PROGRAMMING BY QUESTIONNAIRE

Allen S. Ginsberg, Harry M. Markowitz and Paula M. Oldfather

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND
PROGRAMMING BY QUESTIONNAIRE

Allen S. Ginsberg, Harry M. Markowitz and Paula M. Oldfather

This research is sponsored by the United States Air Force under Project RAND—Contract No. AF 49(638)-700 monitored by the Directorate of Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

DDC AVAILABILITY NOTICE
Qualified requesters may obtain copies of this report from the Defense Documentation Center (DDC).
PREFACE

RAND has played an active role in the use of digital computers for simulation studies. In particular, the Logistics Department has constructed some large simulations of portions of the Air Force Logistics System. From this work, it has become clear that great amounts of effort and time are required to prepare the necessary computer programs. As an outgrowth of this work, a technique for reducing this effort and program preparation time has been developed. This technique, "Programming by Questionnaire" (or the Program Generation concept), allows a user to obtain a simulation program by filling out an English language questionnaire. This approach has sufficient generality that it promises application to areas of computer programming other than simulation.

This Memorandum describes the questionnaire technique, compares it to existing techniques, and discusses potential applications. The workings of the technique are described in terms of the Job Shop Simulation Program Generator, an example developed to test the feasibility and desirability of the concept.

Programming by questionnaire should be of interest to all those concerned with developing major computer programs. Within the Air Force, particular interest may be found at Headquarters USAF and AFLC because of their interests in simulation, and at ESD because of their general concern with computer usage.
SUMMARY

This Memorandum discusses a technique for reducing the cost and time required to produce computer programs within specified areas of application. The approach is to present the user with a set of possible options, expressed in an English language questionnaire. By choosing from these options, the user specifies all the information necessary to construct his desired computer program. A "Generator" program, informed of the selected options, then constructs the desired program and specifies the data that the user must supply in order to execute the generated program.

The components of the Generator program are:

(1) A "Statement List" that includes all the commands which may be needed in the construction of any of the programs.

(2) A set of "Decision Tables" that specify which commands are to be included in the generated program, based on the choices selected from the Questionnaire.

(3) An "Editor" that processes the Questionnaire, Statement List, and Decision Tables.

A number of advantages accrue from this particular method of program generation:

(1) Use of an English language questionnaire, requiring no knowledge of a detailed programming language or format specification.

(2) Generation of computer programs that are efficient in their usage of computer time and memory space (relative to other techniques).

(3) Ability to provide a very large range of options in any one Questionnaire.

The mechanics of the program generation process are discussed in terms of an illustrative Job Shop Simulator Program Generator (JSSPG). The Generator concept is not restricted to job shops or to simulations, however. Its application to specific areas which require a relatively large number of different computer programs appears promising.

In conclusion, the Memorandum treats other possible application areas, as well as the techniques for constructing and modifying generators. In addition, the Generator concept is compared to existing techniques. Work is continuing on solving the two outstanding difficulties of the Generator techniques: assuring correctness of generated programs and developing flexible methods of presenting results.
CONTENTS

PREFACE ................................................................. iii

SUMMARY ............................................................... v

Section
I. INTRODUCTION .................................................... 1

II. THE MECHANICS OF PROGRAM GENERATION ....................... 3
   Using a Program Generator .................................. 3
   The Questionnaire ............................................. 4
   The Statement List ............................................. 6
   Decision Tables ............................................... 8
   The Editor ................................................... 11

III. OBSERVATIONS AND DISCUSSION .................................. 13
   Other Techniques ............................................ 13
   Extensions of the Program Generation Concept ................. 15
   Changing the Generator and Modifying Generated
   Programs .................................................. 16
   Variants of the Questionnaire .............................. 18
   Difficulties Yet Unsolved ................................... 18
   Conclusion ................................................. 19

Appendix
A. QUESTIONNAIRE .................................................. 20
B. SAMPLE STATEMENT LIST AND DECISION TABLES .............. 38
I. INTRODUCTION

The time and effort required to produce a computer program have always been one of the major impediments to the practical application of computers. In the field of simulation, for example, the time required to build a model is critical since the program must be readied, the analysis run, and the conclusions drawn in time to affect a time-dependent decision. Considerable progress has been made in reducing programming time and cost by the development of advanced programming languages such as FORTRAN, COBOL, SIMSCRIPT, etc. The main objective of the techniques discussed here is to further reduce the effort required to produce large computer programs within specified areas. The purpose of this Memorandum is twofold: to explain the technique, and to discuss its merits, problems, and future implications.

The Questionnaire approach presents the analyst with a set of options centered around a basic model or technique. By choosing from these options, expressed in English, the user specifies all the information necessary to construct his desired computer program. A "Generator" program, informed of all the chosen options, then constructs the desired program. We characterize this approach as "Programming by Questionnaire," or the Program Generation concept.

The user of a Program Generator specifies the characteristics of the desired program by answering a questionnaire consisting of a set of multiple-choice questions. Submitting these answers to a computer, along with a Program Generator, results in the generation of a program whose logic is described by the options chosen on the Questionnaire. The user need not be aware of the internal parts of a Program Generator -- the Editor, a Statement List, and a set of Decision Tables.

In Sec. II, the workings of the Program Generation technique are explained. Wherever the case for ease of explanation is served, we will refer to a specific example, the Job Shop Simulation Program Generator.


(JSSPG), which the authors developed to demonstrate the feasibility of the Program Generator approach and to illustrate its mechanisms. Since the JSSPG is primarily a demonstration vehicle, it does not have some features that are likely to be found in real life shops, such as lot splitting, lap phasing, and alternate routings. The JSSPG will be documented fully in a future Memorandum. The only information presented here regarding the JSSPG is the complete Questionnaire.

The Program Generation concept is not limited to job shops or to simulations, but has potential application to other areas of computer usage. Sec. III will discuss this further. We have delayed our general comments and discussion of the Program Generation concept until Sec. III in order that the reader have an understanding of the workings of the technique.
II. THE MECHANICS OF PROGRAM GENERATION

USING A PROGRAM GENERATOR

The user of a Program Generator need concern himself only with the Questionnaire and the input data he specifies for executing the generated program. It is not necessary for him to know anything of the inner workings of the Generator. He need only understand what it is he wishes to model, answer the Questionnaire accordingly, and then, once the program is generated, supply the input data as indicated by the Generator.

For example, a user interested in job shop models specifies which of the statements on the Questionnaire are applicable to the particular model he wants to generate. These statements -- about 140 of them in our JSSPG -- concern the nature of arrivals, routing and process times of jobs, decision rules for job dispatch and resource selection, the existence of shift changes, absenteeism, etc. The answers to these questions, when given to the computer, which contains the Statement List and Decision Tables of the JSSPG and the Editor, result in a job shop simulation computer program. Also generated is a description of the numerical data which the user must supply to the generated program to quantify the characteristics of the shop.

A large number of different job shops can be generated in this manner. While there are not a full $2^{140}$ different models, since not every possible combination of answers is permissible, the number of possible models is at least $2^{30}$, or over one billion models. The work required to develop the JSSPG, however, was comparable to the effort required to build a few large models. We avoided the combinatorial problem by bringing together a number of devices, which, taken with the Questionnaire, form the Program Generator. The devices are:

(a) A "Statement List" which is a set of computer commands and any other statements which may be needed in the construction of any of the many programs.

(b) A set of "Decision Tables" that specify which commands are to be included in the generated program as a function of the answers to the Questionnaire.

(c) An "Editor" that processes the Questionnaire, Statement List, and Decision Tables, thus building the desired program and providing
a list of the input data the user must supply in order to use the program.

We will describe each of these components in the following paragraphs.

THE QUESTIONNAIRE

The Questionnaire is the key element of the Program Generator. It completely defines the scope and structure of all models that can be constructed. The Questionnaire for the JSSPG is shown in Appendix A starting on p. 21. It consists of two parts: the questions and an answer sheet. The user responds to the questions by marking the answer sheet, which is keyed to the questions. The answers describe the basic structure of the desired model. At the same time, the user selects the various probability distributions (which describe the form of the statistical behavior of various parts of the model) from another section on the same answer sheet, as directed by instructions on the question sheet. Punched cards prepared directly from this sheet form the input to the computer run which generates the specified computer program.

For ease of understanding, the JSSPG Questionnaire is divided into several sections. Section A, Resource Description, asks the user to specify some of the principal characteristics of the shop itself, such as the shift configuration, the resource availability on each shift, and whether one or two types of resources (i.e., men and/or machines) are used to process the jobs. Section B requires specification of the characteristics of jobs to be processed in the shop, such as frequency of arrival at the shop, routing through the shop, quantity, and process times. Job characteristics required for use in decision rules and analysis are chosen in Section C (e.g., ratio of estimated processing time to actual processing time, due date, value, and cost). Section D presents the options for decision rules such as priority rules for dispatching jobs, and rules for selecting combinations of resources to process jobs. Section E provides a choice of the frequency and types of analysis of results. Section F, which lists the types of probability distributions that may be assigned to certain variables (e.g., arrival frequency, processing time, and quantities), is included for clarification purposes only, since the instructions in Sections A to E have already directed the user to fill in Section F on the answer sheet. In Section G, the
user is asked to supply a small amount of numerical data such as the number of different types of resources, and the number of shifts per day.

The listing of probability distributions in Section F illustrates an important feature of the Program Generator. If the Questionnaire does not provide the particular distribution a user wishes, he has the option of obtaining any distribution by writing his own subprogram. For example, even though the Questionnaire does not provide for the quantity of items per lot to be distributed log normally, the user can achieve a log normal quantity distribution by providing his own subprogram.

One important characteristic of the Program Generator concept becomes evident upon reading the Questionnaire: it asks the user to specify only the basic structure of the model he desires. It asks for only a small amount of the numerical data necessary to completely describe the model. The user supplies the remainder of the data when he is ready to execute the program that the Generator constructs. As the data required will vary, depending on which options are selected on the Questionnaire, the Generator specifies exactly what data is needed for execution of any particular program (see "The Statement List," p. 6 for details).

An example of this characteristic of the Questionnaire is the description of shifts. The user can specify, for instance, that he wants every day in his simulation to be the same, each day having more than one shift. While the number of shifts per day are specified on the Questionnaire, their starting times and resource disposition are not, but rather are part of the input data that the user is asked to supply when executing the program. Thus, the user may change the number of resources available on each shift or their position within a day from one run to another by changing the input data without having to return to the Questionnaire.

Another example is the arrival of jobs at the shop. The user chooses only the mechanism for determining the time between job arrivals such as "each type of job has its own probability distribution, each type of job having the same form of distribution" (e.g., exponential, uniform, etc.). If he chooses the option stating that the
different types of jobs have different forms of the distribution, at execution time, he must supply data showing the distribution form associated with each type of job. In either case, the parameters of the distributions are supplied as data when executing the program.

The only numerical data the user must supply prior to construction of a program by the Generator are the dimensions of certain "permanent entities" as defined in the SIMSCRIPT language. In the JSSPG, this amounts to specifying the number of "primary resources" (machine groups, for instance), and if applicable, the number of "secondary resources" (e.g., men), the number of different types of jobs, and other information regarding shifts. In order to change these characteristics of the model, it is not necessary for the user to reconstruct the entire program, but only to change Section G of the Questionnaire where these values are listed. He then obtains only the SIMSCRIPT "Initialization"** portion of the program, rather than an entire program.

THE STATEMENT LIST

The Statement List is the "menu" from which the generated program is constructed. It consists of all the commands required to build all of the programs that can be described on the Questionnaire. The list also contains any other information the programming system requires, such as definitions of variables and initialization data required by the SIMSCRIPT language. All generated programs contain some subset of the Statement List. A sample routine from the JSSPG is shown starting on p. 39 of Appendix B, accompanied by its corresponding Decision Tables.

As suggested above, the Statement List can be written in any computer language from direct machine language to higher languages such as COBOL and SIMSCRIPT. The effort involved in building a Program Generator is, to a large extent, a function of the length and complexity of the Statement List. Therefore, it is easier to build a Generator using higher languages because of their simplicity, flexibility, and the need for fewer commands to accomplish a given task. The language in the resulting generated program will, of course, be the same one used in the Statement List (i.e., if Fortran is used in the Statement List, a Fortran program will be generated).

---

*Markowitz, Hausner, Karr, SIMSCRIPT, p. 3.

**Ibid., p. 115.
The commands in the Statement List look exactly as they would in any other program, except for their "identification numbers." If a command has no identification number, it is considered part of the preceding command that has an identification number. Thus, whole groups of commands can be called from the Statement List by simply selecting one identification number. A Statement List command may also differ from a normal command since it may be only part of a complete command. For example, when constructing a Statement List in the SIMSCRIPT language, it is sometimes found that a given phrase (such as FIND FIRST) may be controlled by a number of different control phrases (such as FOR EACH JOB), depending on which options the user has chosen on the Questionnaire. Rather than repeat the phrase common to all the options it is convenient to give this portion of the total command its own identification number as well as numbering each of the modifiers. The Editor then combines the appropriate parts into a single command.

In constructing the Statement List, we found it convenient to organize it as we would the most complex of the generated programs. Thus, the Statement List for the JSSPG was divided into the "Exogenous Events," "Endogenous Events,"* and subroutines necessary to write a SIMSCRIPT version of complex job shop simulation. Each of these routines was written separately, an effort being made to include the proper logic and commands to satisfy all feasible combinations of answers to the Questionnaire.

In addition to commands, the Statement List contains other information necessary to run the generated program. In the SIMSCRIPT language this includes the Definition Cards (which specify the variables used in the program) and the Initialization Cards (which specify the array sizes in memory).** Also, the user must know what data he is required to supply in order to use the generated program. All three of these components, the Definition and Initialization Cards and the Input Data Requirements, are lines in the Statement List, just as are the commands, and are selected, based on the user's answers to the Questionnaire, in the same manner as the commands.

---

* Markowitz, Hausner, Karr, SIMSCRIPT, p. 66.

** Ibid., pp. 103, 115.
DECISION TABLES*

The Decision Tables are the means used by the Editor to decide, based on the answers to the Questionnaire, which lines of the Statement List to include in the generated program. In building the Statement List, the programmer has in his mind the various combinations of commands necessary to satisfy the options chosen on the Questionnaire. The Decision Tables are merely a formal statement of these relationships. They give the Editor a set of rules, in a standardized manner, so it can make the proper choices.

Figures 1a and 1b illustrate the use of Decision Tables in the Program Generator scheme. Figure 1a shows the names usually attached to the four quadrants of a Decision Table. The entries in the condition stub are the numbers given to the questions on the Questionnaire, and the entries in the action stub are the identification numbers given to the lines in the Statement List. On the right-hand side of the vertical double line are the conditions and actions. The condition entries are the answers to the questions listed in the condition stub.

Each column in the conditions section represents a single combination of feasible answers to the questions. For example, in the hypothetical decision table in Fig. 1b, the three questions, Q1, Q2, and Q3, may be answered, in combination, any one of four ways. (There are only four combinations because the nature of the questions precludes the other four in this example.) The first column of three "N's" would apply if the user had not chosen any of the three questions, or, in effect, had answered them all "No." The second column indicates a "Yes" to Q1, a "No" to Q3, and indifference to Q2. The third column illustrates a peculiarity of the Decision Tables as applied in this manner; for the purpose of simplifying the tables, we found it convenient to make the order of the columns meaningful, in the sense that the first column found to match the answers on the Questionnaire, when scanning from left to right in the table, will be the column that applies. Thus, the third column appears to imply indifference to Q3; it actually will be chosen only if all three answers are "Yes" (i.e., Y, Y, Y) since the combination Y, Y, N, will cause the second column to be chosen.

### Fig. 1a -- Decision Table Terminology

<table>
<thead>
<tr>
<th>CONDITION STUB</th>
<th>ACTIONS STUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Question Nos.)</td>
<td>(Statement Nos.)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CONDITIONS</th>
<th>ACTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Answers to Questions)</td>
<td>(Selected Statements)</td>
</tr>
</tbody>
</table>

### Fig. 1b -- Sample Decision Table

<table>
<thead>
<tr>
<th>Q1</th>
<th>Y</th>
<th>Y</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q2</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Q3</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>16</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>105</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+536</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
When a column is found to match the answers on the Questionnaire, the actions to be taken are found in the same column in the action portion of the table. For example, if only Q1 is selected on the Questionnaire, then the action taken by the Editor is to include the statements numbered 16, 105, and 536 in the generated program.

Two other features of the use of the Decision Tables are illustrated by Fig. 1b. The "4" in front of the number 536 tells the Editor that this statement is not a regular command but is, instead, a special line of information such as a definition or initialization card (in this case, a definition card), or any other information needed in addition to the regular program commands. The other feature is illustrated by the last column in the table. This column will be reached only if none of the feasible combinations of answers to the questions has been realized, which means the user has made an error in filling out the Questionnaire. The Editor, upon finding both condition and action portions blank, would then write an error message, telling the user where he had erred.

The decision as to how extensive a single Decision Table should be arose a number of times during the construction of the JSSFG. Conceivably each Decision Table could develop, say, one complete routine from the Statement List. For routines of nontrivial length, however, this is a bad practice for two reasons. The complexity in constructing a large table (many entries in the condition stub) goes up much more rapidly than does the size of the table; and the Editor will usually execute its function more rapidly with smaller Decision Tables. This latter feature arises because the execution time will be largely a function of the number of comparisons that must be made in the condition portion of the table. Thus, the smaller the volume of this portion of the tables, the faster the Editor will operate. Experience has shown that breaking one large table into a number of smaller ones, where convenient, will usually reduce the over-all volume of comparisons, even though some of the question numbers must be repeated in the condition stub portion of the smaller tables. In the JSSFG, we frequently found it economical to let a Decision Table control the selection of only one or two lines of the Statement List. An illustration of a set of Decision Tables used in the JSSFG is shown on pp. 40-42 of Appendix B.
We have been speaking of the Editor's function in general terms. Specifically, the Editor is the computer program that translates the user's responses to the Questionnaire into the desired computer program. The current version of the Editor is written in SIMSCRIPT, though any language could have been used. It treats as input data the three parts of the Generator: answers to the Questionnaire, the Statement List, and the Decision Tables. Even though the content of these three inputs will be different for Program Generators other than the JSSPG, their formats and logic of construction will be the same. Thus, the current version of the Editor can be used for other Program Generators in any area of application, if the language used for the Statement List is SIMSCRIPT, and if the principal components are constructed in the manner outlined herein.

The functions of the Editor are multi-fold:

(1) To translate the answers to the Questionnaire into a computer program.

(2) To supply all control cards and dictionary information necessary to execute the above generated program.

(3) To provide a list of the names, meanings and dimensions of the variables whose values must be supplied to the generated program by the user prior to execution.

(4) To check the answers to the Questionnaire for completeness and consistency.

In addition to a printed listing of the program, the Editor supplies a corresponding deck of cards containing all the required control cards. This deck may be resubmitted, as is, for a SIMSCRIPT compilation. If the user also wishes to execute the program, he need only place the required input data at the back of the deck provided.

Execution time and memory space requirements for the Generator itself are moderate. The generation of relatively large programs has been accomplished in less than three minutes on an IBM 7044 computer. Memory space is conserved by reading in the Statement List and the corresponding Decision Tables in independent blocks. Independent blocks are formed by associating a set of Decision Tables with a portion of the Statement List. The blocks are made independent by assuming that the entries in the action stubs of the Decision Tables call
only for statement numbers contained in the associated portion of the Statement List.
III. OBSERVATIONS AND DISCUSSION

In this section we discuss some of the attributes of the Program Generation concept. Included is a comparison to other methods aimed at the same objective, suggestions for application to other areas, comments on building generators, and on modifying both generators and generated programs. We will also note some of the problems of the technique as well as some of its favorable features.

OTHER TECHNIQUES

The idea of automatic preparation of computer programs is not new. Other approaches, aimed at reducing programming costs, have been suggested. We will discuss a few of these approaches that have received the greatest attention, pointing out their shortcomings relative to the Program Generator approach.

In one of these techniques, which we shall call the "Modular" approach, the user specifies and builds the desired program by selecting and fitting together a number of pre-programmed or "canned" subroutines. This approach has proven difficult to implement in most situations because of the difficulties in making the various subroutines logically compatible and/or because of the very large library of subroutines necessary to make the method work. If the set of options presented to a user is relatively small, such as in a sort program generator, the modular approach may prove feasible. Because the Program Generator uses decision tables and compiles programs from individual commands, rather than from larger modules, it alleviates most of the above difficulties, allowing the user a much wider range of options.

The "Generalized Model" approach uses one large program containing all possible options; the options to be executed in a given run are specified by parameter assignment in the input data.** The principal

---

* See, for example, IBM 7090 Generalized Sorting System, IBM Systems Library, File No. 7090-33.

** One illustration: The Job Shop Simulator, Mathematics and Applications Department, Data Systems Division, IBM.
difficulty with this method is the inefficient use of computer time and memory space. In order to decide which options to perform, the program must continually interrogate the parameters the user specified, using the result of the interrogation to decide which parts of the program are to be performed. If the program contains a large number of options, the time spent in deciding which ones to execute will be a significant portion of the total running time. The Program Generator does not suffer from this difficulty since it goes through the process of choosing the options only once, at the time the program is generated.

The inefficient use of computer memory space in the general-type program occurs because all the options and the logic necessary to choose from amongst them must reside in the computer memory during the program's execution. Therefore, memory size places a limit on the number of options that can be made available. Since a Program Generator constructs only the code necessary to execute the chosen options, the generality of the approach is usually not limited by available memory space.

Program generators are not new. An example of a previous generator is the GQS.* As a consequence of how the user specifies his desired model and of the method used to generate the program, the range of options that can be offered in any one such compiler is very limited, as compared to the Questionnaire method.

In summary, the features of Programming by Questionnaire which distinguish it from other methods of program generation are:

1. An English language questionnaire, requiring no special knowledge of either detailed programming languages or format specifications.

2. Generation of computer programs which are as efficient as possible (given the limitations of both the language used for the Statement List and the person building the generator) in terms of computer memory space and computer time.

3. The ability to provide an almost unlimited range of options in any one generator.

EXTENSIONS OF THE PROGRAM GENERATION CONCEPT

As mentioned previously, the motivation for developing the Generator technique came from our work in the area of simulation. Now that the concept has been developed into something workable, it is clear that the technique is not necessarily limited to simulations. It is not immediately evident, however, about "where" (i.e., which problem areas) or "when" to apply the Generator technique (i.e., as opposed to the use of other techniques or the usual practice of writing the program for the specific application).

An answer to the "where" question lies in the definition of the term "problem area." It is obvious from the description of the workings of the Generator that no one questionnaire can cover all possible computer programs. As in the case of the JSSPG, it must be limited to some fairly specific area. This problem area is characterized by a basic structure or model with many possible variations. In the area of job shop simulations, for example, the basic model we envisioned was a facility with a fixed set of resources, with jobs arriving at the shop at some interval, and with each job having a set of instructions attached to it telling which resources are to process the job at each of its stages. The alternatives superimposed on this basic model were such things as choices of shift configuration, method of job arrivals, job parameters (type, quantity, routing process times, etc.), queue disciplines, and resource selection rules. Even though these and other alternatives may cause various job shop models to look quite different, they all have a basic structure in common, where the commonality might be expressed as a "queueing network," or as a "task-resource system." If it is possible to model an area in this manner, then the Generator concept is likely to be applicable. Examples of some of these areas are given later in this section.

In order to answer the question about "when" to apply the proposed Generator technique as opposed to other alternatives, two aspects to consider are the frequency of usage and the range of options available. If a generator in a particular area, such as the JSSPG, is to be utilized in only one or two applications, clearly it is more economical to write the desired programs as needed. On the other hand, more frequent
utilization, say twenty-five or more applications, points toward automatic generation techniques. It is very difficult to locate the cross-over point even if the usage and costs could be accurately projected, since the existence of a generator may have some non-measurable consequences. For example, the user who has a generator at his disposal is likely to try experiments that he otherwise might forego for reasons of lack of time or funds. Thus, the decision about whether or not to supply some form of automatic programming is now, and is likely to remain, a largely subjective evaluation. Likewise, the decision regarding which of the available generation techniques to apply is largely qualitative, since it depends to a large extent on the range of options to be presented to the user. As pointed out earlier, the Questionnaire technique is particularly suitable when the number of options is large. The generalized or modular techniques may be easier to apply when the range of options is small, but by the nature of their mechanisms will almost always result in programs less efficient in use of computer space and time.

From the nature of the Questionnaire and the method used to construct programs, it seems that the Program Generation concept could be profitably applied to many problem areas. In the general field of simulation, there are a number of areas which appear to have both the wide user interest and the wide range of options necessary to justify the construction of a Program Generator. Some of these areas are urban transportation, communication networks, inventory systems, and data processing facilities. Many of the logistics systems of the Air Force could also be modeled in this manner. Outside the area of simulation, one possible application is the area of data reduction. Many organizations store in machine-readable form large volumes of data that they wish to analyze and report in diverse ways. A generator which builds special purpose data retrieval and/or data reduction programs might be feasible and useful. Certain everyday data processing jobs, such as inventory control, payrolls, etc., also appear to be attractive candidates.
CHANGING THE GENERATOR AND MODIFYING GENERATED PROGRAMS

In order to apply the Generator concept to any new problem area, the Questionnaire, the Decision Tables and the Statement List must be written for that specific area. The Editor, as it now exists, need not be rewritten, since it treats the above three elements as data. As long as these three elements of the Generator are constructed as described herein, the Editor will properly perform its function of building a program from the Questionnaire answers.

Constructing a new generator is a fairly difficult task. The builder must know enough about the problem area not only to construct the basic model, but also to anticipate enough of the important options to make the resulting generator useful. At this point in the development of the technique, we cannot recommend the best way to go about building a new generator. In the case of the JSSPG, we first wrote the Questionnaire and from this, constructed the Statement List and Decision Tables. Because the authors had extensive experience in job shop simulations, it is probably not fair to say this is the best procedure, even though it worked very well. It has been pointed out that a reasonable alternative to this procedure is to first build a very large program containing many of the desired options and distill from this the components of a generator.

The usefulness of the Generator concept can also be extended by adding to a given Questionnaire or modifying generated programs. If it becomes clear that some number of desired options have been omitted from a Questionnaire for any particular problem area, it is possible to add this option. The appropriate changes must be made to the three variable elements of the Generator. An alternative to adding to the Generator is to modify the generated program. That is, in cases where the generator can provide a program which is almost, but not quite, what the user wants, he can make the necessary changes to the generated program. Again, it is a fairly difficult task to extend a given generator with either method. However, the orderliness introduced by the Decision Tables and the use of an easily understood language, such as SIMSCRIPT, for the commands in the Statement List, make these kinds of changes entirely feasible.
VARIANTS OF THE QUESTIONNAIRE

Although we have not tried them in practice, we have discussed two variations that can be applied to the Questionnaire. The first concerns the form of the questions. In the JSSPG, the questions are "multiple choice" type, though they could have been "fill-in" type. With fill-in type questions, the Questionnaire would be shorter and simpler, but the Editor would be more complex. The other variant is in the form of presenting the Questionnaire. In the JSSPG, the user must follow various directions which instruct him to answer certain questions, to skip over others, etc., as a function of his answers. If the Questionnaire were stored in a computer, it could be presented to the user on a console, such as a typewriter, question by question, in the proper order. As the user supplied the answers, the computer would choose and present the appropriate questions. Upon completion of the Questionnaire, the computer would compile the desired program, and execute it if directed. This kind of on-line dialogue between user and computer can be a very effective experimental tool. No longer is there a complex programming language or a long time span separating the user from answers to certain kinds of problems.

An example of how the on-line dialogue might be a useful technique is found in the design of computer models of complex systems. The ease and rapidity with which the design of a model might be varied or the decision rules changed might encourage the user to perform sensitivity tests with the model that he might otherwise be reluctant to undertake. Particularly in the case of simulation models, this kind of experimentation can be extremely important and revealing.

DIFFICULTIES YET UNSOLVED

Two principal difficulties with the Program Generator are not yet completely solved. The difficulties are in the area of results presentation and debugging of a generator. Because of the very large number of ways to analyze and summarize raw data and to present the results, it is very difficult to supply a complete and flexible set
of options for analysis results. In the JSSPG, the user is supplied with a limited set of choices as to what results will be computed and given no choice as to their form of presentation. We simply made what choices we thought were necessary and useful for analysis of the outcomes. The user who is knowledgeable in computer programming can transcend this difficulty by use of an intermediate output of the JSSPG. As each event takes place during the simulation, a message is written on tape containing pertinent information (i.e., type of event, values of changed variables, time, etc.). A special program can be written to analyze this raw data and present the results as the user desires. This does not solve the problem, however, since the user is now required to write a program manually. More thought on how to enhance the analysis capabilities of the Generator concept is necessary.

The Generator is difficult to debug since it is very difficult to assure that the very large number of different programs that it can generate are all error free. In other words, it is hard to have complete assurance that the generator is fully debugged. Our present attempt to overcome this problem consists of careful construction of the Statement List and Decision Tables, and thorough checking of a large number of generated programs. The testing of these programs should assure that all options on the Questionnaire have been tried, at least independently of each other, along with as many of the combinations of options as practicable. Again, this approach is neither easy nor completely satisfying.

CONCLUSION

The JSSPG will be published in its entirety in the near future. In the meantime, it is hoped that this Memorandum will serve those who wish to contemplate the proposed technique, or offer suggestions and criticisms, or possibly try construction of their own Generator.

We feel that easy-to-use, flexible, and efficient problem-oriented techniques for the preparation of computer programs are important in broadening the scope of application of digital computers. We hope that the technique described herein will be a contribution to this development.
Appendix A

QUESTIONNAIRE
A. RESOURCE DESCRIPTION

IF ONLY THE INITIAL INPUT VALUES ARE TO BE CHANGED (I.E., ONLY A NEW INITIALIZATION DECK IS DESIRED, NOT AN ENTIRE NEW PROGRAM), COMPLETE SECTION G ONLY. OTHERWISE, CONTINUE HERE.

SHIFT CHANGE OPTIONS - ALL DAYS HAVE 24 HOURS. IF EMPTY SHIFTS ARE DESIRED, THEY MUST BE SPECIFIED.

CHOOSE ONE OF A1, A2, A3, OR A4

EVERY DAY IS THE SAME... A1. ONLY ONE SHIFT PER DAY, OF 24-HOUR DURATION. (IF A1 IS CHOSEN, FILL IN G2-G10)
A2. MORE THAN ONE SHIFT PER DAY. (IF A2 IS CHOSEN, FILL IN G2-G14)

NOT EVERY DAY IS THE SAME... A3. EVERY WEEK IS THE SAME. (IF A3 IS CHOSEN, FILL IN G2-G18)
A4. NOT EVERY WEEK IS THE SAME. (IF A4 IS CHOSEN, FILL IN G2-G22)

TYPE OF RESOURCES

CHOOSE ONE OF A5 OR A6

A5. A JOB IS PROCESSED BY ONE UNIT OF ONE RESOURCE AT EACH OF ITS STAGES. (IF A5 IS CHOSEN, FILL IN G23-G26, THEN GO TO A9)
A6. AT EACH OF ITS STAGES A JOB IS PROCESSED BY ONE UNIT OF A PRIMARY RESOURCE AND ONE UNIT OF ANY OF THE SECONDARY RESOURCES WHICH MAY SERVE THAT PRIMARY RESOURCE. (IF A6 IS CHOSEN, FILL IN G23-G30)

IF A6 WAS CHOSEN AND ONE OF A2, A3, OR A4 WAS CHOSEN, CHOOSE ONE OF A7 OR A8

THE NUMBER OF UNITS OF EACH SECONDARY RESOURCE PER SHIFT IS...
A7. NON-RANDOM
A8. RANDOM

IF A2, A3, OR A4 WAS CHOSEN, CHOOSE ONE OF A9 OR A10

THE NUMBER OF UNITS OF EACH PRIMARY RESOURCE PER SHIFT IS...
A9. NON-RANDOM
A10. RANDOM
B. JOB CHARACTERISTICS

TYPES OF JOBS

B1. EACH JOB IS ASSIGNED A JOB TYPE (REQUIRED ONLY IF ONE OR
MORE OF THE JOB CHARACTERISTICS LISTED BELOW, SUCH AS
FREQUENCY OF JOB ARRIVALS, PROCESS TIMES, ETC., DEPEND
ON JOB TYPE). IF B1 IS CHOSEN, FILL IN G31-G34.

EXOGENOUS INPUTS

CHOOSE EITHER B2 OR ONE OF B9-B11.

B2. JOB ARRIVALS ARE DETERMINED EXOGENOUSLY (I.E. EACH
ARRIVAL OCCURRENCE IS SPECIFIED INDIVIDUALLY AS INPUT DATA).

IF B2 WAS CHOSEN, CHOOSE ONE OF B3-B8. OTHERWISE, GO TO B9.

B3. THE SEQUENCE OF OPERATIONS PERFORMED ON A JOB AND THE
CORRESPONDING PROCESS TIMES ARE SPECIFIED EXOGENOUSLY.
IF B3 IS CHOSEN, THE DUE-DATE AND/OR DOLLAR VALUE CAN BE
MADE TO DEPEND ON THE TYPE OF THE JOB BY SPECIFYING THEM
EXOGENOUSLY (I.E. C6 AND/OR C9 MUST BE CHOSEN). IF B3
IS CHOSEN, GO TO C1.

B4. A TYPE AND A QUANTITY ARE SPECIFIED EXOGENOUSLY FOR EACH
JOB. (IF B4 IS CHOSEN, CHOOSE ONE OF B12-B15 FOR THE JOB
ROUTINGS, AND THEN GO TO B20)

B5. ONLY THE TYPE IS SPECIFIED EXOGENOUSLY FOR EACH JOB.
(IF B5 IS CHOSEN, GO TO B12)

B6. ONLY THE ARRIVAL OF THE JOB IS SPECIFIED EXOGENOUSLY.
THE TYPE OF JOB IS DETERMINED BY A RANDOM TABLE LOOK-UP.
(IF B6 IS CHOSEN, GO TO B12)

B7. THERE IS NO JOB TYPE (I.E., B1 WAS NOT CHOSEN). THE
QUANTITY IS SPECIFIED EXOGENOUSLY FOR EACH JOB. (IF B7
IS CHOSEN, CHOOSE EITHER B12 OR B14, AND THEN GO TO B20)

B8. THERE IS NO JOB TYPE (I.E., B1 WAS NOT CHOSEN). ONLY THE
ARRIVAL OF THE JOB IS SPECIFIED EXOGENOUSLY. (IF B8 IS
CHOSEN, CHOOSE EITHER B12 OR B14, AND THEN GO TO B16)

FREQUENCY OF JOB ARRIVALS

IF THERE ARE NO JOB TYPES (I.E., B1 WAS NOT CHOSEN), AND JOB
ARRIVALS ARE NOT EXOGENOUS (I.E., B2 WAS NOT CHOSEN), CHOOSE ONE
OF F1-F6 FOR THE FORM OF ALL INTER-ARRIVAL TIME DISTRIBUTIONS,
THEN GO TO B12. OTHERWISE, CHOOSE ONE OF B9-B11.
B9. THERE IS A DISTRIBUTION OF DELAY TIMES BETWEEN JOB ARRIVALS. WHEN AN ARRIVAL OCCURS, THE TYPE OF JOB IS DETERMINED BY A RANDOM TABLE LOOK-UP. (IF B9 IS CHOSEN, ALSO CHOOSE ONE OF F1-F6 FOR THE FORM OF INTER-ARRIVAL DISTRIBUTION)

B10. EACH TYPE OF JOB HAS ITS OWN PROBABILITY DISTRIBUTION OF INTER-ARRIVAL TIMES. THESE DISTRIBUTIONS ALL HAVE THE SAME FORM, BUT THEIR PARAMETERS MAY BE DIFFERENT. (IF B10 IS CHOSEN, ALSO CHOOSE ONE OF F1-F6 FOR THE FORM OF ALL INTER-ARRIVAL TIME DISTRIBUTIONS)

B11. EACH TYPE OF JOB HAS ITS OWN PROBABILITY DISTRIBUTION OF INTER-ARRIVAL TIMES. THESE DISTRIBUTIONS MAY HAVE DIFFERENT FORMS AND MAY HAVE DIFFERENT PARAMETERS.

JOB ROUTINGS

CHOOSE ONE OF B12-B15.

THE SEQUENCE OF OPERATIONS PERFORMED ON A JOB (OF A PARTICULAR TYPE) IS...

B12. FIXED (NON-RANDOM), DOES NOT DEPEND ON TYPE*
B13. FIXED (NON-RANDOM), DEPENDS ON TYPE
B14. RANDOM, DOES NOT DEPEND ON TYPE*
B15. RANDOM, DEPENDS ON TYPE
*IF B12 OR B14 IS CHOSEN, THE PROCESS TIMES CANNOT DEPEND ON THE TYPE OF THE JOB.

PROCESS TIMES

IF EACH NEW ORDER IS ASSIGNED A QUANTITY TO BE USED IN PROCESS TIME CALCULATIONS, GO TO B18. IF EACH ORDER IS NOT ASSIGNED A QUANTITY (I.E. PROCESS TIMES APPLY TO THE LOT AS A WHOLE), CONTINUE HERE. IF THERE ARE NO JOB TYPES (I.E., B1 WAS NOT CHOSEN), OR IF THE ROUTING DOES NOT DEPEND ON THE TYPE OF THE JOB (I.E., B13 OR B15 WAS CHOSEN), CHOOSE ONE OF F7-F14 FOR THE FORM OF ALL PROCESS TIME DISTRIBUTIONS. OTHERWISE, CHOOSE ONE OF B16 OR B17.

B16. EACH TYPE OF JOB HAS ITS OWN PROBABILITY DISTRIBUTION OF PROCESS TIMES PER LOT. THESE DISTRIBUTIONS ALL HAVE THE SAME FORM, BUT THEIR PARAMETERS MAY BE DIFFERENT. (IF B16 IS CHOSEN, ALSO CHOOSE ONE OF F7-F14 FOR THE FORM OF ALL PROCESS TIME DISTRIBUTIONS)

B17. EACH TYPE OF JOB HAS ITS OWN PROBABILITY DISTRIBUTION OF PROCESS TIMES PER LOT. THESE DISTRIBUTIONS MAY HAVE DIFFERENT FORMS AND MAY HAVE DIFFERENT PARAMETERS.

IF EACH NEW ORDER IS NOT ASSIGNED A QUANTITY, GO TO B24. IF THERE ARE NO JOB TYPES (I.E., B1 WAS NOT CHOSEN), CHOOSE ONE OF F15-F19 FOR THE FORM OF ALL QUANTITY DISTRIBUTIONS. OTHERWISE, CHOOSE ONE OF B18 OR B19.
B18. Each type of job has its own probability distribution of quantities. These distributions all have the same form, but their parameters may be different. (If B18 is chosen, also choose one of F15-F19 for the form of all quantity distributions)

B19. Each type of job has its own probability distribution of quantities. These distributions may have different forms and may have different parameters.

If there are no job types (i.e., B1 was not chosen), or if the routing does not depend on the type of the job (i.e., B12 or B14 was chosen), choose one of F20-F27 for the form of all set-up time distributions. Otherwise, choose one of B20-B21.

B20. Each type of job has its own probability distribution of set-up times per lot. These distributions all have the same form, but their parameters may be different. (If B20 is chosen, also choose one of F20-F27 for the form of all set-up time distributions)

B21. Each type of job has its own probability distribution of set-up times per lot. These distributions may have different forms and may have different parameters.

If there are no job types (i.e., B1 was not chosen), or if the routing does not depend on the type of the job (i.e., B12 or B14 was chosen), choose one of F28-F35 for the form of all process time distributions. Otherwise, choose one of B22 or B23.

B22. Each type of job has its own probability distribution of process times per unit. These distributions all have the same form, but their parameters may be different. (If B22 is chosen, also choose one of F28-F35 for the form of all process time distributions)

B23. Each type of job has its own probability distribution of process times per unit. These distributions may have different forms and may have different parameters.

If the total processing time of a job at any stage is multiplied by a random factor which is associated with the primary resource required at that stage, choose one of B24 or B25. Otherwise, go to B26.

B24. Each primary resource has its own probability distribution of the factor. These distributions all have the same form, but their parameters may be different. (If B24 is chosen, also choose one of F36-F39 for the form of all distributions of the factor)

B25. Each primary resource has its own probability distribution of the factor. These distributions may have different forms and may have different parameters.

B26. If jobs are processed by a primary and a secondary resource (i.e., A6 was chosen), then choosing this option indicates that each combination of primary and secondary resources has an efficiency factor associated with it, the total processing time to be multiplied by this factor.
C. JOB CHARACTERISTICS REQUIRED FOR DECISION RULES AND ANALYSIS

A NUMBER OF THE JOB SELECTION DECISION RULES LISTED BELOW USE PROCESSING TIME IN SELECTING A JOB. IF THE ESTIMATED PROCESSING TIME USED IN THE DECISION RULES IS DIFFERENT FROM THE ACTUAL PROCESSING TIME AS GENERATED ABOVE, SELECT ONE OF C1-C5. OTHERWISE, GO TO C6.

C1. THE FORM OF THE PROBABILITY DISTRIBUTION, AND ITS PARAMETERS, OF THE RATIO OF ESTIMATED TO ACTUAL PROCESSING TIMES IS THE SAME FOR ALL JOBS AND FOR ALL RESOURCES. (IF C1 IS CHOSEN, ALSO CHOOSE ONE OF F40-F44 FOR THE FORM OF THE DISTRIBUTION)

EACH TYPE OF JOB HAS ITS OWN PROBABILITY DISTRIBUTION OF THE RATIO OF ESTIMATED TO ACTUAL PROCESSING TIMES...

C2. THESE DISTRIBUTIONS ALL HAVE THE SAME FORM, BUT THEIR PARAMETERS MAY BE DIFFERENT. (IF C2 IS CHOSEN, ALSO CHOOSE ONE OF F40-F44 FOR THE FORM OF ALL DISTRIBUTIONS)

C3. THESE DISTRIBUTIONS MAY HAVE DIFFERENT FORMS AND MAY HAVE DIFFERENT PARAMETERS.

EACH TYPE OF PRIMARY RESOURCE HAS ITS OWN PROBABILITY DISTRIBUTION OF THE RATIO OF ESTIMATED TO ACTUAL PROCESSING TIMES...

C4. THESE DISTRIBUTIONS ALL HAVE THE SAME FORM, BUT THEIR PARAMETERS MAY BE DIFFERENT. (IF C4 IS CHOSEN, ALSO CHOOSE ONE OF F40-F44 FOR THE FORM OF ALL DISTRIBUTIONS)

C5. THESE DISTRIBUTIONS MAY HAVE DIFFERENT FORMS AND MAY HAVE DIFFERENT PARAMETERS.

THE 'DUE DATE', BY WHICH TIME A JOB SHOULD BE COMPLETED, IS REQUIRED BY SEVERAL OF THE JOB SELECTION DECISION RULES LISTED BELOW (D5, D9, D10). IT IS ALSO USED IN THE COMPUTATION OF JOB LATENESS STATISTICS. IF JOB STATISTICS ARE NOT DESIRED, AND IF THE JOB SELECTION DECISION RULE IS NOT ONE OF THE AFOREMENTIONED, GO TO C9. OTHERWISE, CONTINUE HERE.


THE DUE-DATE IS EQUAL TO THE TIME OF ARRIVAL PLUS A RANDOM INCREMENT. IF THERE ARE NO JOB TYPES OR IF THE INCREMENT DOES NOT DEPEND ON THE JOB TYPE, CHOOSE ONE OF F45-F51 FOR THE PROBABILITY DISTRIBUTION OF THE INCREMENT. OTHERWISE, CHOOSE ONE OF C7 OR C8 (NOT ALLOWED IF B3 WAS CHOSEN)

C7. EACH TYPE OF JOB HAS ITS OWN PROBABILITY DISTRIBUTION OF THE INCREMENT. THESE DISTRIBUTIONS ALL HAVE THE SAME FORM, BUT THEIR PARAMETERS MAY BE DIFFERENT. (IF C7 IS CHOSEN, ALSO CHOOSE ONE OF F45-F51 FOR THE FORM OF ALL DISTRIBUTIONS)

C8. EACH TYPE OF JOB HAS ITS OWN PROBABILITY DISTRIBUTION OF THE INCREMENT. THESE DISTRIBUTIONS MAY HAVE DIFFERENT FORMS AND MAY HAVE DIFFERENT PARAMETERS.
THE DOLLAR VALUE ASSIGNED TO A JOB IS REQUIRED BY ONE OF THE
JOB SELECTION DECISION RULES LISTED BELOW (D6). IT IS ALSO USED IN
THE COMPUTATION OF SOME OF THE ANALYSIS STATISTICS. IF JOB
STATISTICS ARE NOT DESIRED, AND IF THE JOB SELECTION DECISION RULE
SELECTED IS NOT ONE OF THE AFOREMENTIONED, GO TO D1. OTHERWISE,
CONTINUE HERE.

C9. THE DOLLAR VALUE IS READ IN EXOGENOUSLY (ONLY AVAILABLE

IF THERE ARE NO JOB TYPES (I.E., B1 WAS NOT CHOSEN), OR IF THE
DOLLAR VALUE DOES NOT DEPEND ON THE TYPE OF THE JOB, CHOOSE
ONE OF F52-F57 FOR THE FORM OF ALL DOLLAR VALUE DISTRIBUTIONS.
OTHERWISE, CHOOSE ONE OF C10 OR C11 (NOT ALLOWED IF B3 WAS CHOSEN).

C10. EACH TYPE OF JOB HAS ITS OWN PROBABILITY DISTRIBUTION OF
DOLLAR VALUE. THESE DISTRIBUTIONS ALL HAVE THE SAME FORM,
BUT THEIR PARAMETERS MAY BE DIFFERENT. (IF C10 IS CHOSEN,
ALSO CHOOSE ONE OF F52-F57 FOR THE FORM OF ALL DOLLAR VALUE
DISTRIBUTIONS)

C11. EACH TYPE OF JOB HAS ITS OWN PROBABILITY DISTRIBUTION OF
DOLLAR VALUE. THESE DISTRIBUTIONS MAY HAVE DIFFERENT
FORMS AND MAY HAVE DIFFERENT PARAMETERS.
D. DECISION RULES

SELECT ONE OF D1-D12 FOR THE RULE TO BE USED TO SELECT A JOB FROM
THE QUEUE OF A PRIMARY RESOURCE

D1. FIRST COME, FIRST SERVED
D2. SHORTEST PROCESSING TIME AT CURRENT STAGE
D3. LONGEST PROCESSING TIME AT CURRENT STAGE
D4. EARLIEST ARRIVAL AT SHOP
D5. EARLIEST DUE DATE OF JOB COMPLETION
D6. LARGEST VALUE
D7. RANDOM SELECTION FROM QUEUE
D8. A FLOATING POINT FUNCTION SUBPROGRAM, CALLED 'PRITY',
    WILL BE SUPPLIED BY THE USER. 'PRITY' HAS ONE ARGUMENT,
    THE JOB. SMALLER VALUES OF 'PRITY' HAVE PRECEDENCE.

D9-D15 PERMITTED ONLY IF FIXED ROUTING (I.E., B12 OR B13 WAS
CHOSEN), OR EXOGENOUS ROUTING (I.E., B3 WAS CHOSEN).

SMALLEST SLACK WHERE SLACK = FINAL DUE DATE - CURRENT TIME -
'BACKOFF' WHERE...

D9. 'BACKOFF' = (A)(NO. OF REMAINING OPERATIONS) + (B)(SUM OF
    REMAINING PROCESSING TIMES)

D10. 'BACKOFF' = A + (B)(PROCESS TIME AT CURRENT OPERATION) + C
    + (D)(PROCESS TIME AT NEXT OPERATION) + E + (F)(PROCESS
    TIME AT FOLLOWING OPERATION) +......+ Y + (Z)(PROCESS TIME
    AT LAST OPERATION), WHERE THE SUM IS OVER ALL REMAINING
    OPERATIONS AND THE A, B,......, Z DEPEND ON THE PRIMARY
    RESOURCE TO BE USED AT THE OPERATION.

LOOK AHEAD TO THE QUEUE OF THE SUBSEQUENT RESOURCE USED BY THE
JOB. AMONG THOSE JOBS WHICH HAVE A SUBSEQUENT OPERATION, THE JOB
TO BE PROCESSED IS THE ONE WITH THE SMALLEST...

D11. NUMBER OF JOBS ALREADY IN THE SUBSEQUENT QUEUE AND DUE TO
    ARRIVE AT THIS QUEUE AT THE COMPLETION OF THEIR CURRENT
    OPERATION

D12. AS IN D11, BUT WITH TOTAL HOURS OF WORK (I.E., BACKLOG)
    INSTEAD OF NUMBER OF JOBS

IF A LOOK-AHEAD RULE WAS CHOSEN (I.E., D11 OR D12 WAS CHOSEN),
CHOOSE ONE OF D13, D14, OR D15

D13. A JOB WHICH WILL NOT BE COMPLETED AT THE CURRENT OPERATION
    ALWAYS HAS PREFERENCE OVER A JOB WHICH IS TO BE COMPLETED
    AT THIS CURRENT OPERATION

D14. A JOB WHICH WILL BE COMPLETED AT THE CURRENT OPERATION
    ALWAYS HAS PREFERENCE OVER A JOB WHICH IS NOT TO BE
    COMPLETED AT THIS CURRENT OPERATION

D15. A JOB WHICH WILL NOT BE COMPLETED AT THE CURRENT OPERATION
    WILL HAVE PREFERENCE OVER A JOB WHICH WILL BE COMPLETED AT
    THIS CURRENT OPERATION IF AND ONLY IF THE NUMBER IN THE QUEUE
    (OR BACKLOG, AS APPLICABLE) AT THE SUBSEQUENT OPERATION OF
    THE JOB NOT TO BE COMPLETED IS LESS THAN A SPECIFIED PARA-
    METER WHICH DEPENDS ON THE RESOURCE OF THE NEXT OPERATION
IF A LOOK-AHEAD RULE WAS CHOSEN (I.E., D11 OR D12 WAS CHOSEN),
CHOOSE ONE OF D1-D7 AS THE RULE OF CHOICE FOR JOBS TO BE
COMPLETED AT THE CURRENT OPERATION

SELECTION OF SECONDARY RESOURCE TO USE WITH PRIMARY RESOURCE

IF A JOB IS ALWAYS PROCESSED BY ONE UNIT OF ONE RESOURCE AT EACH OF
ITS STAGES (I.E., A5 WAS CHOSEN), GO TO SECTION E. OTHERWISE,
CHOOSE ONE OF D16-D18.

AMONG THE SECONDARY RESOURCES THAT CAN SERVE THE PARTICULAR
PRIMARY RESOURCE IN QUESTION... :

D16. CHOOSE THE FIRST WHICH HAS AN AVAILABLE UNIT. (THE ORDER
IN WHICH SECONDARIES ARE EXAMINED FOR A GIVEN PRIMARY
RESOURCE IS DETERMINED BY THE USER’S INPUT LIST)

D17. CHOOSE THE AVAILABLE SECONDARY RESOURCE WITH THE LARGEST
NUMBER OF AVAILABLE UNITS

D18. CHOOSE THE AVAILABLE SECONDARY RESOURCE WITH THE LARGEST
VALUE OF A \times (B) (NO. OF AVAILABLE UNITS), WHERE A AND B
DEPEND ON THE PARTICULAR PRIMARY/SECONDARY RESOURCE
COMBINATION

SELECTION OF PRIMARY RESOURCE TO WHICH A NEWLY AVAILABLE SECONDARY
RESOURCE IS TO BE ASSIGNED

CHOOSE ONE OF D19-D23

AMONG THE PRIMARY RESOURCES THAT CAN BE SERVED BY THE PARTICULAR
SECONDARY RESOURCE IN QUESTION... :

D19. CHOOSE THE FIRST WHICH HAS AN AVAILABLE UNIT. (THE ORDER
IN WHICH PRIMARIES ARE EXAMINED FOR A GIVEN SECONDARY
RESOURCE IS DETERMINED BY THE USER’S INPUT LIST)

D20. CHOOSE THE AVAILABLE PRIMARY RESOURCE WITH THE LARGEST
NUMBER OF JOBS IN QUEUE

D21. CHOOSE THE AVAILABLE PRIMARY RESOURCE WITH THE LARGEST
BACKLOG HOURS IN QUEUE*

D22. CHOOSE THE AVAILABLE PRIMARY RESOURCE WHOSE FIRST JOB IN
QUEUE HAS THE GREATEST PRIORITY

D23. CHOOSE THE AVAILABLE PRIMARY RESOURCE WITH THE LARGEST VALUE
OF A \times (B) (NO. OF JOBS IN QUEUE) \times (C) (BACKLOG) \times (D) (PRIORITY OF THE FIRST JOB IN QUEUE), WHERE A, B, C, AND D
DEPEND ON THE PARTICULAR PRIMARY/SECONDARY RESOURCE
COMBINATION. THE + SIGN IS USED IF THE GREATEST PRIORITY
IS ASSOCIATED WITH THE HIGHEST PRIORITY NUMBER AND THE
- SIGN IF THE GREATEST PRIORITY IS ASSOCIATED WITH THE
LOWEST PRIORITY NUMBER.*

*NOT PERMITTED IF RANDOM OR LOOK-AHEAD JOB SELECTION RULE
WAS CHOSEN (I.E., D7, D11, OR D12 WAS CHOSEN)
ORDER OF DISPOSING OF PRIMARY AND SECONDARY RESOURCES

CHOOSE ONE OF D24 OR D25

WHEN A PRIMARY AND A SECONDARY RESOURCE BECOME AVAILABLE SIMULTANEOUSLY AT THE COMPLETION OF A JOB...
D24. THE PRIMARY RESOURCE IS DISPOSED OF FIRST (AS PER D16-D18 ABOVE) AND THEN THE SECONDARY IS DISPOSED OF IF STILL AVAILABLE (AS PER D19-D23 ABOVE)
D25. SAME AS IN D24, BUT SECONDARY DISPOSED OF FIRST

CHOOSE ONE OF D26 OR D27

RESOURCES ARE ASSIGNED AT THE START OF A SHIFT BY...
D26. DISPOSING OF THE PRIMARY RESOURCES (PER D16-D18 ABOVE)
D27. DISPOSING OF THE SECONDARY RESOURCES (PER D19-D23 ABOVE)
E. ANALYSIS OF RESULTS

FORM OF OUTPUT

CHOOSE ONE OF E1 OR E2

E1. AN ANALYSIS TAPE WITH EXPLANATIONS OF OUTPUT MESSAGES IS DESIRED. THE ANALYSIS PROGRAM WILL BE WRITTEN BY THE USER. (IF E1 IS CHOSEN, SKIP E3-E10)

E2. THE STANDARD ANALYSIS PROGRAM PRODUCED BY THE GENERATOR IS DESIRED

INTERNALLY GENERATED REPORTS

CHOOSE ONE OF E3 OR E4

INTERIM REPORTS...

E3. CAN BE CALLED FOR EXOGENOUSLY (I.E., EACH REQUEST FOR INTERIM REPORTS IS SPECIFIED INDIVIDUALLY AS INPUT DATA)

E4. OCCUR PERIODICALLY, EACH TYPE OF REPORT HAVING ITS OWN PERIOD

CHOOSE ANY NUMBER OF ALTERNATIVES FROM E5-E10 BELOW TO INDICATE WHICH TYPES OF REPORTS ARE DESIRED ON AN INTERIM AND/OR SUMMARY BASIS.

<table>
<thead>
<tr>
<th>TYPE OF REPORT</th>
<th>INTERIM</th>
<th>SUMMARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESOURCE UTILIZATION</td>
<td>E5</td>
<td>E8</td>
</tr>
<tr>
<td>QUEUE STATISTICS</td>
<td>E6</td>
<td>E9</td>
</tr>
<tr>
<td>JOB STATISTICS</td>
<td>E7</td>
<td>E10</td>
</tr>
</tbody>
</table>

THE FOLLOWING TWO SECTIONS, F AND G, ARE INCLUDED FOR CLARIFICATION PURPOSES ONLY. THESE SECTIONS SHOULD HAVE BEEN COMPLETED ON THE ANSWER SHEET BY THIS POINT. HOWEVER, READING THIS PORTION OF THE QUESTIONNAIRE WILL SERVE AS A CHECK ON THE ANSWERS PREVIOUSLY GIVEN TO THESE SECTIONS.
F. PROBABILITY DISTRIBUTIONS

JOB ARRIVALS


THE FORM OF THE PROBABILITY DISTRIBUTION OF INTER-ARRIVAL TIMES IS:

- F1. UNIFORM
- F2. CONSTANT (NON-RANDOM)
- F3. RANDOM TABLE LOOK-UP
- F4. EXPONENTIAL
- F5. LOG NORMAL

PROCESS TIMES PER LOT

IF EACH NEW ORDER IS ASSIGNED A QUANTITY TO BE USED IN PROCESS TIME CALCULATIONS, OR IF THE DISTRIBUTIONS OF PROCESS TIME PER LOT MAY HAVE DIFFERENT FORMS (I.E., B17 WAS CHOSEN), GO TO F15. OTHERWISE, CHOOSE ONE OF F7-F14.

THE FORM OF THE PROBABILITY DISTRIBUTION OF PROCESS TIMES PER LOT IS:

- F7. UNIFORM
- F8. CONSTANT (NON-RANDOM)
- F9. RANDOM TABLE LOOK-UP
- F10. EXPONENTIAL
- F11. LOG NORMAL
- F12. WEIBULL
- F13. ERLANG

IF ONE OF F7-F14 WAS CHOSEN, GO TO F36.
QUANTITIES

IF EACH NEW ORDER IS NOT ASSIGNED A QUANTITY, GO TO F36. IF THE
DISTRIBUTIONS OF QUANTITIES MAY HAVE DIFFERENT FORMS (I.E., B19
WAS CHOSEN), GO TO F20. OTHERWISE, CHOOSE ONE OF F15-F19.

THE FORM OF THE PROBABILITY DISTRIBUTION OF QUANTITIES IS...
F15. UNIFORM
F16. CONSTANT
F17. RANDOM TABLE LOOK-UP
F18. POISSON
F19. A FUNCTION SUBPROGRAM, CALLED 'PQTY', WILL BE SUPPLIED
BY THE USER. IF THERE ARE NO JOB TYPES (I.E., B1 WAS
NOT CHOSEN), 'PQTY' HAS NO ARGUMENTS. OTHERWISE,

SET-UP TIMES PER LOT

IF THE DISTRIBUTIONS OF SET-UP TIME PER LOT MAY HAVE DIFFERENT FORMS
(I.E., B21 WAS CHOSEN), GO TO F28. OTHERWISE, CHOOSE ONE OF F20-F27.

THE FORM OF THE PROBABILITY DISTRIBUTION OF SET-UP TIMES PER LOT
IS...
F20. UNIFORM
F21. CONSTANT (NON-RANDOM)
F22. RANDOM TABLE LOOK-UP
F23. EXPONENTIAL
F24. LOG NORMAL
F25. WEIBULL
F26. ERLANG
F27. A FUNCTION SUBPROGRAM, CALLED 'STIME', WILL BE SUPPLIED

PROCESS TIMES PER UNIT

IF THE DISTRIBUTIONS OF PROCESS TIME PER UNIT MAY HAVE DIFFERENT
FORMS (I.E., B23 WAS CHOSEN), GO TO F36. OTHERWISE, CHOOSE
ONE OF F28-F35.

THE FORM OF THE PROBABILITY DISTRIBUTION OF PROCESS TIMES PER
UNIT IS...
F28. UNIFORM
F29. CONSTANT (NON-RANDOM)
F30. RANDOM TABLE LOOK-UP
F31. EXPONENTIAL
F32. LOG NORMAL
F33. WEIBULL
F34. ERLANG
F35. A FUNCTION SUBPROGRAM, CALLED 'UTIME', WILL BE SUPPLIED
FACTOR

IF B24 WAS CHOSEN, CHOOSE ONE OF F36-F39. OTHERWISE, GO TO F40.

THE FORM OF THE PROBABILITY DISTRIBUTION OF THE FACTOR IS...
F36. UNIFORM
F37. EXPONENTIAL
F38. NORMAL
F39. A FUNCTION SUBPROGRAM, CALLED 'FACTR', WILL BE SUPPLIED
      BY THE USER. 'FACTR' HAS ONE ARGUMENT, THE PRIMARY
      RESOURCE.

ESTIMATED TO ACTUAL PROCESSING TIMES

IF EITHER C1, C2, OR C4 WAS CHOSEN, CHOOSE ONE OF F40-F44.
OTHERWISE, GO TO F45.

THE FORM OF THE PROBABILITY DISTRIBUTION OF THE RATIO OF ESTIMATED
TO ACTUAL PROCESSING TIMES IS...
F40. UNIFORM
F41. CONSTANT
F42. RANDOM TABLE LOOK-UP
F43. LOG NORMAL
F44. A FUNCTION SUBPROGRAM, CALLED 'EVSA', WILL BE SUPPLIED BY
      THE USER. IF C1 WAS CHOSEN, 'EVSA' HAS NO ARGUMENTS.
      IF C2 WAS CHOSEN, IT HAS ONE ARGUMENT (I.E., THE TYPE OF THE
      JOB). IF C4 WAS CHOSEN, IT HAS ONE ARGUMENT (I.E., THE
      RESOURCE AT THE CURRENT OPERATION).

DUE-DATE INCREMENT

IF THE 'DUE-DATE' IS NOT REQUIRED, GO TO F52. IF THE DUE-DATE
IS EXOGENOUS (I.E., C6 WAS CHOSEN), OR IF THE DISTRIBUTIONS OF
THE INCREMENT MAY HAVE DIFFERENT FORMS (I.E., C8 WAS CHOSEN),
GO TO F52. OTHERWISE, CHOOSE ONE OF F45-F51.

THE FORM OF THE PROBABILITY DISTRIBUTION OF THE DUE-DATE
INCREMENT IS...
F45. UNIFORM
F46. CONSTANT (NON-RANDOM)
F47. RANDOM TABLE LOOK-UP
F48. EXPONENTIAL
F49. LOG NORMAL
F50. NORMAL
F51. A FUNCTION SUBPROGRAM, CALLED 'DINC', WILL BE SUPPLIED
      BY THE USER. IF THERE ARE NO JOB TYPES (I.E., B1 WAS
      NOT CHOSEN), OR IF THE INCREMENT DOES NOT DEPEND ON JOB
      TYPE, 'DINC' HAS NO ARGUMENTS. OTHERWISE, 'DINC' HAS ONE
DOLLAR VALUE

IF THE DOLLAR VALUE IS NOT REQUIRED, GO TO SECTION G. IF THE 
DOLLAR VALUE IS EXOGENOUS (I.E., C9 WAS CHOSEN), OR IF THE 
DISTRIBUTIONS OF DOLLAR VALUE MAY HAVE DIFFERENT FORMS (I.E., C11 
WAS CHOSEN), GO TO SECTION G. OTHERWISE, CHOOSE ONE OF F52-F57.

THE FORM OF THE PROBABILITY DISTRIBUTION OF DOLLAR VALUES IS...
F52. UNIFORM
F53. CONSTANT (NON-RANDOM)
F54. RANDOM TABLE LOOK-UP
F55. EXPONENTIAL
F56. LOG NORMAL
F57. A FUNCTION SUBPROGRAM, CALLED 'VALUE', WILL BE SUPPLIED 
BY THE USER. IF THERE ARE NO JOB TYPES (I.E., B1 WAS 
NOT CHOSEN), OR IF THE DOLLAR VALUE DOES NOT DEPEND 
ON JOB TYPE, 'VALUE' HAS NO ARGUMENTS. OTHERWISE, 
G. INITIAL INPUT VALUES

GENERAL INFORMATION

G1. CHOOSING THIS OPTION INDICATES THAT ONLY THE INITIAL INPUT VALUES ARE TO BE CHANGED (I.E., ONLY A NEW INITIALIZATION DECK WILL BE PROVIDED, NOT AN ENTIRE NEW PROGRAM). THE USER MUST NOW COMPLETE THE REMAINDER OF SECTION G AND SUBMIT THE NEW 'G' INFORMATION WITH THE PREVIOUS 'A-F' CHOICES.

THE FOLLOWING INFORMATION IS REQUIRED FOR ALL JOB SHOPS...

G2-G5. THE USER'S JOB NUMBER
G6-G10. THE USER'S MAN NUMBER (AN ALPHABETIC CHARACTER FOLLOWED BY A FOUR-DIGIT NUMBER)

SHIFT CHANGE INFORMATION

IF EVERY DAY IS THE SAME WITH ONLY ONE SHIFT PER DAY (I.E., A1 WAS CHOSEN), GO TO G23. OTHERWISE, CONTINUE HERE.

G11-G14. INSERT ON THE QUESTIONNAIRE ANSWER SHEET THE MAXIMUM NUMBER OF SHIFTS IN ANY DAY

IF EVERY DAY IS THE SAME (I.E., A2 WAS CHOSEN), GO TO G23. OTHERWISE, CONTINUE HERE.

G15-G18. INSERT ON THE QUESTIONNAIRE ANSWER SHEET THE NUMBER OF DIFFERENT TYPES OF DAYS

IF EVERY WEEK IS THE SAME (I.E., A3 WAS CHOSEN), GO TO G23. OTHERWISE, CONTINUE HERE.

G19-G22. INSERT ON THE QUESTIONNAIRE ANSWER SHEET THE NUMBER OF DIFFERENT TYPES OF WEEKS

RESOURCE INFORMATION

THE FOLLOWING INFORMATION IS REQUIRED FOR ALL JOB SHOPS...

G23-G26. THE MAXIMUM NUMBER OF PRIMARY RESOURCES

IF EACH JOB IS PROCESSED BY ONLY ONE RESOURCE AT EACH STAGE (I.E., A5 WAS CHOSEN), GO TO G9. OTHERWISE, CONTINUE HERE.

G27-G30. INSERT ON THE QUESTIONNAIRE ANSWER SHEET THE MAXIMUM NUMBER OF SECONDARY RESOURCES
JOB TYPES

IF EACH JOB IS ASSIGNED A JOB TYPE (I.E., B1 WAS CHOSEN),
THE FOLLOWING INFORMATION IS REQUIRED...

G31-G34. THE NUMBER OF DIFFERENT TYPES OF JOBS
**A. Resource Description**

| Shift Change Options | | |
|----------------------|----------------|
| No shifts, every day the same | Every shift, every week, the same, general |

**B. Job Characteristics**

| There are job types | | |
|---------------------|----------------|
| Job arrivals are exogenous, | Routing & process time, type & quantity, type, random, not by type, arrival only, no types |

<table>
<thead>
<tr>
<th>Work-rule arrival times</th>
<th>Same form, random, same form, random, no types, same form, random,</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Process time/lot</td>
<td>Same form, different forms, same form, different forms,</td>
<td></td>
</tr>
<tr>
<td>Quantities</td>
<td>Same form, different forms, same form, different forms,</td>
<td></td>
</tr>
</tbody>
</table>

**C. Characteristics for Decision Rules**

<table>
<thead>
<tr>
<th>Estimated to actual process time</th>
<th>Different forms, types,</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Due-date increment</td>
<td>Exogenous, same form, different forms, different forms,</td>
<td></td>
</tr>
<tr>
<td>Dollar value</td>
<td>Exogenous, same form, different forms, different forms,</td>
<td></td>
</tr>
</tbody>
</table>

**D. Decision Rules**

<table>
<thead>
<tr>
<th>General priority rules</th>
<th>First come, first served, shortest process time, longest process time, earliest arrival, earliest due-date, largest value, random, user's function,</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Slack rules</td>
<td>Equal weights, unequal weights,</td>
<td></td>
</tr>
</tbody>
</table>

**E. Analysis**

| Output | Input | Summary | | |
|--------|-------|---------|----------------|
| Output | Target, periodic reports, | |
| Input | Exogenous, resource utilization, | |
| Summary | Resource utilization, queue statistics, job statistics, | |

**F. Probability Distributions**

<table>
<thead>
<tr>
<th>Distribution type</th>
<th>Type of system, exponential, normal, lognormal,</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Process times/lot</td>
<td>Product of times/unit, quantities, setup times/unit,</td>
<td></td>
</tr>
</tbody>
</table>

**C. Initial Input Values**

<table>
<thead>
<tr>
<th>Job No.</th>
<th>Man No.</th>
<th>Ships/day</th>
<th>Days</th>
<th>Week Type</th>
<th>Primary Resource</th>
<th>Secondary Resource</th>
<th>Job Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
</tr>
</tbody>
</table>

**Fig. 2 -- Answer Sheet**
Appendix B

SAMPLE STATEMENT LIST AND DECISION TABLES
*IBFTC SALE
ENDOGENOUS EVENT SALE
EXOGENOUS EVENT SALE
SAVE
CREATE JOB
LET TYPEJ(JOB) = STYPE(SALE)
LET TYPEJ(JOB) = IDIST(PTYPE)
CAUSE SALE AT TIME + TSAVE(TYPEJ(JOB))
CAUSE SALE AT TIME + TSAVE
READ TYPEJ(JOB)
READ TYPEJ(JOB), QTY(JOB)
READ QTY(JOB)
READ DDATE(JOB)
READ COST(JOB)
X, DDATE(JOB)
X, COST(JOB)

FORMAT (I4)
FORMAT (D3.3)
FORMAT (D6.2)
X, D3.3
X, I4
X, D6.2
X)
DIMENSION RS(6), PR(6)
10 READ RS(I), PR(I), FOR I = (1)(6)

FORMAT 6(I4,D4,3)
DO TO 20, FOR I=(1)(6)
IF (RS(I) GE (999), GO TO 30
CREATE STAGE CALLED S
LET RES(S) = RS(I)
LET TIMEA(S) = PR(I)
FILE S IN RTG(JOB)
20 LOOP
GO TO 10
X
30 CONTINUE
LET QTY(JOB) = PQTY
LET QTY(JOB) = PQTY(TYPEJ(JOB))
LET DDATE(JOB) = TIME + DINC
LET DDATE(JOB) = TIME + DINC(TYPEJ(JOB))
LET COST(JOB) = VALUE
LET COST(JOB) = VALUE(TYPEJ(JOB))
LET ADATE(JOB) = TIME
CALL DISP(JOB)
RETURN
END
T    SALE
B 2 N
B 4 N
B 5 N
B 7 N
C 5 N
C 9 N

U
202 X X X
1 X
2 X X
3 X
4 X X X

E

T    SALE
B 1 N Y
B 2 N
B 4 N
B 5 N
B 9 N

U
5 X
6 X

E

T    SALE
B 2 Y
B 10 N
B 11 N

U
7 X
8 X

E
## SALE

<table>
<thead>
<tr>
<th>T</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>N</td>
</tr>
</tbody>
</table>

## U

<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>18</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>19</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>101</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>102</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>484</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>485</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>487</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>488</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

## E

<table>
<thead>
<tr>
<th></th>
<th>12</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>18</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>N</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>N</td>
</tr>
</tbody>
</table>

## U

<table>
<thead>
<tr>
<th></th>
<th>12</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

## E
T SALE

C 7 N N
C 8 N N
F 45 N
F 46 N
F 47 N
F 48 N
F 49 N
F 50 N
F 51 N

U

103 X
104 X

E

T SALE

C 10 N N
C 11 N N
F 52 N
F 53 N
F 54 N
F 55 N
F 56 N
F 57 N

U

105 X
106 X

E

T SALE

D 4 N
E 5 N
E 8 N

U

107 X
15 X X
27 X X
16 X X

E