<tr>
<td>21 (error)</td>
<td>330</td>
<td>3, 30</td>
<td>none</td>
</tr>
<tr>
<td>22 (initd)</td>
<td>all initial values</td>
<td></td>
<td>none</td>
</tr>
<tr>
<td>23 (sumry)</td>
<td>220-326, 329</td>
<td>1, 2, 4-22, 26, 28, 29</td>
<td>none</td>
</tr>
<tr>
<td>24 (dist)</td>
<td>16-19, 327, 328, 331</td>
<td>none</td>
<td>none</td>
</tr>
</tbody>
</table>
Delete all.
Recall item 16 (data).
Done.
Type all.

2.07 Reset timer.
2.08 Page.
2.08a Type "*** NOTE: Most Berman interchanged rows and cols of***.
2.08b Type " the arrays R, B, C, D, and F. 2/7/70 ***".
2.08c Line.
2.08d Type "Initialization Procedure".
2.08e Line.
2.12 Demand W(1) as "Simulation start time".
2.13 Set T=W(1).
2.13a Demand Q as "The no. of digits in fp of time(e.g. xxx.xx is 2)".
2.13b Set F(1,1)=0.
2.13c Set F(3,1)=0.
2.13d Set F(4,1)=0.
2.13e Demand W(6) as "Max. No. of attributes for events".
2.13f Demand W(5) as "No. of words of R to store these attributes".
2.13g Set F(2,1)=W(6)*10^8+W(5)*1000+101.
2.14 Demand W(2) as "Max no. of sets ever used".
2.14a To step 2.16 if W(2)<1.
2.14b Line.
2.14c Type "Set codes are: 1=LVF, 2=LVF, 3=FIFO, 4=LIFO, 5=1-LIFO, 6=2-FIFO".
2.14d Line.
2.15 Do part 312 for I=2(1)W(2).
2.16 Let F be sparse.
2.16a Let H be sparse.
2.16b Line.
2.17 Demand W(3) as "Max. No. of variables collected in part 6".
2.18 To step 2.20 if W(3)<0.
2.18a Type form 24.
2.19 Do part 315 for I=2(1)W(3).
2.20 Let C be sparse.
2.20a Line.
2.21 Demand W(4) as "Max. No. of variables collected in part 7".
2.22 To step 2.24 if W(4)<0.
2.23 Do part 316 for I=2(1)W(4).
2.24a Let D be sparse.
2.24 Line.
2.31 Demand W(5) as "Max. no. of parameter sets used".
2.32 To step 2.35 if \( W(5) \leq 0 \).
2.33 Do part 316 for \( I = 1(1)W(5) \).
2.34 Let \( B \) be sparse.
2.35 Line.
2.36 Set \( h(1,2) = 0 \).
2.37 Let \( H \) be sparse.
2.38 Demand \( W(6) \) as "Random number seed (a 0 sets seed to \( 47594118 \))."
2.39 Set \( W(6) = 47594118 \) if \( W(6) = 0 \).
2.40 Set \( Z = W(6) \).
2.41 Line.
2.42 Set \( X(-99) = 0 \).
2.43 Let \( X \) be sparse.
2.44 Set \( Y(-99) = 0 \).
2.45 Let \( Y \) be sparse.
2.46 Do part 317.

312.05 Type I in form 23.
312.1 Demand \( M(6) \) as "The max. no. of attributes for this set".
312.2 Demand \( M(5) \) as "Max. no. of words to store these attributes".
312.25 Set \( M(8) = 0 \).
312.26 Demand \( H(7) \) as "Code for establishing priority".
312.3 Type "No ranking attribute required for FIFO or LIFO." if \( 3 \times H(7) = 4 \).
312.4 To step 312.6 if \( 3 \times M(7) \neq 4 \).
312.5 Demand \( M(9) \) as "The ranking attribute no.".
312.6 Set \( F(2, I) = H(6) \times 10^{84} + H(5) \times 100 + H(7) \times 10 + H(8) \).
312.7 Set \( F(4, I) = T \).
312.8 Line.
312.8 Set \( D(3, I) = T \times 10^4 \).

314.2 Demand \( C(3, I) \).
314.1 Line.
314.12 Done if \( C(3, I) = 0 \).
314.15 Set \( M(-9) = 1 \).
314.2 Demand \( H(6) \) as "Lower limit of cell 2 in interval \( [1, 99.9] \)."
314.3 Demand \( H(7) \) as "Width of each cell in interval \( [1, 99.9] \)."
314.4 Demand \( H(8) \) as "Number of cells (including end cells)".
314.5 Set \( H(1, I) = H(6) \times 10^{86} + H(7) \times 10^3 + H(8) \).

316.04 Line.
316.05 Type I in form 25.
316.06 Line.
316.1 Demand \( B(1, I) \) as "Parameter 1".
316.2 Demand \( B(2, I) \) as "Parameter 2".
316.3 Demand \( B(3, I) \) as "Parameter 3".
316.4 Demand \( B(4, I) \) as "Parameter 4".

317.1 Delete part 2, part 312, part 314, part 315, part 310.
317.15 Delete form 23, form 24, form 25.
317.2 Do part 332.
317.22 Delete part 332, part 333.
317.31 Delete part 6, part 8 if \( W(3) \leq 0 \).
317.32 Delete part 7 if \( W(4) \leq 0 \).
317.325 To step 317.35 if \( W(3) \leq 0 \).
317.33 Delete part 8 if M(9)≠1.
317.35 Demand M(6) as "Type 1 if JACP deviate generators are used".
317.36 To step 317.591 if M(6)≠1.
317.37 Recall item 24 dist.
317.38 Demand M(6) as "Uniform generator is used".
317.39 Delete part 16 if M(6)≠1.
317.41 Delete part 16 if M(6)≠1.
317.42 To step 317.55 if M(6)≠1.
317.43 Demand I as "Parameter Set No. to be modified (0 if no more)".
317.44 To step 317.57 if I≠0.
317.45 Set B(4,1)=log(B(4,1)²/B(1,1)²+1).
317.46 Set B(2,1)=log(B(2,1)).
317.47 Set B(3,1)=log(B(3,1)).
317.48 Set B(1,1)=log(B(1,1))-.5*B(4,1).
317.49 Set B(4,1)=sqrt(B(4,1)).
317.50 To step 317.43.
317.55 Demand M(6) as "Type 1 if Normal generator is used".
317.56 Delete part 17 part 331 if M(6)≠1.
317.57 Demand M(6) as "Type 1 if Erlang generator is used".
317.58 Delete part 19 part 328 if M(6)≠1.
317.591 Line.
317.592 Type "Step 334.3 files all as item 22 (initd).".
317.594 Line.
317.6 Do part 334.
332.03 Recall item 17 (jasp).
332.1 Set R(1,1)=0.
332.2 Let R be sparse.
332.21 Set A(1)=0.
332.22 Let A be sparse.
332.3 Type "Initial Set Entries (Entities). A 0 indicates no more sets."
332.4 Type "A value ≤ -10^8 indicates no more attributes for this entry".
332.45 Line.
332.5 Demand I as "Entry is for set".
332.6 To step 332.95 if I≠0.
332.63 Set M(7)=G(2,4,0,F(2,1)).
332.65 Do part 333 for J=1(1)M(7).
332.66 Demand M(6) as "Respond with a 1 if OK to file the above".
332.665 Type "Entry not filed" if M(6)≠1.
332.666 Line.
332.67 To step 332.5 if M(6)≠1.
332.7 Do part 3.
332.8 To step 332.5.
332.95 Done.
333.1 Demand A(J).
333.2 Done if A(J)>-10^5.
333.3 Set A(J)=0.
333.31 Set J=J+1.
333.32 To step 333.3 if J≤M(7).
333.4 Quit.
334.1 Delete part 317, step 334.1.
334.3 file all as item 22 (inits).

Form 23:

Form 24:
Set C(3,1)=0 or 1. A 1 implies that a histogram is desired for var. I.

Form 25:

Type size:
\[ \text{size} = 036 \]
Use file 714 (p8260).
Roger.
Delete all.
Recall item 17 (jasp).
Done.
Type all.

1.06 Reset timer.
1.1 Set J=0.
1.2 Do part 4 for I=1.
1.31 Set E=A(2).
1.32 Set T=A(1).
1.325 To step 1.4 if E≥300.
1.33 Do part 30.
1.34 Do part 11 if M(-99)=1.
1.35 To step 1.1.
1.4 Done if E=999.
1.5 Recall item 20 (rpts) if M(-99)=0.
1.6 Do part 11.
1.7 To step 1.1.
3.10 Do part 310 for J=1(1)999.

3.11 Set M(6)=G(2,4,0,F(2,I)).

3.12 Do part 309 for M(7)=1(1)M(6).

3.16 To step 3.20 if N(I)≠0.

3.17 Set F(1,I)=F(1,I)+J*(10^8*3+1).

3.18 Set R(1,J)=R(1,J)+999*10^8*6.

3.19 To step 3.61.

3.20 Set M(1)=G(1,2,0,F(2,I)).

3.21 To step [M(1)=1: 3.31; M(1)=2: 3.34; M(1)=3: 3.24; M(1)=4: 3.28; 3.22].

3.22 To step [M(1)=5: 3.31; M(1)=6: 3.41; 3.23].

3.23 Do part 10 for M(1)=3.21.

3.24 Set M(2)=L(I).

3.25 Set R(1,M(2))=R(1,M(2))+J*10^3.

3.26 Set F(1,I)=F(1,I)+1000*(J-M(2)).

3.27 Set R(1,J)=R(1,J)+10^6*M(2).

3.275 To step 3.61.

3.28 Set M(2)=K(I).

3.285 Set R(1,M(2))=R(1,M(2))+(J-999)*10^6.

3.29 Set F(1,I)=F(1,I)+J-M(2).

3.295 Set R(1,J)=R(1,J)+M(2)·1000+999·10^6.

3.30 To step 3.61.

3.31 Set M(2)=L(I).

3.32 Set M(3)=1.

3.33 Set M(7)=100·fp(F(2,I)/100).

3.331 Set M(4)=M(3)·A(M(7)).

3.332 Set M(5)=J.

3.333 To step 311.03.

3.335 Set M(2)=P(M(2)) if M(3)=-1.

3.34 To step 3.28 if M(2)=999.

3.345 To step 3.25 if M(2)=L(I).

3.35 To step 3.51.
3.41 Set $M(2) = K(I)$.  
3.42 Set $M(3) = -1$.  
3.43 To step 3.33.  
3.51 Set $M(3) = S(M(2))$.  
3.52 Set $R(1,J) = R(1,J) + 10^{86} \times M(2) + 10^{83} \times M(3)$.  
3.53 Set $R(1,M(2)) = R(1,M(2)) + 1000 \times (J - M(3))$.  
3.54 Set $R(1,M(3)) = R(1,M(3)) + 10^{86} \times (J - M(2))$.  
3.61 Set $F(3,I) = F(3,I) + N(I) \times (T - F(4,I))$.  
3.62 Set $F(1,I) = F(1,I) + 10^{86}$.  
3.63 Set $M(4) = (F(2,I)/10^{86})$.  
3.64 Set $F(2,I) = 10^{86} \times (N(I) - M(4)) + F(2,I)$ if $N(I) > M(4)$.  
3.65 Set $F(4,I) = T$.  

4.11 Set $J = K(I)$ if $J = 0$.  
4.12 Do part 10 for $M(1) = 4.12$ if $J = 0$.  
4.21 Do part 14.  
4.22 Set $T = A(1)$ if $I = 1$.  
4.23 Do step 3.61.  
4.24 Do step 3.65.  
4.31 To step $[N(I) = 1: 4.45; J = K(I); 4.32; J = L(I); 4.35; 4.38]$.  
4.32 Set $F(1,I) = F(1,I) + S(J) - J$.  
4.33 Set $R(1,S(J)) = R(1,S(J)) + 10^{86} \times (P(J) - J)$.  
4.34 To step 4.51.  
4.35 Set $F(1,I) = F(1,I) + (P(J) - J) \times 1000$.  
4.36 Set $R(1,P(J)) = R(1,P(J)) + 10^{83} \times (S(J) - J)$.  
4.37 To step 4.51.  
4.38 Do step 4.33.
4.39 To step 4.36.
4.45 Set \( F(1,I)=0 \).
4.46 To step 4.53.
4.51 Set \( F(1,I)=F(1,I)-10^6 \).
4.53 Set \( R(1,J)=0 \).
4.535 Set \( M(5)=G(1,3,0,F(2,I)) \).
4.537 Set \( M(4)=[J\neq1:1;2] \).
4.54 Do step 4.97 for \( M(7)=M(4)(1)M(5) \).
4.55 Done.
4.97 Delete \( R(M(7),J) \).

5.1 Set \( J=K(I) \).
5.13 Set \( M(2)=1 \).
5.15 Done if \( J=0 \).
5.2 Do part 14.
5.3 Do part \( M(1) \).
5.35 Done if \( M(2)=0 \).
5.4 Set \( J=S(J) \).
5.5 To step 5.15.

6.1 Set \( C(1,I)=C(1,I)+X(I) \).
6.2 Set \( C(2,I)=C(2,I)+X(I)\cdot X(I) \).
6.3 Set \( C(3,I)=C(3,I)+10^5\cdot (\text{ip}(X(I))-\text{ip}(C(3,I)/10^5)) \) if \( X(I)\cdot 10^5>C(3,I) \).
6.4 Set \( C(3,I)=C(3,I)+10 \).
6.5 Do part 8 if \( \text{fp}(C(3,I)/10)>0 \).

7.1 Set \( M(1)=D(4,I) \).
7.2 Set $D(1,I) = D(1,I) + Y(I) \cdot (T - M(1))$.
7.3 Set $D(2,I) = D(2,I) + Y(I) \cdot Y(I) \cdot (T - M(1))$.
7.4 Set $D(4,I) = T$.
7.5 Set $D(3,I) = Y(I)$ if $Y(I) > D(3,I)$.

8.1 Set $M(1) = G(3,5,1,H(I,1))$.
8.15 To step 8.72 if $X(I) < M(1)$.
8.2 Set $M(2) = \text{ip}((X(I) - M(1)) / G(3,2,1,H(I,1))) + 2$.
8.3 Set $M(3) = 100 \cdot \text{fp}(H(I,1) / 100)$.
8.4 Set $M(2) = M(3)$ if $M(2) > M(3)$.
8.5 Set $M(3) = \text{ip}((M(2) - 1) / 3)$.
8.55 Set $M(4) = \text{fp}(M(2) / 3)$.
8.6 Set $M(1) = [M(4) = 0: 0; M(4) < .34: 6; 3]$.
8.7 Set $H(I,M(3) + 2) = H(I,M(3) + 2) + 10^2 M(1)$.
9.71 Done.
9.72 Set $H(I,2) = H(I,2) + 10^8$.

9.1 Do part 7 for $I = 1(1) W(4)$ if $W(4) = 0$.
9.2 Delete part 1, part 3, part 4, part 5, part 10, part 15, part 311.
9.22 Delete part 6 if $W(3) > 0$.
9.23 Delete part 7 if $W(4) > 0$.
9.3 Recall item 23 (sumry).
9.4 Do part 329.
9.5 Recall item 22 (initd).
9.6 Set $A(1) = T - .01$.
9.7 Set $A(2) = 999$.
9.8 Do part 3 for $I = 1$.

10.1 Page.
10.2 Recall item 21 (error).
10.3 Do part 330.
10.4 Type "Error Report Completed".
10.5 Page.

10.6 Demand M(5) as "Your Move?".

14.1 Set M(6)=G(2,4,0,F(2,1)).
14.2 Do part 308 for M(7)=1(1)M(6).

15.0 Set Z=fp(23•Z)-ip(23•Z)/10^7.

308. Set A(M(7))=[M(7)>2:R(M(7),J);M(7)=1:R(2,J);1000•fp(R(1,J)/1000)].

309.1 Set R(2,J)=A(1) if M(7)=1.
309.2 Set R(1,J)=A(2) if M(7)=2.
309.3 Set R(M(7),J)=A(M(7)) if M(7)>2.

310. Quit if R(1,J)=0.

311.03 Set M(8)=A(M(7)).
311.04 Do part 308 for J=M(2).
311.05 Set M(9)=A(M(7)).
311.06 Set A(M(7))=M(8).
311.07 Set J=M(5).
311.075 To step 311.19 if M(1)≥5.
311.08 To step 3.335 if $A(M(7)) \geq M(9)$.
311.1 Set $M(2) = F(M(2))$ if $M(3) = 1$.
311.14 To step 3.28 if $M(2) = 999$.
311.15 Set $M(2) = S(M(2))$ if $M(3) = -1$.
311.17 To step 3.24 if $M(2) = 0$.
311.18 To step 311.03.
311.19 To step 3.335 if $A(M(7)) > M(9)$.
311.20 To step 311.1.

\[
G(D, L, Q, W) = 10^A(D-Q) \cdot fp(ip(W/10^A L)/10^A D)
\]
\[
K(I) = 10^A 3 \cdot fp(F(1, I)/10^A 3)
\]
\[
L(I) = G(3, 3, 0, F(1, I))
\]
\[
N(I) = ip(F(1, I)/10^A 6)
\]
\[
P(J) = ip(R(1, J)/10^A 6)
\]
\[
S(J) = ip(R(1, J)/1000) - P(J) \cdot 1000
\]
\[
U(D, L, Q, W, V) = 10^A L \cdot (W/10^A L - G(D, L, 0, W) + V \cdot 10^A Q)
\]

Type size.

size = 866
Use file 714 (p8260).
Roger.
Recall item 20 (rpts).
Done.
Type all.

11.05 Recall item 20 (rpts) if E>300.
11.10 Set M(-99)=1-M(-99) if E=300.
11.11 To step 11.20 if M(-99)=0.
11.12 Type T,P in form 27.
11.125 Set I=1.
11.13 Do part 14 for J=K(1).
11.14 Type "Attributes of next event are".
11.145 Type form 1.
11.15 Type J,P(J),S(J),A(1),A(2),A(3),A(4),A(5),A(6) in form 2.
11.16 Line.
11.20 To step [E=301:11.3; E=302:11.4; E=303:11.5; 11.57].
11.30 Do part 12 for I=1.
11.39 To step 11.55.
11.40 Do part 13.
11.49 To step 11.55.
11.50 Do step 11.61 for I=1(1)4.
11.55 Delete part 12, part 13.
11.57 Done.
11.61 Do step 9.1 + I/10.

12.05 Page if ip(S/43)#0.
12.1 Type ____________.
12.12 Do step 11.12 if I=1.
12.14 Line.
12.2 Type I in form 28.
12.25 Line.
12.27 Set J=K(I).
12.3 Type form 1 if J#0.
12.32 Line.
12.345 Type I in form 29 if J=0.
12.346 Done if J=0.
12.35 Do part 14.
12.36 Do step 11.15.
12.37 Set J=S(J).
12.375 Done if J=0.
12.38 To step 12.35.

13.1 Do part 12 for I=1(1)W(2).

Form 1:

<table>
<thead>
<tr>
<th>J</th>
<th>P</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

Form 2:

_________ __________ __________ __________ __________ __________

Form 27:

Event Time is _______ Event Code is _______

Form 28:

Attributes of Entries in Set _______

Form 29:

No Entries in set _______
Delete all.
Recall item 21 (error).
Done.
Type all.

330.05 Recall item 20 (rpts).
330.1 Type W(1) in form 30.
330.2 Do part 13.
330.22 Type
330.25 Type "..." Attributes of Sets - Array F".
330.3 Do step 330.91 for J=1(1)W(2).
330.4 Type "..." Storage for X-variables - Array C".
330.5 Do step 330.93 for J=1(1)W(3) if W(3)#0.
330.6 Type "..." Storage for Y-variables - Array D".
330.7 Do step 330.95 for J=1(1)W(4) if W(4)#0.
330.8 Delete part 330, form 3, form 30.
330.82 Done.
330.91 Type J,F(1,J),F(2,J),F(3,J),F(4,J) in form 3.
330.93 Type J,C(1,J),C(2,J),C(3,J) in form 3.
330.95 Type J,D(1,J),D(2,J),D(3,J) in form 3.

Form 3:

            ...........            ...........            ...........            ...........

Form 30:
JASP Error Report - Error at step _______

Type size.
size = 124
Use file 714 (p8260).
Roger.
Recall item 23 (sunny).
Done.
Type all.

320.1 Do step 320.2 for J=1(1)4.
320.3 Type I,A(1),A(2),A(3),A(4) in form 11.
320.5 Done.
320.2 Set A(J)=B(J,I).

321.1 Set M(1)=10^(3)*fp(ip(C(3,I)/10)/10^(3)).
321.2 Set M(2)=ip(C(3,I)/10^(5)).
321.3 Type I in form 26 if M(1)=0.
321.4 Set M(4)=sqrt((C(2,I)-M(1)*M(3)*M(3))/(M(1)-1)) if M(1)>1.
321.5 Type I,M(1),M(3),M(4),M(2) in form 10.

322.1 Set M(1)=10*fp(C(3,I)/10).
322.2 Done if M(1)=0.
322.3 Set M(1)=100*fp(H(I,J)/100).
322.4 Set M(2)=100*fp(ip(H(I,J)/1000).
322.5 Set M(3)=100*fp(ip(H(I,J)/10^5)/1000).
322.6 Set M(-2)=ip(M(I)-1)/3+2.
322.7 Do part 323 for J=2(1)M(-2).
322.8 Type I,M(3),M(2) in form 14.
322.9 Do part 326 for J=1(1)ip[(M(I)-1)/12+1].
322.10 Delete A.
322.11 Set A(I)=0.
322.12 Let A be sparse.

323.1 Do step 323.2 for E=1(1)3.
323.5 Done.
323.2 Set M(3+J-3+E)=10^(3)*fp(ip(H(I,J)/10^(3*(3-I)))/10^(3)).

324.04 Type " No values recorded. T=start time." if T>W(1).
324.1 Set M(1)=D(3,I).
324.2 Set M(2)=D(1,I)/(T-W(1)).
324.3 Set M(3)=sqrt(D(2,I)/(T-W(1))-M(2)*M(2)).
324.4 Type I, T, M(2), M(3), M(1) in form 18.

325.1 Set M(1)=ip(F(1,I)/10^(8)).
325.2 Set M(2)=ip(F(2,I)/10^(8)).
325.5 Set M(3)=ip(F(3,I)+M(1)*(T-F(4,I)))/(T-W(1)).
325.52 Line.
325.53 Type I in form 19.
325.54 Line.
325.6 Type M(3) in form 20.
325.7 Type M(2) in form 21.
325.8 Type M(1) in form 22.
325.85 Recall item 20 (rpts) if I=1.
325.9 Do part 12 if $M(1) \neq 0$.
326.04 Delete A.
326.05 Do step 326.5 for $E=1(1)12$.
326.06 Let A be sparse.
326.1 Do step 326.4 for $E=1(1)12$.
326.20 Set $A(13)=I$.
326.22 Set $A(14)=J$.
326.24 Set $I=A(11)$.
326.26 Set $J=A(12)$.
326.3 Type $A(1), A(2), A(3), A(4), A(5), A(6), A(7), A(8), A(9), A(10), I, J$ in form 15.
326.32 Set $I=A(13)$.
326.34 Set $J=A(14)$.
326.36 Done.
326.4 Set $A(E)=M(3+12\cdot(J-1)+E)$ if $12\cdot(J-1)+E \leq M(1)$.
326.5 Set $A(E)=-1$.
329.01 Page.
329.02 Type form 4.
329.03 Type ___timer.
329.04 Line.
329.05 Type $W(6)$ in form 5 if $W(6) \neq 0$.
329.06 Set $W(0)=0$.
329.07 Type $W(1)$ in form 6.
329.08 Type $T$ in form 7.
329.09 Type ___.
329.10 To step 329.13 if $W(5)=0$.
329.11 Do part 320 for $I=1(1)W(5)$.
329.12 Type ___.
329.13 To step 329.245 if $W(3) \leq 0$.
329.14 Type form 8.
329.15 Line.
329.16 Type form 9.
329.17 Line.
329.18 Do part 321 for $I=1(1)W(3)$.
329.185 Delete form 4, form 5, form 6, form 7, form 8, form 9, form 10, form 11.
329.19 To step 329.25 if $M(-9) \neq 1$.
329.20 Type ___.
329.21 Type form 12.
329.22 Line.
329.23 Type form 13.
329.24 Do part 322 for $I=1(1)W(3)$.
329.244 Line.
329.25 To step 329.315 if $W(4) \leq 0$.
329.26 Type ___.
329.27 Type form 16.
329.28 Line.
329.29 Type form 17.
329.30 Line.
329.31 Do part 324 for $I=1(1)W(4)$.
329.315 Delete part 320, part 321, part 322, part 323, part 324.
329.316 Delete form 12, form 13, form 14, form 15, form 16, form 17.
329.32 Do part 325 for $I=1(1)W(2)$.
329.33 Type ___timer.
Page 34
Set A(1)=0.
Delete M_A.
Set A(1)=0.
Set M(1)=0.
Let M be sparse.
Let A be sparse.
Delete part 325, part 326.
Delete form 18, form 19, form 20, form 21, form 22, form 26.
Do step 329.99 for I=329.01(.01)329.43.
Do step 329.99 for I=329.315, 329.3185, 329.316, 329.244, 329.44.
Delete step I.

Form 1:
J  P  S  1  2  3  4  5  6

Form 2:

Form 4:
** SUMMARY REPORT **

Form 5:
RANDOM NUMBER SEED = ____________

Form 6:
START TIME = ____________

Form 7:
SUMMARY REPORT PRINTED AT ____________

Form 8:
** STATISTICS COLLECTED IN PART 6 **

Form 9:
VARIABLE NO. OBS. AVERAGE STD. DEV. MAX.

Form 10:
X(____) ____________ ____________ ____________ ____________

Form 11:
PARAMETER SET _______ _______ _______ _______ _______ _______

Form 12:
** HISTOGRAMS **

Form 13:
VAR. LL(2) WID 1 2 3 4 5 6 7 8 9 10 11 12

Form 14:
X(____) ____________

Form 15:
Form 16: **STATISTICS COLLECTED IN PART 7**

Form 17:
VARIABLE        END TIME    AVERAGE    STD. DEV.    MAX VALUE

Form 18:
Y(____)        ..........    ..........    ..........    ..........

Form 19: **PRINTOUT FOR SET ___**

Form 20:
AVERAGE NUMBER IN SET WAS ___

Form 21:
MAXIMUM NUMBER IN SET WAS ___

Form 22:
CURRENT NUMBER IN SET IS ___

Form 26:
X(____)        No Values Observed

Form 28:
Attributes of Entries in Set ___

Form 29:
No Entries in set ___

Type size, size = 752
Delete all.
Recall item 24 (dist).
Done.
Type all.

16.1 Do part 15.
16.2 Set \( V(I) = B(2, I) + (B(3, I) - B(2, I)) \cdot Z \).

17.1 Set \( V(I) = 0 \).
17.2 Do part 331, 12 times.
17.3 Set \( V(I) = B(4, I) \cdot (V(I) - 6) + B(1, I) \).
17.4 Do part 327.

18.1 Do part 17.
18.2 Set \( V(I) = \exp(V(I)) \).

19.1 Set \( M(2) = 1 \).
19.2 Do part 328, \( B(4, I) \) times.
19.3 Set \( V(I) = -B(1, I) \cdot \log(M(2)) \).
19.4 Do part 327.

327.1 Set \( V(I) = B(2, I) \text{ if } V(I) < B(2, I) \).
327.2 Set \( V(I) = B(3, I) \text{ if } V(I) > B(3, I) \).

328.1 Do part 15.
328.2 Set \( M(2) = M(2) \cdot Z \).

331.1 Do part 15.
331.2 Set \( V(I) = V(I) + Z \).

Type size.

\[
\text{size} = 95
\]
REFERENCES


4. Miller, L. W., and B. C. Markowitz, VIMCOS II: A Scheduling Machine Simulation on JOSS, The RAND Corporation (to be published.)