Regulations for the Use of Theses

Unpublished theses submitted for the Master's and Doctor's degrees and deposited in the Georgia Institute of Technology Library are open for inspection and consultation, but must be used with due regard for the rights of the authors. Passages may be copied only with permission of the authors, and proper credit must be given in subsequent written or published work. Extensive copying or publication of the thesis in whole or in part requires the consent of the Dean of the Graduate Division of the Georgia Institute of Technology.

This thesis by ROBERT LOUIS EIGEL has been used by the following persons, whose signatures attest their acceptance of the above restrictions.

A library which borrows this thesis for use by its patrons is expected to secure the signature of each user.

NAME AND ADDRESS OF USER BORROWING LIBRARY DATE

CGSC - DC
FREDERICK, MD

20 Jun
In presenting the dissertation as a partial fulfillment of the requirements for an advanced degree from the Georgia Institute of Technology, I agree that the Library of the Institute shall make it available for inspection and circulation in accordance with its regulations governing materials of this type. I agree that permission to copy from, or to publish from, this dissertation may be granted by the professor under whose direction it was written, or, in his absence, by the Dean of the Graduate Division when such copying or publication is solely for scholarly purposes and does not involve potential financial gain. It is understood that any copying from, or publication of, this dissertation which involves potential financial gain will not be allowed without written permission.

3/17/65

[Signature]
A PERT SIMULATION GAME

A THESIS
Presented to
The Faculty of the Graduate Division
by
Robert Louis Eigel

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Industrial Engineering

Georgia Institute of Technology
March, 1968
A PERT SIMULATION GAME

Approved:

Chair:

Date approved by Chairman: 4-15-68
ACKNOWLEDGMENTS

The author is indebted to many persons whose help and encouragement made this work possible. First, he would like to express his deepest appreciation to Dr. J. Gordon Davis. Dr. Davis' generous assistance and patient encouragement have meant very much to the author. He would also like to thank Dr. Kong Chu and Dr. W. W. Hines for their advice and service on the reading committee.

Finally, the author would like to express his appreciation to Miss Charlotte Ross whose generous support has been of invaluable assistance during the preparation of this thesis.
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>ii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>iv</td>
</tr>
<tr>
<td>List of Illustrations</td>
<td>v</td>
</tr>
<tr>
<td>Summary</td>
<td>vi</td>
</tr>
<tr>
<td><strong>Chapter</strong></td>
<td></td>
</tr>
<tr>
<td><strong>I. INTRODUCTION.</strong></td>
<td>1</td>
</tr>
<tr>
<td>Objective</td>
<td></td>
</tr>
<tr>
<td>General</td>
<td></td>
</tr>
<tr>
<td>Literature Survey</td>
<td></td>
</tr>
<tr>
<td><strong>II. METHOD OF PROCEDURE</strong></td>
<td>8</td>
</tr>
<tr>
<td>General</td>
<td></td>
</tr>
<tr>
<td>A Formulation of the Problem</td>
<td></td>
</tr>
<tr>
<td>A Determination of the Probability Distributions</td>
<td></td>
</tr>
<tr>
<td>of the Status Variables to be Included</td>
<td></td>
</tr>
<tr>
<td>in the Environment Program</td>
<td></td>
</tr>
<tr>
<td>A Formulation of the Computer Program</td>
<td></td>
</tr>
<tr>
<td>A Test of the Program's Validity</td>
<td></td>
</tr>
<tr>
<td>The Development of a Supplemental Training Form</td>
<td></td>
</tr>
<tr>
<td><strong>III. RESULTS</strong></td>
<td>49</td>
</tr>
<tr>
<td>The PERT Project</td>
<td></td>
</tr>
<tr>
<td>The Computer Program</td>
<td></td>
</tr>
<tr>
<td>A PERT Simulation Game</td>
<td></td>
</tr>
<tr>
<td>The Play of the Game</td>
<td></td>
</tr>
<tr>
<td>Explanation of Tables</td>
<td></td>
</tr>
<tr>
<td>Procedure</td>
<td></td>
</tr>
<tr>
<td><strong>IV. CONCLUSIONS AND RECOMMENDATIONS</strong></td>
<td>82</td>
</tr>
<tr>
<td><strong>APPENDIX A</strong></td>
<td>90</td>
</tr>
<tr>
<td><strong>APPENDIX B</strong></td>
<td>93</td>
</tr>
<tr>
<td><strong>LITERATURE CITED</strong></td>
<td>115</td>
</tr>
<tr>
<td><strong>OTHER REFERENCES</strong></td>
<td>116</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th></th>
<th>Table Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Activity Time Estimates with Associated Parameters and Probabilities</td>
<td>20</td>
</tr>
<tr>
<td>2</td>
<td>Resource Availability Probabilities</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>Possible States of a Process</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>The Sequencing Set</td>
<td>41</td>
</tr>
<tr>
<td>5</td>
<td>Game Activities</td>
<td>52</td>
</tr>
<tr>
<td>6</td>
<td>Activities Which Can Be Accelerated</td>
<td>54</td>
</tr>
<tr>
<td>7</td>
<td>Input Variables for Each Activity</td>
<td>57</td>
</tr>
<tr>
<td>8</td>
<td>Resource Costs</td>
<td>66</td>
</tr>
<tr>
<td>9</td>
<td>Activity Time Estimates</td>
<td>72</td>
</tr>
<tr>
<td>10</td>
<td>Resource Availability</td>
<td>75</td>
</tr>
<tr>
<td>Illustration Description</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>1. Components of the Computer Program</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2. BETA Distribution</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>3. An ALGOL Block</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>4. An ALGOL Program with a Nested Block</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>5. A Three-Level ALGOL Program</td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>6. Flow Diagram for the Main Program</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>7. Flow Diagram for SIMULA Activities</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>8. A Simple PERT Network with a Dummy Activity</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>10. Network for the Game Project</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>11. PERT Activity Listing Subroutine</td>
<td>62</td>
<td></td>
</tr>
<tr>
<td>12. Typical Program Output for a Six Day Workweek</td>
<td>63</td>
<td></td>
</tr>
<tr>
<td>13. Game Data Chart</td>
<td>80</td>
<td></td>
</tr>
</tbody>
</table>
It has been ten years since critical path methods were introduced as an aid to Project Management. In that time, PERT/CPM management techniques have gained wide popularity among military and industrial managers.

The extensive utilization of modern PERT/CPM methods creates a need for vigorous training in this art for those potential managers who are most likely to be concerned with either military or industrial projects. As an aid to future educational programs dedicated to this end, it was the purpose of this study to develop a training device for PERT/CPM students which would relate classroom lessons to the uncertainties of a realistic project in the form of a game.

This study describes a computer simulation game based on a typical construction project which will present the student with a challenging test of his ability to implement PERT/CPM techniques. The game requires the student to "manage" the hypothetical project with standard PERT/CPM procedures as variable project data is supplied to him by the computer.

It is anticipated that simulation games of the sort described herein, but refined and improved beyond the state of this rudimentary game, will considerably enhance the state of the educational art with respect to PERT and CPM.
CHAPTER I

INTRODUCTION

Objective

The objective of this thesis is to design and construct a simulation game which will assist students of the management technique known as the Program and Evaluation and Review Technique (PERT). The game is based on a hypothetical construction project which is managed by PERT methods. The use of this game as a part of a course of instruction on PERT procedures will improve the student's understanding of the real world complexities normally associated with PERT analyses.

General

The introduction in 1958 of the project management techniques known as the Program and Evaluation Review Technique (PERT) and the Critical Path Method (CPM) marked the first significant innovation in the field of project management since Henry Gantt introduced the bar chart in 1900. The influence of these two techniques in both military and industrial situations can hardly be overstated.

PERT and CPM are network planning devices which differ principally from previous management systems in that they "can handle uncertainty and (have) capacities for prediction and simulation."* Both

techniques incorporate what is called the "critical path method" to plan, schedule, and control any given project. The network for each consists of a number of activities which are completed in the order as drawn on the network. PERT includes a variable activity performance time whereas CPM is based on a completely deterministic activity performance time. Other than this difference, the two systems are very similar and are subject to the same rules of procedure.

PERT/CPM methodology is fairly straightforward. The obvious logic of network planning is the major advantage of these two management techniques and because of this many managers have benefited from the application of critical path methods to tightly scheduled projects with limited resource availability.

Although the principles involved are fundamental, the actual implementation of PERT/CPM methodology to real projects can be deceptively easy. Inexperienced users commonly underestimate the exhausting logic that is required to construct a workable network. This difficulty is made worse as the project to be networked becomes more complex. This phenomenon is similar to that experienced by an apprentice computer programmer who is not familiar with the precise logic of the computer. Although an attempt is made to acquaint the beginner with this problem in the classroom, only experience born of repeated trials will ultimately overcome it.

A second difficulty with networks, and PERT networks in particular, is the degree, if any, to which the network will include the element of uncertainty normally associated with the real world. It is not always sufficiently impressed upon a student that real world
situations do not generally conform to the precision of classroom models. Here again, experience will serve to alleviate any mistaken impressions a learner might develop in an academic environment. However, it is possible to artificially introduce a modicum of uncertainty in academic problems through the use of computer simulation.

Generally, simulation is defined to be an operable representation of some real system. But Shubick offers a more precise definition.

A simulation of a system or an organism is the operation of a model or simulator which is a representation of the system or organism. The model is amenable to manipulation which would be impossible, too expensive or impractical to perform on the entity it portrays. The operation of the model can be studied and, from it, properties concerning the behaviour of the actual system or its sub-systems can be inferred. *

The model or simulator in Shubick's definition can be identified as one of three types of simulation models that have been defined by Buffa. ** An Iconic model is one which actually looks like the thing it represents but which is usually scaled up or down from the original. Model airplanes and planetariums are examples of Iconic models. Analogue models are models which represent something by analogy and which establish a relationship between a variable in the system and an analogous variable in the model. A graph of a firm's sales by months uses the lengths of the lines as analogous to the sales and time. The last type of model is the Symbolic model which substitutes symbols for

---


the components of a real world system. These symbols are usually related mathematically.

Symbolic simulation of realistic situations was, until recently, technologically difficult. During and after World War II, however, certain mathematical techniques were introduced that resolved technical problems which had been either too expensive to solve experimentally or too complex for analytical treatment. And, in the past 20 years, the art of simulation has expanded to include experimentation on the digital computer in the fields of business management and economics.

The application of symbolic simulation principles to a hypothetical PERT project would have many useful results. Simulation of this sort would afford a project manager the opportunity to test changes in scheduling and/or resource allocation before actually incorporating these changes in a real project. The advantages of such simulation include the avoidance of certain operational mistakes (which could be very expensive) and an increased awareness of the consequences of the proposed changes. Additionally, the time required to adequately evaluate the effects of changes is not a factor in simulation whereas it is of major importance in genuine operations.

Another use of a simulated PERT project would be as an educational device. The simulation could be designed to include human interactions with the model in the form of a game. Games are exercises during the course of which the model is manipulated by one or more human "players" as well as other factors. Games are commonly held to
be valuable training devices. *

Games have been most often applied to military and business operations. In both cases games have been used as training aids to assist those who study the effects of different strategies under simulated conditions of war or business.

The definition of a business game introduced in 1961 is as follows:

A business game is a contrived situation which imbeds players in a simulated business environment, where they must make management-type decisions from time to time, and their choices generally affect the environmental conditions under which subsequent decisions must be made. Further, the interaction between decisions and environment is determined by a refereeing process which is not open to argument from the players. **

If a computer is employed as the "refereeing process" mentioned above, it is possible to introduce game variables which would otherwise be too tedious to evaluate by hand calculations.

The purpose of this thesis, then, is to design and construct a computer program for a simulated PERT project in the form of a game. The game would serve as an introduction to the multiple uncertainties which normally confront a modern project manager. As the "player" in a simulated PERT game, the student is cast as the decision-maker for the hypothetical project and he is thereby given the opportunity to implement the principles of PERT/CPM that he has learned in the

---

*Proceedings of the Conference on Business Games, Sponsored by the Ford Foundation and the School of Business Administration, Tulane University, April 26-28, 1961.

classroom. The value of such a game led Tiffin to observe that "to
the extent that the game (does) parallel the problems and situations
of the real world, it would be expected that some of the 'learning'
would rub off on the participants, and would, therefore, have a
carry-over effect to their jobs."* The active participation of the
student in this game should, therefore, enhance his awareness of the
uncertainties of real world projects and consequently improve his
academic preparation toward that end.

Literature Survey
The literature on the applications of computer simulation to
various disciplines such as medicine, physics and engineering is exten-
sive. General methodologies dealing with simulation techniques are
available. The relevance of established simulation theory to the prob-
lem of a PERT/CPM simulation game has been established by Archibald and
Villoria.**

Unfortunately, very little has been done in this particular area
of simulation. A few PERT/CPM simulation games have been introduced
which require the use of tedious hand-processed algorithms. A computer-
assisted simulation of a PERT project, called CAPERTISM, *** has been
developed by the U. S. Army. This program does not generate a dynamic

p. 313.


*** CAPERTISM Student Manual, U. S. Army Logistics Management
Center, Fort Lee, Virginia, 1961.
environment, but merely processes input data in a manner similar to static PERT computer programs. Although one purpose of CAPERTISM is to train military and civilian personnel in the use of PERT, it was designed to acquaint students with the use of the computer as an aid to management. (This is not the purpose of the game described herein since it is assumed that the students who are to "play" the game are at least familiar with computer operations.) The dual purpose of CAPERTISM necessarily restricts the flexibility of the program.

The reasons PERT/CPM simulation games are currently uncommon are unclear since the complexity of the appropriate simulation model is within the capabilities of modern simulation techniques. The situation is succinctly described by Archibald and Villoria when they observe that although students can benefit from such a model,

The type of model described . . . is not being used because none is presently programmed. However, this type of model is technologically feasible as is demonstrated by the existing models that contain the same type of features as this one.*

The state of the simulation art, highlighted by the improved special-purpose computer languages such as SIMULA and SIMSCRIPT is such that a translation of the PERT/CPM simulation model into a workable management game is not only desirable but attainable as well. Consequently, a simple PERT simulation game was designed and built and is described in the following chapters.

*Archibald and Villoria, op. cit., p. 407.
CHAPTER II

METHOD OF PROCEDURE

General

Since there have been no previous attempts to simulate a PERT project on a computer in which variable input data dynamically interacts with information generated by the computer program, it was necessary for the author of this paper to develop such a program by following classical simulation procedures and incorporating along the way those variables and program details as were necessary for the construction of a useful training device. Many questions confronted the author concerning not only the program's content but also the nature of the program itself. Before proceeding to a detailed discussion of these questions, it would be best to give an overview of the program with some explanatory terms.

In simulation games, part of the computer program is that portion which is sometimes called the Environment Program. Certain variables evaluated by the program for the purposes of the game are contained within the Environment Program. The numerical source of these variables is their associated probability distributions which are either assumed or derived from historical observation. An illustration would be the probability distribution assigned to the weather variable in a construction project game. It is assumed that the variables in the Environment Program take on values which follow the probability
distributions experienced in the real world. In those computer simulations where predictive accuracy must be high, great care must be exercised when assigning probability distributions to these variables. In this study, however, the simulation game is not intended to be a predictive vehicle but rather a training device used to familiarize students with the PERT/CPM techniques. Therefore, the probability distributions assigned to the program's internal variables are arbitrary and not rigidly derived from real data.

The computer program includes three general categories of variables: exogenous, status, and endogenous variables. Exogenous variables are the independent or input variables as supplied by the decision-maker or player. Status variables describe the state of the system before, during, or after some specified time period. These variables interact with the exogenous variables according to a functional relationship and are evaluated by the Environment program as previously discussed. Endogenous variables are the dependent or output variables of the program.

The computer program itself consists of three major components. First, the decision-maker or player provides variable input data. Second, the Environment Program computes the value of the status variables. The result of the interaction between the exogenous and status variables are given to the decision-maker for further analysis (Fig. 1).

(The Environment Program is carefully delineated here in order that an optional Network Planning Program, not included as part of this study, can be related to it in a later section of this thesis.)
With this general concept of the program in mind, the author followed a procedure consisting of five steps in the construction of the final result. The five steps are as follows:

1. A Formulation of the Problem
2. A Determination of the Probability Distributions of the Status Variables to be Included in the Environment Program
3. A Formulation of the Computer Program
4. A Test of the Program's Validity
5. The Development of a Supplemental Training Form

A Formulation of the Problem

In order to formulate the problem for the simulation program, three major determinations must be accomplished: A) the management information system to be used; B) the size and detail of the dummy project; and C) the variables to be included among the status variables.
As to the consideration of the management information system to be used, a number of possibilities exist. The program could include the standard outputs of a PERT/CPM program in which the decision-maker is required to manage the project based on information concerning the three parameters of time, cost, and manpower. On the other hand, the experimental simulation program could be designed to exclude any one or two of these variables. However, since this simulated game is to be used as an instructional aid, the program should include all three of the informational outputs of a standard PERT/CPM program.

The second necessary determination involved answers to two questions. First, should the program model be based on a singular dummy project or would it be better, if feasible, to construct a program which could accept a number of different projects subject to certain general constraints? While it could be argued that a generalized, more flexible program has certain advantages not found in a program with a singular dummy project, there are several aspects inherent in the former which inveigh against such a program here.

It should be remembered that the program developed in this study is to be used by relatively inexperienced students. A complicated and advanced multi-purpose program could be advantageous to users already familiar with PERT/CPM techniques and who seek a wider experience base through the simulation medium. But the flexibility of such a program could not be used constructively by student beginners. Moreover, a program of this power and scope would require the time and efforts of more than one man for a period longer than was available to the author.
Therefore, the program model is designed to accept a singular dummy project.

Having resolved the first question, it was then necessary to determine the size and detail of the dummy project. PERT networks vary greatly in size according to the desired level of detail. The use to which the dummy project was to be directed made the consideration of detail less important than it is in the real world. It was necessary to have a resultant network which would be representative of some real project in sufficient detail to make the game credible but which would not be so large as to interfere with the expeditious conduct of the game.

A hypothetical project which meets the above criteria was designed by Dr. J. Gordon Davis and is used by him in his classes on PERT at the Georgia Institute of Technology. The project concerns the construction of a house and in its original form consisted of 32 activities. It has been modified slightly for the purposes of the simulation, and in this thesis it is composed of 35 activities, including four dummy activities.

The third and last consideration in the formulation of the problem concerned which variables were to be included among the status variables. The status variables in the Environment Program are the essence of the gaming situation. There are many problems which normally confront a project manager. Some of these problems are labor strikes, weather phenomena, funding delays, material shortages, employee absenteeism and changes in the scheduled activity completion dates. It would be ideal if all sources of project variation could be incorporated
in the simulation model. Such a model is impractical, however, and a balance, between the complexities of the real world and a model sufficiently uncomplicated so as to permit easy understanding, was the objective of this study. To this end, then, four types of status variables were included: weather, funding, resource, and activity duration variables. All of these variables, with the possible exception of the weather variable to which construction projects are particularly sensitive, are sources of variability common to most projects which are subjected to PERT methodology. They give the game a marked degree of realism but are not so many as to preclude easy user comprehension.

With the above information the problem can be briefly summarized. The student-player's objective will be to use PERT/CPM techniques to manipulate given and program data to construct a hypothetical house. The project consists of 35 activities for which the player will be required to allocate resources and manage costs within scheduling constraints. Resource requirements, cost data, expected resource availability, maximum project duration without penalty, and PERT time estimates for each activity are given data. Actual resource availability, actual activity duration times, actual available funds, and work stoppages due to weather are program-generated data.

A Determination of the Probability Distributions of the Status Variables to be Included in the Environment Program

As mentioned above, there are four types of status variables included in the Environment Program. These variables are: A) Activity Duration Variable, B) Resource Availability Variable, C) Funding Variable, and D) Weather Variable.
The special purpose simulation language (SIMULA) used in this thesis has the capability of randomly sampling from several commonly-used functional forms. Among these functional forms that are a permanent part of the language are the normal, negative exponential, and Poisson distributions. To randomly sample from one of the above distributions, one need only specify the parameters of the distribution in question and call the appropriate SIMULA subroutine by name. Thus, if the mean and standard deviation of a normally distributed variable are defined, the computer will generate a pseudo-random stream of numbers from this distribution whenever the procedure identified by the word NORMAL is called. For other functional forms that are not a permanent part of the language, it is possible to generate a pseudo-random stream of values from the desired distribution if a histogram defining the relative frequencies of the values within the range of the function are given.

The Funding Variable and the Weather Variable have been assumed to vary according to functional forms that are a permanent part of the language. The activity durations and the resource availabilities have been assumed to vary according to functional forms that are not a permanent part of the language. For the latter two types of status variables, then, it was necessary to tabulate the probability of occurrence for each value within the range of each variable.

The functional form which is most often used to approximate the probability density curve of activities with three time estimates
(PERT) is the beta distribution. This distribution is entirely contained within a finite interval and can be symmetrical or skewed depending upon the most likely time estimate. Accordingly, the actual activity durations computed by the Environment Program are obtained by sampling from a beta distribution (Fig. 2).

For PERT calculations, the end points and the mode are given as:

- \( a \) = optimistic time estimate or the shortest possible activity duration
- \( m \) = most likely activity duration
- \( b \) = pessimistic or the longest possible activity duration

The probability density function of the random variable "activity time" is:

\[
f(t) = (K)(t - a)^{\alpha}(b - t)^{\beta} \quad \text{for } a < t < b \tag{1}
\]

---

= 0 elsewhere.

where a and b are the optimistic and pessimistic time estimates, respectively (the range limits) and K, α, and β are functions of a, b, and m (the most likely time estimate or the mode). In order to calculate the probabilities associated with values within the range of an activity's time estimates, it was necessary to determine the values of the parameters α and β in Equation (1) for each set of three time estimates used in the computer program.

A previous study* has derived a set of formulas that permit an easy determination of the values of α and β for the distribution of the variable "activity time" if the most likely, optimistic, and pessimistic times are given. Without going into the detail of the report cited below, the probability density function given in Equation (1) can be reduced to the standard form of the beta distribution by the introduction of a random variable x related to t in the following manner:

\[ x = (t - a)/(b - a) \]  

(2)

It is shown in the report that if m is the mode of t, r, the mode of x, is found to be:

\[ r = (m - a)/(b - a) \]  

(3)

where m, a, and b are the time estimates previously defined. Furthermore, the mode and expected value of x were shown to be:

\[ r = \frac{a}{a + \beta} \]  \hspace{1cm} (4)

and

\[ E(x) = \frac{a + 1}{a + \beta + a}. \]  \hspace{1cm} (5)

The last formula presented in the report that is required to determine \( \alpha \) and \( \beta \) is an approximation of the expected value of x as a function of r. This approximation,

\[ E(x) = \frac{4r + 1}{6}, \]  \hspace{1cm} (6)

has been accepted as sufficiently accurate for the purposes of this thesis.* Knowing the value of r and having an acceptable approximation of the expected value of x, it is a simple procedure to determine \( \alpha \) and \( \beta \) using Equation (4) and Equation (5).

For example, if the time estimates for an activity are \( a = 1 \), \( m = 2 \), and \( b = 4 \), the parameters, \( \alpha \) and \( \beta \), for the curve so defined are evaluated as follows:

\[ r = \frac{2 - 0.5}{4.5 - 0.5} \]

---

*See Archibald and Villoria, op.cit., pp. 448-453.
\[ E(x) = \frac{4(3/8) + 1}{6} \]
\[ = 5/12 \]

Therefore

\[ 3/8 = \frac{\alpha}{\alpha + \beta} \]

and

\[ 5/12 = \frac{\alpha + 1}{\alpha + \beta + \alpha} \]

which result in the values

\[ \alpha = 3/2, \quad \beta = 5/2 \]

These values for \( \alpha \) and \( \beta \) uniquely determine the beta distribution for the activity in question. It is now possible to find the probability of occurrence over any interval within the range of this activity's time estimates through the use of Pearson's Tables of the Incomplete Beta-Function.*

---

The commonly followed procedure when using a continuous function to approximate a discrete function is to assign each discrete value a probability which is the cumulative probability over an interval of the continuous function. Thus, the probability that the duration of the subject activity will be two is the probability over the interval from 1.5 to 2.5 in the \textit{beta} distribution defined by the above values of $\alpha$ and $\beta$. Using Pearson's Tables, the probabilities associated with this activity are as follows:

\begin{align*}
P\{t = 1\} &\equiv P\{t < 1.5\} = 0.194 \\
P\{t = 2\} &\equiv P\{1.5 < t < 2.5\} = 0.503 \\
P\{t = 3\} &\equiv P\{2.5 < t < 3.5\} = 0.279 \\
P\{t = 4\} &\equiv P\{3.5 < t\} = 0.021
\end{align*}

As mentioned previously, ten different \textit{beta} distributions were required for the simulation game. The time estimates, parameters, and probabilities for each of these ten are presented in Table 1.

The second type of status variable to be defined is the resource variable. In this program, five resource variables have been included, which is to say, that on any given day of the construction project there is a variable number of carpenters, masons, electricians, plumbers, and painters available to work on that day.
Table 1. Activity Time Estimates with Associated Parameters and Probabilities

<table>
<thead>
<tr>
<th>Time Estimates</th>
<th>Probability that Activity Duration Will Be X Days</th>
</tr>
</thead>
<tbody>
<tr>
<td>a  m  b  α  β 1  2  3  4  5  6</td>
<td></td>
</tr>
<tr>
<td>1 1 2 1 3</td>
<td>0.875 0.125 0.0 0.0 0.0 0.0</td>
</tr>
<tr>
<td>1 1 3 4/3 20/3</td>
<td>0.169 0.830 0.001 0.0 0.0 0.0</td>
</tr>
<tr>
<td>1 2 3 3 3</td>
<td>0.209 0.582 0.209 0.0 0.0 0.0</td>
</tr>
<tr>
<td>1 2 4 3/2 5/2</td>
<td>0.195 0.503 0.279 0.021 0.0 0.0</td>
</tr>
<tr>
<td>2 3 4 3 3</td>
<td>0.0 0.209 0.582 0.209 0.0 0.0</td>
</tr>
<tr>
<td>2 3 5 3/2 5/2</td>
<td>0.0 0.195 0.503 0.279 0.021 0.0</td>
</tr>
<tr>
<td>3 4 5 3 3</td>
<td>0.0 0.0 0.209 0.582 0.209 0.0</td>
</tr>
<tr>
<td>3 4 6 3/2 5/2</td>
<td>0.0 0.0 0.195 0.503 0.279 0.021</td>
</tr>
<tr>
<td>1 2 5 6/5 14/5</td>
<td>0.241 0.431 0.232 0.091 0.005 0.0</td>
</tr>
<tr>
<td>1 3 5 3 3</td>
<td>0.016 0.259 0.450 0.259 0.016 0.0</td>
</tr>
</tbody>
</table>

The student-player is given the expected value of each resource for each day subject to such indeterminates as absenteeism, a higher priority job at another location and so forth. The computer will simulate the influence of indeterminate factors on the resource availability by computing the number of workers available each day from a probability density function.

Every project, whether it is a construction project or not, is subjected to different circumstances with differing resource requirements which make the formulation of a generalized probability density function for all jobs of a particular type very difficult. There is
insufficient data to support any generalized hypothesis relevant to this problem anywhere in the literature. For the thesis, then, the probabilities associated with each resource were assumed and are shown in Table 2. Table 2 also shows the assumed expected number and the assumed maximum number of each resource that are to be available each day of the project.

Table 2. Resource Availability Probabilities

<table>
<thead>
<tr>
<th>Resource</th>
<th>Expected Number</th>
<th>Maximum Number</th>
<th>Probability that Number Available Will Be</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Carpenters</td>
<td>4</td>
<td>6</td>
<td>0.0</td>
</tr>
<tr>
<td>Electricians</td>
<td>2</td>
<td>4</td>
<td>0.03</td>
</tr>
<tr>
<td>Masons</td>
<td>2</td>
<td>3</td>
<td>0.09</td>
</tr>
<tr>
<td>Plumbers</td>
<td>2</td>
<td>4</td>
<td>0.03</td>
</tr>
<tr>
<td>Painters</td>
<td>3</td>
<td>6</td>
<td>0.01</td>
</tr>
</tbody>
</table>

The funding variable is the third type of status variable which had to be defined. Available funds were programmed to vary from one week to the next to simulate, on a small scale, the funding difficulties commonly experienced in both governmental and industrial situations. This variable was assumed to vary normally with a given mean and standard deviation. Since the simulation language has the capability of sampling from a normal distribution with a specified mean and standard deviation, only these two parameters had to be specified.
The amount of funds to be made available for each work week is determined at the end of each weekly reporting period and is contained in the program output. The mean and standard deviation for the normal distribution from which this value is drawn are not permanently stored in the program, but are set at the start of the run by the instructor or game monitor. It is left to the instructor to determine how strict the funding constraint will be for each run of the game. Several iterations of a similar but hand-processed game indicate that a relaxed expected value for the funding variable is $1500.00 with a standard deviation of $200.00.

The last type of status variable to be dealt with is the weather variable. Realistic probability distributions for the occurrence of influential weather phenomena can be determined from historical records for a given location at a given time of the year. The construction project described in this paper, however, was not intended to be so specific. The author was interested in introducing a weather variable which would create a more realistic problem atmosphere. Certain activities in the game project were adjudged weather sensitive and all of these were considered equally sensitive. Weather severe enough to prevent work on the sensitive activities was set to occur with a 10 per cent probability each day. The program randomly selects one of the ten integers from 1 to 10 for each day of the week. If a value drawn is less than 10, weather is not considered to influence the project on the day for which the code number was selected. If the value drawn for a particular day is 10, work can not be performed on weather sensitive activities during that day.
Thus far in the procedure, the components, the status variables and their associated parameters, and a general interrelationship between these values have been specified. The next step in the process is to incorporate the fragmented information in the computer program.

A Formulation of the Computer Program

The methodology of computer programming is extensively treated in the literature. The development of the computer simulation game for this study consisted of four commonly recommended steps.

1. Flow Chart
2. Computer Coding
3. Data Input
4. Program Output

It was the experience of this writer, due principally to limited programming experience, that it was advantageous to select the simulation language to be used before attempting the flow chart. This is not to say that an attempt was made to code the program without a flow chart but because of considerable logical differences between the available special purpose languages, it was easier to structure the program in context with a particular language than it was on a more general level.

In approaching this problem, the first consideration concerned the long range disposition of the completed program. It was the intention of the author that the game to be developed would be used in the classroom sessions dealing with PERT/CPM techniques at the Georgia Institute of Technology. Such an arrangement would necessitate an extensive on-line computer capability, a system which is presently being established at the Institute.
Additionally, it is planned that both in-place computers at the school, a Burroughs B5500 and a UNIVAC 1108, will be connected to the on-line network. It was desirable, then, to write the computer program in a language which could be processed on one of these machines.

Several simulation languages including GPSS II, DYNAMO, SIMULA, and, after the study was begun, SIMSCRIPT were available for use. A survey of the major components and logic of these languages* showed that SIMSCRIPT and SIMULA presented the least difficulty with respect to the thesis problem. Since SIMSCRIPT was not yet available at the start of this study, SIMULA was the chosen language.

The selection of SIMULA, an ALGOL-based language, permits the use of the block structure peculiar to the base language. Indeed, the language was specifically designed with the ALGOL block structure in mind. The authors of the language, Ole-john Dahl and Kristen Nygaard, have said that "an important reason why ALGOL has been chosen (as the base language) is that its block structure is similar to what was needed in SIMULA."**

Although it is assumed that the reader is familiar with basic ALGOL concepts, a brief resume of the block concept would serve to clarify the flow diagrams and general program structure which are to follow.


A group of ALGOL declarations, followed by a series of operational statements constitutes an ALGOL block. Each block is enclosed by the reserve words BEGIN and END (Fig. 3).

```
BEGIN
  Declaration Statement
  ...
  Declaration Statement
  Operation Statement
  ...
  Operation Statement
END
```

Figure 3. An ALGOL Block

It is permissible to include within the block certain subroutines, consisting of a series of operation statements, which are not necessarily activated each time the program is run. These subroutines can be given names and can be called into operation as per the program requirements. All variables and constants declared in the set of declaration statements, called the Block Head, can be used without restriction in the subroutines as well as the rest of the program. However, if a particular subroutine has its own set of declaration statements, defining new variables and/or constants for use within the subroutine, this subroutine, with its operation statements, constitutes another block (Fig. 4) and must be enclosed by another BEGIN and END.
The nested block illustrated in Fig. 5 is defined to be on Level Two while the enclosing block is on Level One. If another block were nested within the nested block, this third block would be on Level Three. There is no limit to the number of blocks that can be nested within another block. However, variables and constants declared in the Block Head of the nested block can not be used, under normal circumstances, in the operations of any block on lower-numbered level. Variables and constants, then, are useable in the block in which they are declared and any blocks nested within that block. Thus, in Fig. 4, the variables declared in the nested block could be manipulated only by the operation statements within its own BEGIN and END. Furthermore, if a particular level has more than one block, each block is normally restricted to manipulating data declared in the outer block plus any declared in its own Block Head. Thus, the same dummy variable could
be used in several blocks on the same level without risk of confounding
the computer.

The simple illustration in Fig. 5 will help to clarify the block
structure. In this program there are four blocks on three levels. The
variables declared in each Block Head are shown. The Block One vari-
ables, X, Y, and Z, are common to all blocks and can be used throughout
the program. Block Two has two variables, W and V, which can be used
in the operations of Block Three since this latter block is nested
within Block Two. W and V can not be used ordinarily in the operations
of Block Four. The variable, W, in Block Four is not the same variable
W in Block Two. This illustrates the independence between two blocks
on the same level. The variable T can be manipulated by Block Three

```
BEGIN
  Block One
    Declaration: Variables - X, Y, Z
    Operation Statements
  BEGIN
  Block Two
    Declaration: Variables - W, V
    Operation Statements
  BEGIN
  Block Three
    Declaration: Variable - T
    Operation Statements
    END
  Block Two
    Operation Statements
    END
  BEGIN
  Block Four
    Declaration: Variables - W, S
    Operation Statements
    END
  Block One
    Operation Statements
    END
```

Figure 5. A Three-Level ALGOL Program
operation statements only. The block on Level One can not normally
operate on variables declared in blocks on the higher-numbered levels.

The simulation language, SIMULA, is structured to conform with
the same logic illustrated above. Nested blocks, called Activities,
must be placed immediately after the Block One, Level One (called the
Main Program) declaration statements and before any operation state-
ments.

With these ideas in mind, the flow diagrams presented below can
be more easily understood. For the simulation game, a program on two
levels was required. The block on the first level is the Main Program
and it serves to control the action of the blocks on the second level,
or Activities. A total of 15 second level blocks, plus the Main Pro-
gram, constitute the entire simulation program.

The first 14 Activities (Level Two blocks) manipulate data con-
cerning the 31 PERT activities which are not dummy activities. The
15th Activity (throughout the remainder of this study SIMULA Activities
are denoted by capital A and PERT activities are denoted by a small a)
involves the operations performed on the four dummy activities. Since
operations performed by the first 14 Activities are very similar, the
flow diagram of one will be sufficient to illustrate the logical
sequence of all 14.

The Main Program serves to control the routing of PERT activities
into the proper SIMULA Activities and to discontinue operations in the
subroutines after a specified time lapse. In this particular program,
the Main Program compiles output information and causes this informa-
tion to be printed at the end of the reporting period.
In the flow diagram of the Main Program (Fig. 6), three instructions deserve some comment. In some simulation languages, certain subroutines, usually random number generators, are a permanent part of the language. These subroutines can be called into operation by the use of their associated reserve words or names. Therefore, if the random number generator is a permanent part of the language, the programming chore is eased considerably. In the flow diagram of the Main Program, Instruction (1) and Instruction (2) are instructions to the computer to evaluate variables according to the specified SIMULA subroutine. To accomplish this evaluation, the SIMULA subroutine utilizes the probability information discussed in an earlier section. This probability information is a permanent part of the program which is stored in memory during the compilation phase of the computer run.

A major advantage of SIMULA (and other simulation languages) is that the simulated system time is automatically controlled. Instruction (3) in the flow diagram of the Main Program, HOLD \((x + 1)\), takes advantage of this feature. It is an instruction to the computer to delay any further Main Program operations until the system time is equal to \(x + 1\) time units. The operations scheduled to occur elsewhere in the program before time \(x + 1\) are sequenced automatically in the scheduled order. At time \(x + 1\), operational control returns to the Main Program and the concluding Main Program operations are begun.

The function of the Main Program, then, is to call into operation, or Activate, the various SIMULA Activities according to the input information. It must also evaluate the Weather and Resource Availability status variables since this information must be available
Figure 6. Flow Diagram of the Main Program
to all of the SIMULA Activities. Lastly, the Main Program causes certain data which have been manipulated by the Activities to be printed after it has determined and printed the value of the Funding status variable.

When a particular Activity is activated by the Main Program, the situation is similar to a Procedure Call in ALGOL. That is, the activated Activity's operation statements are the only active statements in the program until the activity is completed or its operation is suspended by an instruction such as the HOLD instruction mentioned above. When an Activity is completed or suspended, program control or operation transfers to the next earliest scheduled event. The next event might be the activation of another Activity or simply a return to the Main Program if the system time has reached \( x + 1 \) units.

As mentioned earlier, the Activity routines for the first 14 Activities are very similar and are typified by any one of them. Fig. 7 is a flow diagram of one of these Activities. Here again, three of the instructions in the diagram should be explained. The first instruction, BASE (I) equals 1, is related to the player-input variable "check." If "check" equals 2, it means that the PERT activity (I) has been completed in a previous reporting period. The variable BASE (I) is then set equal to the code number 75 to permit the point-out routine to show this activity as having been completed at the end of the current reporting period. More will be said about the "check" and BASE (I) variables in later sections of this thesis.

The next two instructions, Instruction (2) and Instruction (3), deal with the subscripted variable named \( \text{Role} \). The variable \( \text{Role} \) is
START
Read PERT Activity Variables

"Next" True
False
Include Reference In Role (i)

Set Daystowork (I) = Accelerated Activity Duration

IF Daysworked (I) = 0

Compute PERT Activity Duration

Compute Completion Day

Set Y = 1

Compute Accelerated Activity Duration

Set Dayswork (I) = Accelerated Act. Duration

IF Role (Y) False

IF Resource Day (Y) Req'd For This Act.

IF Weather False

IF Funds True

False

True

Set Resource Day (Y) = Resource Day (Y) - Resources Req'd For This Activity

Total Cost + Total Cost Activity's Cost

Dayswork (I) = Daysworked (I) + 1

Dayswork (I) = Dayswork (I) + 1

TERMINATE

START
Figure 7. Typical Flow Diagram for SIMULA Activities (14)
actually an array of SIMULA Sets. These sets will be explained after the third and last flow diagram has been presented.

The flow diagram for the operations to be performed on the four dummy PERT activities is brief since the data associated with these activities in itself brief. Dummy PERT activities are inserted in the PERT network to maintain the precedence relationship between other activities. It happens occasionally that an activity can not start until another activity, which can not legally be shown in the network as a predecessor of the first activity, is completed. In this case, a dummy activity is drawn to indicate the precedence relationship. The simple network in Fig. 8 shows that activity 34 can not begin until activity 23 and activity 13 have been completed. But activity 23 is a dummy activity whose time duration is zero and which is inserted in the network to show that activity 34 can not begin until activity 12 and 13 have been completed. The precedence relationship between activity 12 and 34 can not otherwise be shown since PERT networking rules prohibit two activities from starting and ending at common points (events).

Figure 8. A Simple PERT Network with a Dummy Activity
Dummy activities require no resources nor are they sensitive to any project constraint. The dummy activities in the game project are reported in the program output with the other activities. The SIMULA Activity which operates on the four dummy activities simply determines whether the predecessor activities for each dummy activity have been completed and, if so, records the applicable dummy activity itself to have been completed. In the flow diagram (Fig. 9) for the dummy activities, the variable ROLE again appears. It should be evident that this array of SIMULA Sets is the mechanism by which the precedence relationships for the entire PERT network is maintained.

Without attempting to make this thesis a comprehensive explanation of the SIMULA language, certain concepts relevant to the language, such as the Sets, must be clarified now that the flow diagrams have been presented.

SIMULA is a language for describing discrete event systems.* A discrete event system is regarded as a system of component parts whose actions and interactions completely describe the system's behavior. Unlike dynamically nested procedures, the components of the system are on the same dynamic level and operating in quasi-parallel. This means that operations in two different Activities can not be carried out at the same time in the computer. However, two or more Activities can be scheduled to operate at the same system clock time. The simulated time advances only after all operations scheduled to take

---

* A discrete event system is one which can be modeled as series of instantaneous events.
Figure 9. Flow Diagram for SIMULA Activity 15
place at a particular time, or on a particular day, have done so.

In addition to the ALGOL concepts, there are three main SIMULA concepts included in this language. The three are:

1. The Process concept
2. Elements and Sets, which are the means of referencing and grouping Processes
3. The means by which "events" are sequenced in system time.

These concepts are only briefly introduced here and readers are referred to the SIMULA USER'S Manual for a more comprehensive treatment of the subject.

A process is characterized by a "Data Structure" (sequence of declarations) and an "Operation Rule" (sequence of statements). The two aspects are combined into a block in the ALGOL sense (see p. 29). The description of a Process is called an "Activity declaration" and it is similar to a Procedure Call in ALGOL. A distinction is made between the concept of "Activity," which is the class of processes described by one Activity declaration, and "Process," which is one activation of an Activity declaration. For the game described in this report, the PERT activities have been categorized according to common resource requirements and weather sensitivity. From the list of PERT activities which describe the hypothetical construction project (p. 56), it can be seen that there are two activities which require Masons and which are weather sensitive. The data for these two activities are manipulated by one SIMULA Activity which, in the program, is named

Activity BRAVO. There are, then, two Processes to be generated in this Activity during the course of the program.

The actions of a Process are grouped together in active phases, separated by periods of inactivity. Only one Process is active at one time. The currently active Process has complete control since there is no interrupt concept in the language.

An "Event" is defined as an active phase of a Process. It has an associated system time, which is constant during the execution of the Event. Inactive periods can be invoked by "Sequencing Statements" which also specify the order of Events and the associated system times. During an inactive period of a Process, events may occur which are active phases of other Processes.

A Process can be viewed as a self-contained program having its own local sequence control. When a statement invoking an inactive period is executed (such as the HOLD instruction, \( p \)) the local sequence control will stay within this statement during the period of inactivity and will proceed to the next statement at the time of the next active phase of the Process, which is called the "Reactivation Point."

An Event can be scheduled to happen at any time in the system. The Event is the next active phase of some process and there can be at most one Event scheduled for each Process in the system at any one time.

A Process is in one of four possible states depending on whether or not an Event has been scheduled for it and not completed (Table 3).
Table 3. Possible States of a Process

<table>
<thead>
<tr>
<th>States</th>
<th>Event Scheduled?</th>
<th>Reactivation Point?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Suspended</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Passive</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Terminated</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

The current active Process may suspend itself for "T" system time units by executing the statement "HOLD (T)." It may go "passive" by executing the statement "Passivate" (there are no "Passivate" instructions in the subject program). If control leaves the Process through the final "END" of its Operation Rule (that is, the block), or by a "GO To" statement, it becomes "terminated."

The next active phase of a suspended Process normally comes when the stated suspension period has expired. However, no Event can be scheduled for a terminated Process.

Processes are referenced throughout the program by expressions called "Elements." Elements can be declared program variables or simply non-designated references in a "Set." Sets, to simplify, are ordered collections of Elements. Sets must be declared in the sequence of declaration statements. The subscripted variable, Role (J) (pages 31 and 33), is an array of these Sets. Instruction (2) in the flow diagram on page 32 is an instruction to include a reference (or
Element) to the current Process in the appropriate Set. Instruction (3) (page 32), is an instruction to the computer to scan the designated Set and, if there are no references or Elements contained therein, to continue dynamically to the next statement in the Activity. If there are any references to other Processes in the Set, the current Process is suspended for one time unit and the program proceeds to the next scheduled Event. At a later time, when the PERT activity has been completed, another instruction directs the computer to Remove the reference to the completed activity from the Set in which the reference was originally included, thereby reducing the membership in that Set by one. If that Set contains references to all PERT activities which are predecessors to another PERT activity, say activity 5, activity 5 is permitted to begin when the Set membership is zero, or Empty.

As an illustration of the above concept, the simple PERT network on page 37 can be used. In this network, activity 34 has activities 12, 23 and 13 as predecessors. The Main Program would direct that the Activity, say Activity A, which describes the operations to be performed on activity 12 to be activated. When this occurs, activity 12 can be considered as a Process belonging to the class of Processes known as Activity A. Activity A directs that a reference to the Process (activity 12) be included in the Set, ROLE(I). Similar references to the other relevant Processes (activity 23 and activity 13) are included in ROLE(I). During the first unit of system time, all of the network activities are activated and suspended as required. Thus, the Activity, say Activity B, which governs the operations on activity 34, suspends all operations on activity 34 until the Set ROLE(I)
is Empty, which is to say, activities 12, 23, and 13 have been completed. In this example, operations in the appropriate Activities would immediately begin on activities 12 and 13 since there are no predecessor activities for these two activities.

The last SIMULA concept to be discussed here concerns the sequencing of Events. The Sequencing Set (SQS) is an ordered list of Event notices, each representing an Event scheduled, but no completed. Each Event contains a Process (element) reference and a real number called the System Time Reference. The ordering is according to the latter.

The first member of the SQS is called the "current" Event notice. It represents the Event currently in execution. Its system time reference is called the current system time, and its Process reference is to the currently active Process.

When the current active phase (Event) is terminated, the current Event notice is removed from the SQS, and the next Event notice becomes the current one. Control enters the referenced Process at its re-activation point.

Operations on the SQS, that is, removing an Event notice and/or scheduling an Event, are accomplished by Sequencing Statements or Scheduling Statements. The only Sequencing Statement used in this program is the instruction "Terminate" which deletes the Event notice.

---

* It should be mentioned that the use to which Elements and Sets have been put in this program is a very limited one. The power and flexibility of these concepts is much greater than is described above and the interested reader is again referred to the SIMULA User's Manual.
associated with the affected Process and the re-activation point, if any, of the same Process. Scheduling Statements generate an Event notice for a specified process and include it in the SQS. Of the several kinds of Scheduling Statements available, only two were required for the program in this study. The first is the activator statement "Activate" which causes the generation of Event notice for a new Process. This type of Scheduling statement is found only in the Main Program of the system described herein. The other Scheduling statement used in the program is the simple timing clause, HOLD (T). A timing clause specifies a system time reference for the Event notice which determines its position in the SQS. It is normally placed behind other Event notices with the same time reference. Consider, for example, three PERT activities, A, B, and C, scheduled to begin as shown in Table 4.

<table>
<thead>
<tr>
<th>PERT Activity</th>
<th>Scheduled Start Time (Day)</th>
<th>Position in SQS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>1 (1)</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>2 (1)</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>3 (2)</td>
</tr>
</tbody>
</table>

The position of each activity (Process) in the SQS is indicated in the third column with the associated system time reference for each to
begin in parentheses. Suppose, further, that Activity A has been delayed due to weather but that B has proceeded on schedule. A time reference of (2) for activity A is included in the SQS behind the Event notice for activity C. The operations on activity B are performed and, when completed, a time reference of (2) for activity B is included in the SQS behind the Event notice for Activity A (assuming B's duration is greater than one time unit). For both activities, the generated Event notices resulted from a HOLD (1) instruction. In the case of activity A, however, no operations were performed during time 1 whereas the necessary operations on activity B were performed before the HOLD instruction was encountered in the Activity routine. After all Event notices scheduled for the same system time have been removed or rescheduled, the system clock advances to the next discrete time unit and holds at this time until all Event notices scheduled at this later time have been sequentially removed.

A list of all the SIMULA reserved words and functions, with their definitions, that were used in this program are listed in Appendix A.

With the above relationships between the flow diagrams and SIMULA concepts and coding established, the program inputs can be more easily understood. Input data will be required both in the Main Program and the 15 Activities. These data are Exogenous (input) variables that were defined earlier.

The first set of exogenous data is contained on one data card and includes the values of the following variables which are to appear on the data card in the order shown:
1) X: the integer number of days in the week during which work will be performed on the project. 0 ≤ X ≤ 6, usually X = 5 or 6.

2) N: the integer number of activities in the project—in this case, N = 35.

3) FUNDS: the amount of money available to be used during the work period (a real variable determined by the program at the end of the previous work period or given in the problem for the first work period).

4) WE: an integer value denoting the first, second, etc., week of work. WE = 8.

5) BEG: the mean value of the normal distribution from which the future period's Funds will be selected—a real number.

6) EN: the standard deviation of the Normal distribution from which the future period's Funds will be selected—a real number.

7) ST: the first integer number in the series of integer numbers, all of which have equal probability, and from which will be selected the Weather code number for each day of the work week.

8) FI: the last integer number in the series of integer numbers explained in No. 7 above.

9) U1, U2, U3, U4, U5, U6: integer numbers to be used in the random number generators throughout the Main Program.

All of the exogenous variables defined above, with the exception of X, will be given to the player. The number of days to be worked in any week is a game variable.

Each PERT activity will have two data cards associated with it. The first data card will contain only one code number which assigns a value to the variable named DIRECTION. The Main Program reads the value of DIRECTION and activates Processes according to that number.
The value of DIRECTION is equal to or greater than one and equal to or less than 15. Each PERT activity is assigned one of these numbers and they are given to the player.

The next set of exogenous data is read by the various Activity routines and consists of the data associated with each PERT activity. The data for each PERT activity includes the values of the following variables and this data appears on the second data card for each activity in the order shown:

1) I, J, K: integer numbers which control the precedence relationship of the activities explained above. These values, as with the variable DIRECTION, are given for each activity (see page 56).

2) CHECK: a variable which serves two purposes and which is assigned a value of zero, one or two. If equal to one it indicates that the player wishes to accelerate the work on an activity. Used in this way, CHECK is a game variable. If CHECK is set equal to two, it means that the activity was completed in an earlier period. At all other times CHECK is set equal to zero (see page 31).

3) DAYSWORKED (I): the number of days that the activity has been worked on during a previous period(s).

4) DAYSTOWORK (I): the number of days remaining to be worked on an activity. This value is also a function of previous work periods.

5) CARPS, ELECS, PLUMS, PAINS, DRYs, or LABS the number of carpenters, electricians, plumbers, masons, painters, dry wall men or laborers required to perform the work on an activity for one day. These numbers may vary if the player elects to expedite the activity duration. The second data card for each activity will carry values for only those resources required by the activity in question.
6) **BETA:** the code number of the *beta* distribution from which the activity durations will be drawn. This number will determine whether the duration time is selected from the "normal" time estimates or from the "expedited" time estimate. This is a game variable and an integer number.

7) **U7, U8:** integer numbers to be used in the random number generators in the Activity.

Of these exogenous variables, those defined in No. 1 and No. 7 are given to the player. The others are evaluated as a result of the actions of previous periods or they are arbitrarily evaluated as game variables by the player according to the rules of the game. Variables in categories No. 2, 5 and 6 are or can be game variables.

For the four dummy activities, the list of exogenous variables is exactly the same as items No. 1, 2, 3, and 4 in the list above. Of course, for these activities, the DAYSWORKED(I) and DAYSTOWORK(I) will always be zero but are included to maintain a consistent print-out display. For these four activities, the value of "CHECK" only has meaning if it equals two. Here, again, the I, J and K variables will be given.

There are, then, 14 exogenous variables in the first set of input variables. Each PERT activity has two data cards, the first of which contains one integer number between the numbers 1 and 15, inclusive, and the second of which has from 6 to 12 numbers depending on the activity.

The last step to be accomplished in the formulation of the computer program is to specify the output variables. The program output includes a listing of the variable resources that were available each
day of the week and how many, if any, were used each day. A list of all the PERT activities with the number of days already worked and the number of days yet to be worked on each will also be printed. Also, the projected completion date, to fall within two calendar months of the project's beginning, will be available to the player. Activities which have been completed will be so identified instead of being given a projected completion date. Lastly, the project's accumulated cost during the week covered by the report and the funds available for the next period's work are presented in the output.

This concludes the list of input and output variables except for an array of 22 integers that are determined within the program and, since they are the basis for the algorithm which computes the projected completion date and not exogenous in the sense defined earlier in this report, they must be carried over from one reporting period to the next. These 22 numbers will be set equal to zero for the first week's run. At the end of each reporting period, a string of 22 numbers will be printed at the bottom of the print-out. The player will be instructed to transfer this string exactly as shown to the first data card.

With the flow diagrams, the computer coding, the exogenous variables, and the endogenous variables defined, the computer program was generally formulated. All that remained was to actually write the program and explain the algorithms used in the final result.

After the program was written, it was then necessary to test its validity.
A Test of the Program's Validity

The nature of the simulation model described in this thesis does not require that it be shown to be a rigidly realistic model of an operating construction project. Rather, since this model is to be used as an instructional aid, it is necessary to ensure that it operates in a manner which is consistent with the game's description and instructions. The test should consist of a number of "plays" of the entire game, varying the game constraints, such as the weather probability, the requirements, and/or the resource probability density functions. A thorough check for the correct operation of all of the random number generators is essential.

If it can be shown that the exogenous variables do not cause confusion among inexperienced users and, likewise, if the presentation of the endogenous variables in the computer output is easily understood within the game's framework, the computer program, in conjunction with the supplemental game forms, can be considered to be a valid and useful PERT training device.

The Development of a Supplemental Training Form

A very important part of any game is a set of instructions and supplemental forms which enable potential players to participate in the game with minimal difficulties. The last step in the procedure to develop the PERT simulation game is to set the stage, as it were, on which the game is to be played. After a description of the game setting, a set of clear, brief game instructions should follow. Easy-reference forms which serve to organize the game information must be prepared for the player. A game brochure has been designed which
includes these three information components. With this brochure, a UNIVAC 1108 computer, and a pencil, the PERT simulation game can begin.

An explanation of the step-by-step procedure which was followed to develop the PERT game has been presented in this Chapter. The basic model, the program variables, the computer coding, and the supplemental game forms have been defined in this section. In the next Chapter, the final computer program is presented with explanations of the subroutine algorithms. The completed game brochure can also be found in the next chapter. The last chapter will discuss some improvements on and extensions of the game which might be considered for future development.
CHAPTER III

RESULTS

In this chapter, having identified and/or defined all of the terminology and variables in the preceding chapter, the resultant computer program is related to the game project in detail. Certain subroutines, such as the one which computes the projected completion date for each PERT activity, are presented and explained with illustrations taken from the game network. The game brochure is self-explanatory and can be found at the end of this chapter.

The PERT Project

The game project concerns the construction of a house. It is to be accomplished utilizing idle company resources as they become available depending upon the requirements of other, more critical, company contracts. All of the pertinent data, including the project network, are given to the project manager.

As can be seen in Figure 10, the project network is an Activity-On-Arrows (AOA) system which means that the activities are graphically represented by the arrows. The activities are numerically identified by one of the numbers between 1 and 31, inclusive. Four other activities, defined in the previous chapter as dummy activities, are numbered 96 through 99 and are so numbered for ease of recognition both on the network itself and in the computer print-out. It will be noticed that there are other dummy activities which are not numbered and which will
Figure 10. PERT Network for Game Project
not appear anywhere in the game procedures. The reason that four of these dummy activities (perhaps a better name for these four would be artificial activities) were included in the computer program and the others were not is because a correct precedence relationship could not be maintained in the program without them. This will become clear when the variables, I, J, and K (Table 7) are explained below. These four activities can be considered as zero-time, undefined activities and they should present no problems to the player.

Table 5 outlines the normal resource requirements for each activity for one day of work. The resource levels shown are those upon which the PERT time estimates are predicated. The time estimates for each activity are one of the ten sets of time estimates discussed earlier (Table 1, page 20) and are given in the column BETA. Thus, for activity 1, the time estimates are a = 1, m = 1, and b = 2; for activity 13, the time estimates are a = 1, m = 2, and b = 4; and so forth.

Also shown in Table 5 are the activity code numbers, or the variable DIRECTION (page 52). As will be recalled, this number directs the program to Activate the proper SIMULA Activity. It can now be seen where these numbers came from and why it was necessary to have 15 SIMULA Activities. PERT activities 1 and 15 have the code number 8. A study of the resource requirements table indicates that these two activities are both sensitive to weather and that they both utilize laborers only. Activity 3 is unique in that it is the only activity which uses electricians and laborers and is weather sensitive.
<table>
<thead>
<tr>
<th>Activity Number</th>
<th>Activity Description</th>
<th>Code No.</th>
<th>Weather Sensitive</th>
<th>Carpenter</th>
<th>Electrician</th>
<th>Mason</th>
<th>Plumber</th>
<th>Wall Men</th>
<th>Painter</th>
<th>Grader</th>
<th>Labor</th>
<th>BETA</th>
<th>Accelerate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dig Footings</td>
<td>8</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Lay Blocks</td>
<td>2</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Elec. in Grnd.</td>
<td>3</td>
<td>Yes</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>Plum. in Grnd.</td>
<td>4</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>Frame</td>
<td>1</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>Frame Roof</td>
<td>1</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Felt Roof</td>
<td>1</td>
<td>Yes</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Cornice</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Install Windows</td>
<td>1</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Run Plumbing</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td>Set Tubs</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Run Elec.</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Run Heat</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Prime Cornice</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Lay Brick</td>
<td>2</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Hang &amp; Finish Sheetrock</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>17</td>
<td>Install Gutters &amp; DMSPTS</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Grade &amp; Pour</td>
<td>10</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>Install OS Stairs</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Finish Kitchen</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>21</td>
<td>Install Floor</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>22</td>
<td>Mill Work</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>23</td>
<td>Tile Bath</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>Finish AC &amp; Heat</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>25</td>
<td>Finish Elec.</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>26</td>
<td>Plumb. in Bath</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>27</td>
<td>Paint Interior</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>28</td>
<td>Paint Floors</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>29</td>
<td>Finish Exterior</td>
<td>14</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>30</td>
<td>Paint Exterior</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Clean</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Activities 12, 13, 25 and 26 have the same resource requirements as activity 3 but they are not weather sensitive and are, therefore, grouped differently, having a code number 7 while activity 3 has a code number 3. The code numbers correspond to the SIMULA Activities named Alpha, Bravo, etc. Therefore, activity 1 will be a Process in Activity Hotel, the eighth Activity; activity 2 will be a Process in Activity Bravo, the second Activity; and so on.

Certain of the activities, 18 in all, may be accelerated by allocating more resources to these activities at the discretion of the player. The activity time estimates are diminished accordingly. Table 6 is a listing of those activities which the player is permitted to accelerate.*

The resource requirements and accelerated time estimates for these activities, if the player elects to expedite any of them during the course of the play, are also shown. The Beta column in Table 6 identifies the accelerated time estimates for an activity on which no work has yet been done. It is possible, however, that the player may wish to expedite an activity for which some work has been done. This is permissible if the activity involved has more than one day of work remaining. In this event, the player will be directed to select from among the ten sets of time estimates that set which is the most favorably distributed and which has a most likely time estimate of one day less than the number of days remaining to be worked on the activity in

* Table 6 also indicates which activities may be expedited, either initially or at some later time, by an X in the ACCELERATE column for those activities so affected.
Table 6. Activities which Can Be Accelerated

<table>
<thead>
<tr>
<th>Activity Number</th>
<th>Weather Sensitive</th>
<th>Carpenter</th>
<th>Electrician</th>
<th>Mason</th>
<th>Plumber</th>
<th>Men</th>
<th>Painter</th>
<th>Labor</th>
<th>BETA</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Yes</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Yes</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>Yes</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>14</td>
<td>Yes</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>Yes</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>20</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>25</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>30</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
</tbody>
</table>

NOTE: (1) Use data from this table only if *accelerating* an activity.

(2) "CHECK" must be set equal to "1" when using data from this Table.
question. Thus, for example, if activity 16 has been determined (by the computer) to have a duration of five days, only one of which has been completed at the end of a reporting period, it will have four days of work remaining to be completed during the next week. The variables, DAYSTOWORK(16) and DAYSWORKED(16), would be set equal to four and one, respectively, for the next week's play. If it is desired to expedite this activity by one day (the maximum rate of acceleration), the player should select the set of time estimates which has a most likely time estimate of three and which is the most favorably distributed set of estimates about this most likely time. This is Set number five according to Table 1. The input values for DAYSTOWORK(16) and DAYSWORKED(16) remain four and one, respectively, but the variable CHECK is set equal to one signifying to the computer that an accelerated duration is requested. The computer then selects a new value for the time remaining from the specified (accelerated) beta distribution and replaces the DAYSTOWORK(16) variable with this new time. The player, in order to accelerate an activity, then, needs to set CHECK equal to one, select the correct beta distribution and set the activity resource level equal to the numbers shown in Table 6. It should be evident that activities are permitted to be accelerated by only one day each reporting period. This feature was incorporated into the game since most of the activities are of very short duration. Furthermore, accelerations of more than one day at a time would tempt the player to concentrate his efforts on one troublesome activity rather than on the project as a whole.
Table 7 is a listing of the first four input variables for each activity. The first column gives the values of DIRECTION, or the code number, which is the only number on the first of the two data cards for each activity (page 45). The first three numbers on the second data card for each activity are the values for I, J, and K as shown in Table 7. As mentioned earlier, I, J, and K are control numbers which maintain the order of precedence among the activities as depicted on the PERT network diagram. I is simply the number of the activity itself, except for the four dummy or artificial activities which have been numbered in sequence after the first 31. The number J identifies the SIMULA Set, ROLE(J), in which a reference (or Element) to the Process (activity) numbered I is to be placed. The number K identifies the SIMULA Set, ROLE(K), which is the predecessor set. This Set is scanned by the Activity routine and, if Empty, permits the Activity routine to record a day's work performed on activity I, other conditions being fulfilled. Therefore, when a Process for activity 7 is generated in Activity Alpha, it is seen that T equals seven for this activity. A check of the PERT network shows that activity 7 can not begin until activity 6 is completed. A reference to activity 6 has been placed in the SIMULA Set, ROLE(5) and if that Set is Empty, work on activity 7 may be accomplished. A reference to activity 7 has been placed in the Set, ROLE(6). Activities 14, 13, and 12, all shown in Figure 11 to succeed activity 7, should all have a value of K equal to six. A check of these values in Table 7 confirms this to be the case.
Table 7. Input Variables for Each Activity

<table>
<thead>
<tr>
<th>Activity Number</th>
<th>(1) Direction</th>
<th>(2) I</th>
<th>(3) J</th>
<th>(4) K</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>6</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>7</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>8</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>9</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>96</td>
<td>15</td>
<td>32</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>6</td>
<td>11</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>12</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>13</td>
<td>7</td>
<td>13</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>14</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>15</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>16</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>97</td>
<td>15</td>
<td>33</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>17</td>
<td>9</td>
<td>17</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>18</td>
<td>14</td>
<td>11</td>
</tr>
<tr>
<td>19</td>
<td>10</td>
<td>19</td>
<td>15</td>
<td>12</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>20</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>21</td>
<td>5</td>
<td>21</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td>22</td>
<td>5</td>
<td>22</td>
<td>16</td>
<td>13</td>
</tr>
<tr>
<td>23</td>
<td>5</td>
<td>23</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>98</td>
<td>15</td>
<td>34</td>
<td>14</td>
<td>17</td>
</tr>
<tr>
<td>99</td>
<td>15</td>
<td>35</td>
<td>18</td>
<td>17</td>
</tr>
<tr>
<td>24</td>
<td>11</td>
<td>24</td>
<td>19</td>
<td>13</td>
</tr>
<tr>
<td>25</td>
<td>7</td>
<td>25</td>
<td>18</td>
<td>13</td>
</tr>
<tr>
<td>26</td>
<td>7</td>
<td>26</td>
<td>18</td>
<td>13</td>
</tr>
<tr>
<td>27</td>
<td>6</td>
<td>27</td>
<td>18</td>
<td>19</td>
</tr>
<tr>
<td>28</td>
<td>13</td>
<td>28</td>
<td>20</td>
<td>18</td>
</tr>
<tr>
<td>29</td>
<td>12</td>
<td>29</td>
<td>21</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>14</td>
<td>30</td>
<td>22</td>
<td>21</td>
</tr>
<tr>
<td>31</td>
<td>12</td>
<td>31</td>
<td>22</td>
<td>14</td>
</tr>
</tbody>
</table>
Now consider activity 17 which is the successor of the four activities 9, 11, 12, and 13. For activity 17, I equals 17. References to activities 11, 12, and 13 have been included in the Set, ROLE (9). A reference to activity 9 has been included in the Set, ROLE (8). But a reference to activity 9 can be placed in only one Set by virtue of the program's construction.* Therefore, the dummy activity 96, which graphically portrays the precedence relationship, must be introduced to the computer program in order that some reference to activity 9 is included in ROLE (9). Activities 96 and 16 scan ROLE (8), which must be Empty before either can proceed. But activity 96 has zero time duration and the reference to this activity is withdrawn from ROLE (9) at the same system time that activity 9 is completed, thereby permitting activity 17 to proceed, assuming the other three activities, 11, 12 and 13, have also been completed and the references to these activities have been withdrawn from ROLE (9). The activities 97, 98, and 99 have been included in the program for similar reasons. A brief study of the PERT network will verify the necessity to include the latter three dummy activities in the computer program.

It is not necessary that the player understand the function of the variables I, J, and K. He is merely required to transfer the given values to the data card as instructed.

The only variables that remain to be covered are the subscripted U's (pages 43 and 45). These are not game variables in the sense that their value is determined by the player. All SIMULA random number

* See Chapter 4, page 86.
generators establish a pseudo-random number stream from which a basic drawing is obtained. The probabilities given in the program are converted as required by the language subroutines for the purposes of establishing the desired pseudo-random stream. However, an integer value must be specified to satisfy the requirements of the SIMULA algorithm. Therefore, each random drawing in the program must have a specified integer value available in order to accomplish the basic drawing. Adjustments to these variables have the effect of altering which number in the stream is chosen. The values of all \( u_i \) must be integers and they are given as input data for both the Main Program generators and the Activity generators.

All of the variable input data has now been explained. The game brochure (page 69) is a synopsis of this information without most of the programming explanations.

The Computer Program

Using the flow diagrams (Chapter 2) as a starting point, a computer program has been written which satisfies all of the game requirements.

As explained previously, the program consists of the Main Program and 15 SIMULA Activities. The declaration statements for the Main Program (Appendix B) are immediately followed by the 15 Activities and then, finally, the operational statements for the Main Program. The SIMULA compiler conveniently annotates the block divisions.

Considering only the Main Program for the moment, it is seen that the declaration statements appear exactly as one would expect for an ALGOL program. There is a declared integer array for each variable resource and the DAYSWORKED(I) and DAYSTOWORK(I) variables. Real arrays have been declared for the probability distributions and these distributions are defined immediately following the last Activity block. The integer arrays BASE (1:35) and A(1:35) are clarified below.

After reading the Main Program input data, the Main Program determines the availability of each variable resource for each day of the work week (x Days). The weather for each day is also evaluated. Since the original resource levels are decremented as required, these levels are printed before any decrementing operations are performed.

The Main Program then begins to read the DIRECTION values. At this point system time equals one. All of the DIRECTION cards are read before the system time advances. After all the DIRECTION cards have been read (all PERT activities are now Processes in some one of the 15 SIMULA Activities), the system time is still one when the instruction HOLD (x + 1) is encountered. Operations in the Activities continues until Time (x + 1) at which time the Main Program is reactivated at its reactivation-point, that is, the instruction immediately following the HOLD instruction.

After the unused resources for each day of the reporting period have been printed, the subroutine which lists the activities is entered. This listing routine must not only list the DAYSWORKED(I), DAYSTOWORK(I), and the PROJECTED COMPLETION DATE (or COMPLETED if the activity in question has been completed) for each activity but it must
also list the activities in the order shown in Table 8 with the dummy activities numbered as shown. A flow diagram (Fig. 11) may help to clarify this subroutine.

The variable, BASE(I), is the value or date of Projected Completion for each activity. It is extremely remote that this project will endure longer than 61 calendar days. The project has been designed to begin on June 1, 1968. The value of BASE(I) is an integer number. If it has been set equal to 75 (page 31), the listing subroutine will print COMPLETED instead of a Projected Completion Date. If BASE(I) is less than 31, the Projected Completion Date will appear in the print-out as JUNE(BASE(I)), 1968. If BASE(I) is greater than 30, the Projected Completion Date for activity I will be July (BASE(I) - 30), 1968.

The functions of the variables F and G in the listing subroutine are obvious.

Lastly, the Main Program prints the accumulated cost of the reporting period, the funds available for work during the next reporting period, and the values for the A array (see Fig. 12). The latter values are determined within the Activity routines, the next components of the computer program to be covered.

Considering only Activity Alpha (Appendix B), which is typical of all Activities except Activity Oscar, it is seen that the Activity satisfies the definition of an ALGOL block.

After reading the input data, the Activity checks the value of the variable CHECK. If CHECK equals two, the variable BASE(I) is set equal to 75 to indicate that the activity has been completed. Notice
Figure 11. PERT Activity Listing Subroutine (Main Program)
## Resource Availability Week 3

<table>
<thead>
<tr>
<th>Activity</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thurs</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carpenters</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>Masons</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Electricians</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Plumbers</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Painters</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

## Resources Not Used Week 3

<table>
<thead>
<tr>
<th>Activity</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thurs</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carpenters</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Masons</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Electricians</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Plumbers</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Painters</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

## Activity

<table>
<thead>
<tr>
<th>Activity</th>
<th>DAYSWORKED</th>
<th>DAYSTOWORK</th>
<th>PROJECTED COMPLETION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>COMPLETED</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>COMPLETED</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>2</td>
<td>10 JULY 1968</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>3</td>
<td>12 JULY 1968</td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>1</td>
<td>10 JULY 1968</td>
</tr>
</tbody>
</table>

Cost to Date: $950.00

Funds Available Week 4 $1300.00

Figure 12. Typical Program Output for a Six-Day Workweek
that the values for $\text{DAYSWORKED}(I)$ and $\text{DAYSTOWORK}(I)$ have been recorded by the READ statement and will be available for the print-out. Of course, $\text{DAYSTOWORK}(I)$ will be zero in this case.

Next, a reference to the Process is included in the Set, $\text{ROLE}(J)$ (page 53). If $\text{DAYSWORKED}(I)$ is equal to zero and $\text{CHECK}$ is equal to zero, an activity duration for activity $I$ is randomly selected. PERT activity durations will be drawn for all activities with a $\text{DAYSWORKED}(I)$ and $\text{CHECK}$ equal to zero (except for the four dummy activities) at the start of each week. It is not expected that the higher numbered activities will have any work done on them during the earliest weeks of the project. This means that for these activities several durations (not necessarily the same), for each week which succeeds a reporting period during which no work on the activity(s) was performed, will be drawn during the course of the project. These temporarily assigned activity durations are for the purpose of maintaining an estimate of the project's termination date. The actual activity duration is considered to be undetermined until work on that activity has actually started.

If the variable $\text{CHECK}$ is set equal to one by the player, the procedure to select an accelerated activity duration is performed by the computer (pages 53-55).

The last portion of Activity Alpha scans the available resources and determines whether predecessor activities have been completed. After computing the value of the variable $\text{BASE}(I)$, the Activity reduces the resource availability of each resource by the amount required by the activity in question and increments and decrements $\text{DAYSWORKED}(I)$.
and DAYSTOWORK(I) as required.

The algorithm used to compute the value of BASE(I) is a function of DAYSTOWORK(I), X (the number of days to be worked during the week), W (a counter that identifies which day of the week is being considered), and A(K). Initially all A(K) are set equal to zero. The subscript K is also used to maintain the network precedence relationship (page 56). The algorithm is:

\[
\begin{align*}
\text{IF } & \text{DAYSTOWORK}(I) \leq (X + 1 - W) \text{ THEN} \\
& \text{BASE}(I) = A(K) + \text{DAYSTOWORK}(I) \\
\text{ELSE} \\
\text{IF } & \text{DAYSTOWORK}(I) \leq (2X - W) \text{ THEN} \\
& \text{BASE}(I) = A(K) + \text{DAYSTOWORK}(I) + (6 - X) \\
\text{ELSE} \\
\cdot \cdot \cdot \\
\text{IF } & \text{DAYSTOWORK}(I) \leq (CX - W) \text{ THEN} \\
& \text{BASE}(I) = A(K) + \text{DAYSTOWORK}(I) \\
& \quad + (C - 1)(7 - X) - 1; \\
& \text{FOR } C = 2, 3, \ldots
\end{align*}
\]

The general form applies for all cases except for the situation in which the work remaining can be completed before the end of the week or reporting period.

To illustrate the operation of this routine, consider activity 1. The completion date for activity 1 is the value of BASE (1). The K value associated with activity 1, from Table 7, is 0. A(0) is initially
zero. Assume that activity 1 has a computed duration of one day and that conditions are such that work may begin on the first day. The counter W is one. It is seen that since \( \text{DAYSTOWORK}(1) \) is less than \( X + 1 - W \) (for \( X = 5 \)), that the completion date for activity 1 (\( \text{BASE}(1) \)) equals one. Suppose conditions do not permit work to proceed on this activity on the first day and its start is delayed until the second day. \( A(0) \) is incremented by one by the last instruction in the FOR statement. Thus, \( A(0) \) is set equal to one, W becomes 2, and the procedure is begun anew. \( \text{DAYSTOWORK}(1) \) still equals one and is still less than \( X + 1 - W \). \( \text{BASE} (1) \), then, equals two.

The routine then compares the value \( \text{BASE} (1) \) with \( A(J) \) or, in this case, \( A(1) \). Now \( A(1) \), by an inspection of Table 8, is the \( A(K) \) value for activity 2.

### Table 8. Resource Costs

<table>
<thead>
<tr>
<th>Resource</th>
<th>DAILY WAGES</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Straight Time</td>
<td>Overtime</td>
<td></td>
</tr>
<tr>
<td>Masons</td>
<td>$36.00</td>
<td>$54.00</td>
<td></td>
</tr>
<tr>
<td>Electricians</td>
<td>40.00</td>
<td>60.00</td>
<td></td>
</tr>
<tr>
<td>Plumbers</td>
<td>36.00</td>
<td>54.00</td>
<td></td>
</tr>
<tr>
<td>Carpenters</td>
<td>32.00</td>
<td>48.00</td>
<td></td>
</tr>
<tr>
<td>Dry Wall Men</td>
<td>32.00</td>
<td>48.00</td>
<td></td>
</tr>
<tr>
<td>Painters</td>
<td>32.00</td>
<td>48.00</td>
<td></td>
</tr>
<tr>
<td>Grader</td>
<td>32.00</td>
<td>48.00</td>
<td></td>
</tr>
<tr>
<td>Laborers</td>
<td>20.00</td>
<td>30.00</td>
<td></td>
</tr>
</tbody>
</table>
If the value of BASE (l) is greater than A(l) (which is initially zero), the value of A(l) is replaced by the value of BASE (l). In this way the starting value of A(K) for each activity is updated by its predecessor activities. The incrementation of A(K) by the routine itself also insures that the value of A(K) is current if the activity's predecessor activities are complete but it is delayed for other reasons.

Consider activity 17 which uses A(9) to compute its projected completion date or BASE (17). A(9) takes on the value of the latest projected completion date of activities 96, 11, 12 or 13 by virtue of the comparison of BASE(I) and A(J) in the respective routines. A(9) is set to some value greater than zero before the Process for activity 17 is generated. After generation, the Process for activity 17 itself increments A(9) each day for this activity. After activity 17 has been generated, the comparisons between the value of A(9) and the respective BASE(I) by the predecessor routines will not result in the replacement of A(9) by one of these values since the Process for activity 17 will be correctly incrementing A(9).

After the projected completion day has been computed, the Activity scans the appropriate SIMULA Set for incompleted predecessor activities and insures that sufficient resources are available before work on the activity can proceed. If the appropriate SET is Empty and sufficient resources are available, the resource levels for that day are decremented as required and the cost is computed and added to the total cost for the week.

The cost computations are calculated from four basic rates (Table 8, p. 66). This is the defined cost schedule and it is used as
presented to compute incurred labor costs for any of the first five days of the work week. If a sixth day is worked, overtime rates are used. These rates are one and one-half times the basic rates.

After the cost computations, the variables DAYSWORKED(I) and DAYSWORK(I) are adjusted. If DAYSWORK(I) equals zero, the activity has been completed. The reference to this activity is removed from ROLE(J), BASE(I) is set equal to 75 and the Process is terminated.

After the Process has been terminated, the data associated with it are permanently fixed for the duration of the game. The value of A(K) associated with that Process (or activity) is no longer incremented either by its own routine or its predecessor routines which are themselves necessarily terminated if the Process in question is terminated. The values for the array of A(K)s are printed at the bottom of the computer print-out and must be reintroduced to the program each simulated week to maintain calendar consistency from one week to the next.

Activity Oscar, the 15th activity, is the routine for the four dummy activities. This Activity is an abbreviated form of the others and after including a reference to itself in the proper Set, continues to check its predecessor Set each day until that Set is Empty. When its predecessor Set is Empty, it adjusts the A(J) for its successor activity, if required, and proceeds to termination.

This concludes the discussion of the computer program itself. The last item required to make this computer program a functional training device is a game brochure which will outline the rules of the game and how it is to be played using the computer.
A PERT Simulation Game

The game described in the following pages concerns a construction project which is to be managed using standard PERT methodology. The game instructions describe the game's scenario and define all initial conditions. Game constraints are also defined and these constraints are fixed for the duration of each play of the game. The object of the game is to maximize profits relative to the game project.

The game can be played by any number of players acting either individually or in a group.
The Play of the Game

Your company has contracted to build a house for one of the town's leading citizens. You have been selected to manage this project. The preliminaries of the project, such as the network and the PERT time estimates, have been prepared for you. You are expected to complete the job so as to maximize company profits.

The project is to begin on Monday, June 1, 1968, and if not completed by July 31, 1968, the company is in default and the cost accrued up to that point is a total loss.

If the project is completed in 35 calendar days or less, the company will receive a bonus of $2000.00. The bonus is reduced by $200.00 each day the project continues beyond the 35th day until it is exhausted.

There are eight different kinds of resources required by this project. Of the eight, laborers, dry wall men, and graders are not critical and will be available as required. The other five (carpenters, masons, electricians, plumbers, and painters) are critical and their availability will vary from one day to the next.

During the course of the project, there is only one holiday. The Fourth of July falls on a Saturday and work on that day is not permitted.

* The Table numbers and Figure numbers refer to Tables and Figures throughout Chapter Three of this report. When this brochure is prepared for student use, these Tables and Figures would all be included in this Brochure and renumbered as required. Also, the page references for the Tables and Figures would be deleted.
Certain activities can be expedited by assigning additional resources to these activities as explained below. Additionally, it is permissible to work on all Saturdays except the Fourth of July, but not on Sundays. No other kind of overtime is permitted.

You are to adjust the game variables (described below) at the end of each simulated week of work as required. The status of each activity will be made available to you at the end of the week by the Status Report. The Status Report will also include the cumulative weekly cost and the funds available for the next week.

The UNIVAC 1108 computer compiles the data for the Status Report. The computer program determines the availability of the critical resources mentioned above. The actual activity durations, computed from the time estimates that are given, are also determined by the computer. Lastly, the computer randomly selects the days on which it will rain which in effect delays certain activities that have been identified as Weather Sensitive. These computer data simulate project factors which are normally beyond your control.

Costs are computed according to the schedule shown in Table 8 (page 66). Overtime rates are charged for any work done on Saturday. Normal rates are used in computing the activity costs for any other day of the week. In addition, there is a fixed operating cost of $100.00 per day.

You are expected to use the variable and given data to manage the project in such a way as to maximize profit subject to the given time and cost constraints.
Explanation of Tables

The network of activities for this project is shown in Figure 10 (page 50). The activities are listed, with the job description for each, in Table 5 (page 52). Table 5 also indicates which activities are Weather Sensitive and the normal resource requirements for each. The column named BETA corresponds to one of the ten sets of time estimates in Table 9.

Table 9. Activity Time Estimates

<table>
<thead>
<tr>
<th>BETA</th>
<th>DAYS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

The set of time estimates associated with the given BETA value is the estimated optimistic, most likely, and pessimistic times for the activity. An X in the column ACCELERATE indicates that the activity can be accelerated. Those activities not so marked can not be accelerated.

As an illustration, consider activity 12. It is seen that this activity involves the installation of the electric wiring in the house.
The activity, conducted indoors, is not influenced by weather. The activity normally requires the services of two electricians and two laborers each day. The estimated duration of this activity is given by the value of BETA which equals four. Thus, for activity 12, \( a = 1, m = 2, \) and \( b = 4 \). This activity can be accelerated. The other value given, the Code Number, equals 7 and will be explained below. Table 6, then, identifies the resource requirements and activity durations under normal, or non-accelerated, circumstances.

The activities numbered 96, 97, 98, 99, are dummy activities which utilize no resources and are of zero time duration. They were included in the list of activities to meet the requirements of the computer program. These four activities have only the Code Number 15 associated with them.

Table 6 (page 54) is a listing of those activities which may be accelerated and which were identified in Table 5 by an X in the ACCELERATE column. The resource schedule shown in Table 6 is the schedule used whenever an activity is to be expedited. The BETA column in this table identifies the accelerated time estimates for those activities to which the additional resources have been allocated but on which no work has yet been performed. Thus, if it is desired to shorten the expected duration of an activity before work has begun on that activity, the resource level shown in Table 6 would be allocated to that activity and the time estimates associated with the BETA value shown would be the expedited time estimates.

It is possible to expedite an activity after some work has been performed on the activity. In this case, a value for BETA is chosen.
such that its corresponding most likely time estimate is one less than the number of days remaining to be worked on the activity in question. It is not permissible to select a value for BETA which has a most likely time estimate whose value is other than one day less than the number of days remaining to be worked on the activity. Neither is it permissible to expedite an activity with only one day remaining to be worked. (It should be pointed out, since the value of BETA simply directs the computer to perform a probabilistic computation based on the associated time estimates, that you are not guaranteed that the activity will be actually expedited. But this simulates actual project uncertainties. Moreover, the possibility exists that when an activity is accelerated, the duration may be shortened by more than one day in addition to the possibility that it may not be reduced at all. The most probable event, should an activity be accelerated, is that it will be shortened by one day.) Again, since the activity is being accelerated, the resource schedule in Table 6 would be used.

As an example, suppose you deem it necessary to expedite activity 16. Activity 16, under normal circumstances, would have a BETA equal to 8 (Table 5) which means that for this activity, $a = 3$, $m = 4$, and $b = 6$. But, you may wish to shorten the estimated duration. If no work has yet been performed on this activity, you would use the values in Table 6 wherein the resource requirements are higher but the associated BETA equals 5, yielding the shorter time estimates of $a = 2$, $m = 3$, and $b = 4$. Now suppose activity 16 had been started using normal resources and at the end of the week the Status Report shows that this activity has had one day of work completed with three days yet to be
done. If, for some reason, you wish to reduce the number of days still to be worked, you would use the resource schedule in Table 7 and select a BETA which has as its most likely time estimate a value of one less than the days remaining to be worked and which is the most favorably distributed about this time. In this case, you would select a BETA equal to three since its most likely time estimate is two and it is the most favorably distributed set of time estimates with a most likely time of two. Similarly, if activity 16 had had two days of work performed at the end of the week and you wished to reduce the remaining time to one day instead of two, you would select a BETA equal to one.

Table 10 presents the expected number of resources that will be available for each day of work.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Expected Number Available Each Day</th>
<th>Maximum Number Available Each Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carpenters</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Masons</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Electricians</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Plumbers</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>Painters</td>
<td>3</td>
<td>6</td>
</tr>
</tbody>
</table>

A maximum number available is also given. When scheduling which activities are to be accelerated, you may not attempt to use more than maximum number of any resource on a given day. The expected number of any given resource is the number of that resource which will most probably be
available to you.

The last table, Table 7 (page 57), lists certain necessary input data for each activity and its use is explained below.

**Procedure**

First, you should make the necessary computations to determine the critical path, the expected total cost and the expected project duration using standard PERT techniques. If the expected project duration exceeds 35 days, you may wish to accelerate the schedule for certain activities on the critical path.

Having decided which, if any, activities should be accelerated in accordance with your cost and resource constraints, you are ready to prepare the data cards for the computer program.

There are a total of 72 data input cards that are required to play the game. However, 35 of these cards remain unchanged for the duration of the project.

The first data card will contain a string of 22 integer numbers which will be given to you. For the first week's play, set all 22 numbers equal to zero. After the first week, the desired string of numbers will be found on the last line of the computer print-out. These numbers are to be transferred exactly as shown to the first data card for the next week's play. These numbers are required in order for the computer to calculate the projected completion dates for each activity.

The second data card includes 14 values, only three of which are normally changed from one week to the next. The first four numbers on the data card, in the order in which they are to appear are as follows:
1) X: the number of days in the week during which work is to be performed on the project. \((0 \leq X \leq 6, \text{ normally } X = 5 \text{ or } 6)\)

2) N: number of activities in the network (including the four dummy activities), = 35

3) FUNDS: the money available for your use during the week. Initially, for the first week, $1500.00 are available. The computer will subtract the fixed daily operating costs and the resource costs from this figure each day. If the FUNDS are exhausted before the end of the week, no work will be done on those days for which insufficient money was available. For the second and following weeks until the project is completed, the computer will print the amount of FUNDS AVAILABLE for the week following the reporting period. Any excess money from one week is to be added to the FUNDS AVAILABLE for the next week.

4) WE: the number of the week during which the work is to be done. For the first week, WE is set equal to one, etc.

The last ten values are of no interest to the player and are given in the order they must appear:

1500.00, 200.00, 1, 10, 1, 3, 5, 7, 1, 3

Remember, for each week of play, only the values of X, FUNDS, and WE are changed unless otherwise directed by the game monitor.

The last 70 data cards are composed of two data cards for each activity. The first of the two cards for each activity carries only one number. That number is the value of the variable DIRECTION found in Table 7 (page 57). Thus the first data card for activity 1 will only have the number eight, the first data card for activity 2 will only have the number two, and so on. These cards will be reused as
the first data card for each activity every week of play.

The second data card for each activity consists of from 6 to 12 numbers, depending on the resources required for each. The first three numbers on each of the second data cards are the values of I, J, and K as presented in Table 7 (page 57).

The fourth number on the second data card for each activity is a value for the variable CHECK. CHECK may be 0, 1, or 2. No other values for CHECK are meaningful to the computer. CHECK is set equal to two if the activity has been completed (all activity data cards are resubmitted for every week's play). CHECK is set equal to one any time it is desired to expedite an activity. At all other times, CHECK is set equal to zero. For the first week, then, no activity would have CHECK set equal to two since none have been completed. It may be that you want to expedite some activity scheduled to occur during the first week. This activity(s) would have its value of CHECK set equal to one. All other activities will have a CHECK equal to zero for the first week. For succeeding weeks, CHECK is given a value as required.

The fifth number on the second data card for each activity is the DAYSWORKED(I) variable, where I is the number of the activity. For the first week, all DAYSWORKED(I) are zero. For succeeding weeks, the computer prints the value for each in the Status Report. DAYSWORKED(I), as can be guessed, is the number of days of work that have actually been performed on activity I.

The sixth number on the second data card for each activity is the DAYSTOWORK(I) variable. Obviously, this is the number of days which remain to be worked on activity I. For the first week, all
DAYSTOWORK(I) are set equal to zero. For succeeding weeks, the computer prints the value for each in the Status Report.

These first six values appear in the order indicated on the second data card for every activity. They are the only values included on the second data cards for the four dummy activities 96, 97, 98, and 99.

The next values to be included on the second data card for the remaining 31 activities are the values of the resources required (from Table 5 or Table 6 depending on whether or not you wish to expedite the activity), the value of the appropriate BETA, and two given values. Thus, for activity 1, after the first six values you would have the number six (for the six laborers required), the number one (the value of BETA for this activity), and lastly, the two numbers, one and three (these last two numbers are to be included on every second data card for the 31 activities unless otherwise directed by the game monitor).

A chart to help you prepare and arrange the data on the cards is presented in Figure 13. This chart displays the presentation for the first two data cards and the 35 second data cards for each activity. The first data card for each activity, as you will recall, contains only the value of the variable DIRECTION. The values presented on the chart are those you would use for the first week if you were going to work a five-day week and were expediting activity 4 but no other activities. The resources for activity 4 came from Table 6 as did the value of BETA for this activity. Notice also, CHECK equals one for activity 4.
<table>
<thead>
<tr>
<th>Activity</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>CHECK</th>
<th>DAYS WORKED</th>
<th>DAYSTO WORK</th>
<th>Carpenter</th>
<th>Electrician</th>
<th>Mason</th>
<th>Plumber</th>
<th>Men</th>
<th>Painter</th>
<th>Grad. Tabor</th>
<th>Labor</th>
<th>BETA</th>
<th>GIVEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 13. Game Data Chart
The first card in your data card deck is card A. The second card is card B. The third card is the first data card for activity 1 and has the number eight on it. The fourth card is the second data card for activity 1 and it has the numbers 1, 1, 0, 0, 0, 0, 6, 1, 1, 3 in the order as shown. The other data cards follow in the sequence shown in Table 11. Several blank copies of Table 11 are included in this brochure for your convenience.

It will take a few minutes to prepare the data cards for the first week's play. For the following weeks, except for those activities which are expedited, most of the data remains the same with just a few changes and the original data cards can be duplicated except for the appropriate changes.

If you have any questions, see the game monitor.
CHAPTER IV

CONCLUSIONS AND RECOMMENDATIONS

A "business game" was defined in Chapter I as a "continued situation which imbeds players in a simulated business environment, where they must make management-type decisions from time to time. . . ." The PERT simulation game described in this thesis conforms to that definition inasmuch as the player is required to make decisions concerning the conduct of the project at the end of each simulated week of work. Given specified amounts of resources in the form of cash and different types of labor, the player must decide whether or not he should attempt to expedite certain project activities and/or work overtime in order to maintain or improve the project's time schedule. The player's objective is to maximize profit (with respect to the game project) in the face of uncertainties, such as the occurrence of weather and the actual duration of each activity which may or may not delay the optimum schedule. This game, then, can be said to be a competitive game in the sense that the player attempts to achieve the greatest possible profit by virtue of his decisions. Another player, with the same information, might realize more or less profit with different decisions.

Since each play of this game will most probably result in different circumstances within which a player's decisions will be made, there is "school solution" against which a student may measure the effectiveness of his play. The solution or amount of profit realized
by one player will not necessarily be the same as that of another. This is due to the fact that the determination of the status variables by the computer will probably be different for each player for each simulated period of time and the impact of each player's decisions on the project's profit will differ according to those decisions. Of course, if the status variables are the same for each player for each simulated time period throughout the play of the game, a comparison of the profits achieved by several players would be meaningful. In this case, however, the value of the simulation is compromised. Projects in the real world are unique undertakings by definition. The purpose of this game is to familiarize students with the uncertainties experienced in the real world. A small profit realized by one student as opposed to the large profit achieved by another does not necessarily reflect unfavorably on the ability or decisions of the first. He may have experienced an abnormally high number of weather delays over which he had no control. Conversely, he may have made some decisions which cost him more than he was likely to gain. In this latter event, the game monitor will be able to show him the errors in his decisions. For example, the player might elect to schedule overtime in order to ensure the earliest possible completion of an activity of the critical path. Since scheduled overtime work will be applied to all activities whose preconditions are fulfilled, the costs incurred as a result of the overtime may not justify the earliest possible completion of a critical activity. Another error would occur if the player were to allocate maximum resources to certain activities without regard for delays caused elsewhere in the project due to resource shortages. In this way
the player might concentrate his resources on the critical path activities such that he could cause another path to become critical. The effect of these errors in judgment would be to delay the completion of the entire project and to increase the total cost.

Since it is not desirable to predetermine data so that results can be compared and since profit is not solely the result of the player's decisions, it would be advantageous for a student to play the game more than one time. Several plays of the game would increase the likelihood that he will be exposed to the greatest variety of game circumstances. One play of the game might result in uniformly fortuitous circumstances wherein everything seems "to go right" and the profit is large. A second play may be the reverse of the first wherein everything seems "to go wrong" and the resultant profit is very small. A number of plays of this game would expose the student to the uncertainties of the real world and would serve to alert him to the fact that, in the real world, an optimal solution today will not necessarily be the optimal solution tomorrow. If this is accomplished, the purpose of this game will have been served and the objective of this thesis will have been met.

The PERT simulation game described in this thesis is the first computerized game of its type as far as is known to the author. This simple first game substantiates Archibald and Villoria's contention that such a game is well within the limits of current computer technology. However, several future modifications in the game's existing structure could serve to enhance its instructional and operational value.
It would be advantageous to reduce the amount of input data. Constant data associated with each activity, such as the values of the U variables, might be permanently built into the program using an array of integer values from which each Activity would select as required. It might be desirable to permanently fix the mean and variance of the funding variable and to fix the probability of weather occurrence. In any event, any program innovation which would result in the reduction of the input data would be beneficial.

As presently constructed, the program's system time is computed in days. Only integer time units are possible. This feature of the program is necessary in order that the subroutine which projects the completion date for each activity can properly operate. It is desirable that these completion dates be presented in the Status Report as occurring on a particular day. However, if the present subroutine which makes these calculations (page 64) were to be modified or replaced with another algorithm, it would be possible to use a system time based on hours instead of days. This change would require that the activity time estimates be given in hours with the result that the corresponding BETA distributions from which actual activity durations are calculated could be more accurately and realistically defined. In addition to this, a more realistic overtime function could be incorporated into the game situation if the system time was in terms of hours. The player could designate the number of hours to be worked on each activity for each day of the reporting period. It may be that the present system timing could be retained and that a change in the cost computations in each Activity from daily wage rates to hourly wage rates would be successful.
Here again, though, adjustments to the program would be required in order that the algorithm which projects completion dates would have integer values for the number of days that remain to be worked on each activity.

Further study of the principles of SIMULA operations could lead to an elimination of the four dummy activities. If Elements are declared program variables it would be possible to include a reference to a Process in more than one Set. Thus, a reference to the Process for activity 9 (page 58) could be included in the two Sets in which the reference is required. This would eliminate the need to include activity 96 in the computer program. There would have to be 31 declared Elements and some program modifications would be required.

Another aspect of the program which might be investigated concerns its length. Owing to the similarity of the Activities, it would seem that further study of this game could lead to an elimination and/or combination of some of the common computational routines.

Even if the above improvements are accomplished, the full potential of this type of game would not have been achieved. In addition to the Environment Program, a Network Planning Program (page ) could be added. Essentially, this would involve the combination of a standard PERT computer program with the game program described herein. There are at least 60 different computer programs that process critical path information. These programs offer a wide variety of output displays including bar graphs. One program, named the "UNIVAC 1107 PERT Cost
System," would be an ideal complementary information processing program for the SIMULA-based game. Both programs are written for use on either the UNIVAC 1107 or UNIVAC 1108 computers. With this suggested tandem arrangement the game information, evaluated in the SIMULA program (the Environment Program), could then be processed in a number of ways for various output displays by the PERT program (the Network Planning Program). The simple subroutine now contained in the SIMULA program which computes the projected completion dates could be eliminated since the PERT program would perform this function. Before this arrangement could be achieved, however, a suitable PERT program, such as the one mentioned above, would have to be purchased and the outputs of the SIMULA program would have to be adjusted to conform with the required PERT program inputs.

Finally, the game described herein should be compatible with the on-line network once the program has been transcribed to a tape. One difficulty here, however, is the large amount of input data presently needed to satisfy the program requirements. The amount of input data will probably have to be reduced before an on-line capability will become practically feasible.

Students may use this PERT simulation game in their study of management systems. The network model should help to familiarize the student with the PERT/CPM techniques and uncertainties associated with them.

It is evident from the literature that the influence of critical path methods is expanding in both industry and government. It is a subject of such significance as to be included as a formal course of study in many universities and government educational institutions. The PERT game developed in this study is an attempt to improve and enlarge the educational effectiveness of the formal study of PERT/CPM methodology. In its present form, it is at most a beginning.
<table>
<thead>
<tr>
<th>Reserve Word</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) ACTIVATE(X)</td>
<td>A statement used for scheduling which creates an Event notice for the Process references by element (X) and inserts this notice into the SQS.</td>
</tr>
<tr>
<td>2) ACTIVITY</td>
<td>A class of similar entities called Processes. Similar to a block in the ALGOL sense and consisting of the operations to be performed on each Process in the class.</td>
</tr>
<tr>
<td>3) CURRENT</td>
<td>Refers to the element in the first Event notice of the SQS.</td>
</tr>
<tr>
<td>4) EMPTY(S)</td>
<td>A Boolean expression which is true if the Set S is empty.</td>
</tr>
<tr>
<td>5) HISTD(A,W)</td>
<td>A random number generator whose value is an integer in the range of the one-dimensional array A. U is a specified integer variable which is replaced by the drawn value according to a set algorithm.</td>
</tr>
<tr>
<td>6) HOLD(T)</td>
<td>A timing clause used to reschedule the current Event T time units; hence, T must be non-negative.</td>
</tr>
<tr>
<td>7) INCLUDE(X,S)</td>
<td>Is used to file the process referenced by element (X) in the Set S.</td>
</tr>
<tr>
<td>8) NEW A</td>
<td>The expression for generating a Process.</td>
</tr>
<tr>
<td>9) NORMAL(a,b,U)</td>
<td>A random number generator whose value is a real number drawn from a Normal distribution with mean a and standard deviation b. U is a specified integer which is replaced by the drawn value.</td>
</tr>
<tr>
<td>10) RADINT(a,b,U)</td>
<td>A random number generator whose value is one of the integers a, a + 1, ..., b - 1, b with equal probability. U is a specified integer which is replaced by the drawn value.</td>
</tr>
<tr>
<td>Reserve Word</td>
<td>Definition</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>11) REMOVE(X)</td>
<td>A statement which causes element (X) to be removed from its Set.</td>
</tr>
<tr>
<td>12) SET</td>
<td>An ordered collection of elements.</td>
</tr>
<tr>
<td>13) SIMULA</td>
<td>The code word which introduces the SIMULA block and in which block SIMULA concepts are available.</td>
</tr>
<tr>
<td>14) TERMINATE(X)</td>
<td>A statement which deletes the Event notice and destroys the re-activation point for the process references by element (X).</td>
</tr>
</tbody>
</table>
APPENDIX B
END
INCLUDE (CURRENT, ROLE(J))
IF DAYS WORKED(I) EQ 0 AND CHECK EQ 0 THEN
BEGIN
IF BETA EQ 1 THEN DAYS WORKED(I) = HISTORY(B1, UT) ELSE
IF BETA EQ 2 THEN DAYS WORKED(I) = HISTORY(B2, UT) ELSE
IF BETA EQ 3 THEN DAYS WORKED(I) = HISTORY(B3, UT) ELSE
IF BETA EQ 5 THEN DAYS WORKED(I) = HISTORY(B5, UT) ELSE
DAYS WORKED(I) = HISTORY(B0, UT)
END
IF CHECK EQ 1 THEN
BEGIN
IF BETA EQ 1 THEN DAYS WORKED(I) = HISTORY(B1, UT) ELSE
IF BETA EQ 3 THEN DAYS WORKED(I) = HISTORY(B3, UT) ELSE
IF BETA EQ 5 THEN DAYS WORKED(I) = HISTORY(B5, UT) ELSE
DAYS WORKED(I) = HISTORY(B7, UT)
END
FOR W = (1, X) DO
BEGIN
IF DAYS WORKED(I) LEQ (X - I - W) THEN BASE(I) = BASE(I) + DAYS WORKED(I)
ELSE BASE(I) = BASE(I) + (DAYS WORKED(I) + (6 - X))
IF BASE(I) GT BASE(J) THEN
A(J) = BASE(I)
IF EMPTY (ROLE(K)) THEN
BEGIN
IF CARPENTER(W) GEQ CARPS AND W X(W) LSS 10 AND FUNDS GEQ COST THEN
BEGIN
CARPENTER(W) = CARPENTER(W) - CARPS
IF W LEQ 5 THEN
COST = COST + ((CARPS * RATE(3)) + (LABS * RATE(4))) ELSE
COST = COST + (3 * ((CARPS * RATE(3)) + (LABS * RATE(4)))) / 2
DAYS WORKED(I) = DAYS WORKED(I) + 1
DAYS WORKED(I) = DAYS WORKED(I) = 1
IF DAYS WORKED(I) EQ 0 THEN
BEGIN
REMOVE (CURRENT)
BASE(I) = 75
TERMINATE (CURRENT)
END
GO TO L1
END
L1: HOLD(I)
A(K) = A(K) + 1
END
ACTIVITY BAKERS BEGIN
INTEGER T, MAS, LABS, CHECK, BETA, UT, U8
READ CARDS, J, K, CHECK, DAYSTOWORK(I), DAYSTOWORK(I), MAS, LABS
INCLUDE (CURRENT, ROLES)
IF CHECK EQL 2 THEN
BEGIN
BASE(I) = 75
TERMINATE (CURRENT)
END
IF DAYSTOWORK(I) EQ 0 AND CHECK EQL 0 THEN
BEGIN
IF BETA EQL 2 THEN DAYSTOWORK(I) = HISTD(B2, U7) ELSE
IF BETA EQL 3 THEN DAYSTOWORK(I) = HISTD(B3, U8) ELSE
DAYSTOWORK(I) = HISTD(B5, U8)
END
IF CHECK EQL 1 THEN
BEGIN
IF BETA EQL 1 THEN DAYSTOWORK(I) = HISTD(B1, U8) ELSE
IF BETA EQL 3 THEN DAYSTOWORK(I) = HISTD(B3, U8) ELSE
IF BETA EQL 5 THEN DAYSTOWORK(I) = HISTD(B5, U8) ELSE
DAYSTOWORK(I) = HISTD(B7, U8)
END
FOR T = (1, 1, X) DO
BEGIN
IF DAYSTOWORK(I) LEQ (X+1-T) THEN BASE(I) = A(K) + DAYSTOWORK(I)
ELSE BASE(I) = A(K) + DAYSTOWORK(I) + (6-X)
IF BASE(I) GTR A(J) THEN A(J) = BASE(I)
IF EMPTY(ROLE(K)) THEN
BEGIN
IF MASON(T) GEQ MAS AND WX(T) LSS 10 AND FUNDS GEQ COST THEN
BEGIN
MASON(T) = MASON(T) = MAS
IF T LEQ 5 THEN
COST = COST + MAS * RATE(1) + LABS * RATE(4)) ELSE
COST = COST + 3 * (MAS * RATE(1) + LABS * RATE(8)) / 2
DAYSTOWORK(t) = DAYSTOWORK(t) + 1
DAYSTOWORK(T) = DAYSTOWORK(T) + 1
IF DAYSTOWORK(T) EQ 0 THEN
BEGIN
REMOVE (CURRENT)
BASE(T) = 75
TERMINATE (CURRENT)
END
GO TO L2
END
END
ACTIVITY CHARLIE BEGIN
INTEGER R,ELECS,LABS,CHECK,BETA,UT,UB
READ: CARDS, (CARDS, K, CHECK, DAYSTOWORK(T), ELECS, LABS, BETA, UT, UB)
IF CHECK LSS 2 THEN
BEGIN
BASE(T) = 75
TERMINATE (CURRENT)
END
INCLUDE (CURRENT, ROLE(K))
IF DAYSTOWORK(T) LSS 0 AND CHECK LSS 0 THEN
BEGIN
IF BETA LSS 2 THEN DAYSTOWORK(T) = HISTD(B2, U7) ELSE
DAYSTOWORK(T) = HISTD(B1, U7)
IF CHECK EQL 2 THEN DAYSTOWORK(I) = 1
FOR R = (1:10) DO
BEGIN
IF DAYSTOWORK(I) LEQ (K+1-R) THEN BASE(I) = A(K) + DAYSTOWORK(I)
ELSE BASE(I) = A(K) + DAYSTOWORK(I) + (6-K)
IF BASE(I) GTR A(J) THEN A(J) = BASE(I)
IF EMPTY (ROLE(K)) THEN
BEGIN
IF ELECTRICIAN(R) GEG ELECS AND WK(R) LSS 10 AND FUNDS GEG COST THEN
BEGIN
ELECTRICIAN(R) = ELECTRICIAN(R) - ELECS
IF R LEQ 5 THEN
COST = COST + ELECS*RATE(2) + LABS*RATE(4) ELSE
COST = COST + ($ELECS*RATE(2) + LABS*RATE(4)) / 2
DAYSTWORKED(I) = DAYSTWORKED(I) + 1
DAYSTOWORK(I) = DAYSTOWORK(I) + 1
IF DAYSTOWORK(I) EQL 0 THEN
BEGIN
REMOVE (CURRENT)
BASE(I) = 75
TERMINATE (CURRENT)
END
END
GO TO L3
END
L3: HOLD (1)
A(K) = A(K) + 1
END
END
ACTIVITY DELTAS BEGIN
INTEGER T, PLUMS, LABS, CHECK, BETA, OUTPUT
READ (CARDS, I, J, K, CHECK, DAYSTWORKED(I), DAYSTOWORK(I), PLUMS, LABS, OUTPUT)
IF CHECK EQL 2 THEN
BEGIN
...
BASE(i) = 75
TERMINATE (CURRENT)
END
INCLUDE (CURRENT, ROLE(1))
IF DAYSWORKED(i) EQL 0 AND CHECK EQL 0 THEN
BEGIN
IF BETA EQL 2 THEN DAYSWORK(i) = HISTO(BEUTY) ELSE
DAYSWORK(i) = HISTO(BEUTY)
END
IF CHECK EQL 1 THEN DAYSWORK(i) = 1
FOR T = (1..X) DO
BEGIN
IF DAYSWORK(i) LEQ (X+1) THEN BASE(i) = A(K) + DAYSWORK(i)
ELSE BASE(i) = A(K) + DAYSWORK(i) + (6-X)
IF BASE(i) GT A(J) THEN A(J) = BASE(i)
IF EMPTY (ROLE(K)) THEN
BEGIN
IF PLUMBER(Y) GEC PLUMS AND WAGE LESS 10 AND FUNDS GEC COST THEN
BEGIN
PLUMBER(Y) = PLUMBER(Y) - PLUMS
IF T LEQ 5 THEN
COST = COST + PLUMSERATE(i) + LABSRATE(4) ELSE
COST = COST + (5*PLUMSERATE(i) + LABSRATE(4))/2
DAYSWORK(i) = DAYSWORK(i) + 1
DAYSWORK(i) = DAYSWORK(i) + 1
IF DAYSWORK(i) EQL 0 THEN
BEGIN
REMOVE (CURRENT)
BASE(i) = 75
TERMINATE (CURRENT)
END
END
END
GO TO LA
END
L4: HOLD (L)
A(K) = A(K) + 1
END
BEGIN
INTEGER YPCARPS, LABS, CHECK, BETADUT, UB
READ (CARDS, I, J, K, CHECK, DAYS WORKED(I), DAYSTOWORK(I), CARPS, LABS, BETADUT, UB)
IF CHECK EQL 2 THEN
BEGIN
BASE(I) = 75
TERMINATE (CURRENT)
END IF DAYSTOWORK(I) EQL 0 AND CHECK EQL 0 THEN
BEGIN
IF BETA EQL 1 THEN DAYSTOWORK(I) = HISTD(B1, UB) ELSE
IF BETA EQL 3 THEN DAYSTOWORK(I) = HISTD(B3, UB) ELSE
IF BETA EQL 5 THEN DAYSTOWORK(I) = HISTD(B5, UB)
END
IF CHECK EQL 1 THEN
BEGIN
IF BETA EQL 1 THEN DAYSTOWORK(I) = HISTD(B1, UB) ELSE
IF BETA EQL 3 THEN DAYSTOWORK(I) = HISTD(B3, UB) ELSE
IF BETA EQL 5 THEN DAYSTOWORK(I) = HISTD(B5, UB)
END
FOR Y = (I, J, X) DO
BEGIN
IF DAYSTOWORK(I) LEQ (X*1-Y) THEN BASE(I) = A(K) + DAYSTOWORK(I)
ELSE BASE(I) = A(K) + DAYSTOWORK(I) + (6-X)
IF BASE(I) GTR A(J) THEN A(J) = BASE(I)
IF EMPTY (ROLE(K)) THEN
BEGIN
IF CARPENTER(Y) GEQ CARPS AND FUNDS GEQ COST THEN
BEGIN
CARPENTER(Y) = CARPENTER(Y) + CARPS
IF Y LEQ 5 THEN
COST = COST + CARPS RATE(3) + LABS RATE(4) ELSE


COST := COST + (5 * CARPS * RATE(3) + LABS * RATE(4))/2

DAYSWORKED(I) = DAYSWORKED(I) + 1

DAYSSTOWORK(I) = DAYSSTOWORK(I) + 1

IF DAYSSTOWORK(I) EQL 0 THEN

BEGIN

REMOVE (CURRENT)

BASE(I) = 75

TERMINATE (CURRENT)

END

GO TO L5

END

L5: HOLD (I)

A(K) = A(K) + 1

END

ACTIVITY FO$ BEGIN

INTEGER, PPLUMS, LABS, CHECK, BETA, UT, U8

READ (CARDS, PPLUMS,实验室, CHECK, BETA, UT, U8)

IF CHECK EQL 2 THEN

BEGIN

BASE(I) = 75

TERMINATE (CURRENT)

END

INCLUDE (CURRENT, ROLE(J))

IF DAYSWORKED(I) EQL 0 AND CHECK EQL 0 THEN

BEGIN

IF BETA EQL 1 THEN DAYSSTOWORK(I) = HISTD(B1, UT) ELSE

IF BETA EQL 2 THEN DAYSSTOWORK(I) = HISTD(B2, UT) ELSE

IF BETA EQL 4 THEN DAYSSTOWORK(I) = HISTD(B4, UT) ELSE

DAYSSTOWORK(I) = HISTD(B6, UT)

END

IF CHECK EQL 1 THEN

BEGIN

IF BETA EQL 1 THEN DAYSSTOWORK(I) = HISTD(B1, U8) ELSE

IF BETA EQL 3 THEN DAYSSTOWORK(I) = HISTD(B3, U8) ELSE
IF BETA EQL 5 THEN DAWTSTW0RK(I) = HISTD(B5pU8) ELSE
DAYSrOWKWI) = HISTD(B7pU8)
END
FOR R = (1i2iX) DO
BEGIN
IF DAYSr0WKWI) EQL (Xk1-R) THEN BASE(I) = A(K) + DAYSr0RKWI) ELSE BASE(I) = A(K) + (6-X)
IF BASE(I) GTR A(J) THEN A(J) = BASE(I)
IF EMPTY (ROLE(K)) THEN
BEGIN
IF PLUMBER(R) GEQ PLUMS AND FUNDS GEQ COST THEN
BEGIN
PLUMBER(R) = PLUMBER(R) = PLUMS
IF R LEQ 5 THEN
COST = COST + PLUMS*RATE(1) + LABS*RATE(4) ELSE
COST = COST + (3*(PLUMS*RATE(1) + LABS*RATE(4)))/2
DAYSr0RKED(W) = DAYSr0RKED(I) + 1
DAYSr0WKWI) = DARYSTW0RKWI) = 1
IF DAYSr0RKWI) EQL 0 THEN
BEGIN
REMOVE (CURRENT)
BASE(I) = 75
TERMINATE (CURRENT)
END
END
GO TO L6
END
L6 : HOLD (I)
A(K) = A(K) + 1
END
END
ACTIVITY GEORGE$ BEGIN
INTEGER WAELCS$LABS$CHECK$BETA$U TpU8
READ (CARDS$P$J$K$CHECK$DAYSr0RKED(W)$DAYSr0RKWI) , ELECS , LABSP, BETA$UTpU8)
IF CHECK FQL 2 THEN
BEGIN
BASE(I) = 75
TERMINATE (CURRENT)
END

INCLUDE (CURRENT, ROLE(J))

IF DAYSWORKED(I) EQL 0 AND CHECK EQL 0 THEN
BEGIN
IF BETA EQL 2 THEN DAYSSTWORK(I) = HISTD(B2 , U7) ELSE
IF BETA EQL 4 THEN DAYSSTWORK(I) = HISTD(B4 , U7) ELSE
IF BETA EQL 6 THEN DAYSSTWORK(I) = HISTD(B6 , U7) ELSE
DAYSSTWORK(I) = HISTD(B10 , U7)
END

IF CHECK EQL 1 THEN
BEGIN
IF BETA EQL 3 THEN DAYSSTWORK(I) = 1 ELSE
IF BETA EQL 5 THEN DAYSSTWORK(I) = HISTD(B3 , U8) ELSE
IF BETA EQL 7 THEN DAYSSTWORK(I) = HISTD(B5 , U8) ELSE
DAYSSTWORK(I) = HISTD(B7 , U8)
END

FOR W = (1, 1, X) DO
BEGIN
IF DAYSSTWORK(I) LEQ (X+1-W) THEN BASE(I) = A(K) + DAYSSTWORK(I) ELSE BASE(I) = A(K) + DAYSSTWORK(I) + (6-X)
IF BASE(I) GTR A(J) THEN A(J) = BASE(I)
IF EMPTY(ROLE(K)) THEN
BEGIN
IF ELECTRICIAN(W) GEG ELECS AND FUNDS GEG COST THEN
BEGIN
ELECTRICIAN(K) = ELECTRICIAN(W) = ELECS
IF W LEQ 5 THEN
COST = COST + ELECS*RATE(2) + LABS*RATE(4) ELSE
COST = COST + (3*(ELECS*RATE(2) + LABS*RATE(4)))/2
DAYSSTWORKED(I) = DAYSSTWORKED(I) + 1
DAYSSTWORK(I) = DAYSSTWORK(I) + 1
IF DAYSSTWORK(I) EQL 0 THEN
BEGIN
REMOVL (CURRENT)
BASE(I) = 75
END

END

END
TERMINATE (CURRENT)
END
END
GO TO LT
END
LT: HOLD (I)
A(K) = A(K) + 1
END
END
ACTIVITY HOTELS BEGIN
INTEGER Y, LABS, CHECK, BETA, UP
READ (CARDS, I, J, K, CHECK, DAYSWORKED(I), DAYSTOWORK(I), LABS, BETA, UP)
IF CHECK EQL 2 THEN
BEGIN
BASE(I) = 75
TERMINATE (CURRENT)
END
END
INCLUDE (CURRENT, ROLE(J))
IF DAYSWORKED(I) EQL 0 THEN DAYSTOWORK(I) = HISTO(Bl, U7)
FOR Y = (1, 1, X) DO
BEGIN
IF DAYSTOWORK(I) LEQ (X+1-Y) THEN BASE(I) = A(K) + DAYSTOWORK(I)
ELSE BASE(I) = A(K) + DAYSTOWORK(I) + (6-K)
IF BASE(I) GTR A(J) THEN A(J) = BASE(I)
IF EMPTY (ROLE(K)) THEN
BEGIN
IF WX(Y) LSS 10 AND FUNDS GEQ COST THEN
BEGIN
IF Y LEQ 5 THEN
COST = COST + LABS*RATE(I) ELSE
COST = COST + (3*LABS*RATE(I))/2
DAYSTOWORK(I) = DAYSWORKED(I) + 1
DAYSTOWORK(I) = DAYSTOWORK(I) + 1
IF DAYSTOWORK(I) EQL 0 THEN
BEGIN
REMOVE (CURRENT)
BASE(I) = 75
END
END
END
END
END
END
END
TERMINATE (CURRENT)
END
GO TO L8
END

LB8: HOLD (1)
A(K) = A(K) + 1
END

ACTIVITY INDIA BEGIN
INTEGER T, DRYSLABS, CHECK, BETA, U7, U8
READ (CARDS, J, K, CHECK, DAVSWORKED(I), DAYSTOWORK(I), DRYSLABS, BETA, U7, U8)
IF CHECK EQL 2 THEN
BEGIN
BASE(I) = 75
TERMINATE (CURRENT)
END

INCLUDE (CURRENT, ROLE(J))
IF DAVSWORKED(I) EQL 0 AND CHECK EQL 0 THEN
BEGIN
IF BETA EQL 5 THEN DAYSTOWORK(I) = HISTD(B5, U7)
ELSE
DAYSTOWORK(I) = HISTD(B9, U8)
END
IF CHECK EQL 1 THEN
BEGIN
IF BETA EQL 1 THEN DAYSTOWORK(I) = 1 ELSE
IF BETA EQL 3 THEN DAYSTOWORK(I) = HISTD(B3, U8) ELSE
IF BETA EQL 5 THEN DAYSTOWORK(I) = HISTD(B7, U8)
END
FOR T = (1, 10) DO
BEGIN
IF DAYSTOWORK(I) LEQ (X * 1 - T) THEN BASE(I) = A(K) + DAYSTOWORK(I)
ELSE BASE(I) = A(K) + DAYSTOWORK(I) + (6 - X)
IF BASE(I) GTR A(J) THEN A(J) = BASE(I)
IF EMPT Y (ROLE(K)) THEN
BEGIN IF FUNDS GE Q COST THEN
BEGIN IF T LEQ 5 THEN COST = (DRY$RATE(3) + LABS$RATE(4)) ELSE COST = (3*(DRY$RATE(3) + LABS$RATE(4)))/2 $ DAY$WORKED(I) = DAY$WORKED(I) + 1 $ DAY$STOWORK(I) = DAY$STOWORK(I) - 1 $ IF DAY$STOWORK(I) EQL 0 THEN
BEGIN REMOVE (CURRENT?) $ BASE(I) = 75 $ TERMINATE (CURRENT) $ END $ GO TO L9 $ END $ L9: HOLD (I) $ A(K) = A(K) + 1 $ END $ ACTIVITY JULIET$ BEGIN $ INTEGER T, LABS$CHECK, BETAP, U $ READ (CARDS, J, K, CHECK, DAY$WORKED(I), DAY$STOWORK(I), LABS$CHECK, BETAP, U) $ IF CHECK EQL 2 THEN
BEGIN $ BASE(I) = 75 $ TERMINATE (CURRENT) $ END $ INCLUDE (CURRENT$ROLE(J)) $ IF DAY$WORKED(I) EQL 0 THEN DAY$STOWORK(I) = HISTO(BETAP, U) $ FOR Y = (I, 1, X) DO
BEGIN $ IF DAY$STOWORK(I) LEQ (X+1-Y) THEN BASE(I) = A(K) + DAY$STOWORK(I) $ ELSE BASE(I) = A(K) + DAY$STOWORK(I) + (8-X) $ IF BASE(I) GT A(J) THEN A(J) = BASE(I) $ IF EMPTY (ROLE(K)) THEN
BEGIN
IF WX(T) LSS 10 AND FUNDS GEQ COST THEN
BEGIN
IF T LEQ 5 THEN COST = COST + 32 * LABS * RATE(4) ELSE
COST = COST + (3 * (32 * LABS * RATE(4))) / 2
DAYSWORKED(I) = DAYSWORKED(I) + 1
DAYSTOWORK(I) = DAYSTOWORK(I) - 1
IF DAYSTOWORK(I) EQ 0 THEN
BEGIN
REMOVE (CURRENT)
BASE(I) = 75
TERMINATE (CURRENT)
END
END
GO TO L10
L10: HOLD (I)
A(K) = A(K) + 1
END
ACTIVITY KILOs BEGIN
INTEGER RPM ASpPLUMS DRYS LABS CHECK BETA U7
READ (CARDS Ip, Jp, Kp, CHECK, DAYSWORKED(I), DAYSTOWORK(I), MAS, PLUMS,
DRYS, LABS, BETA, U7)
IF CHECK EQ 2 THEN
BEGIN
BASE(I) = 75
TERMINATE (CURRENT)
END
INCLUDE (CURRENT, ROLE(J))
IF DAYSWORKED(I) EQ 0 THEN DAYSTOWORK(I) = HISTO(B3, U7)
FOR R = (1..X) DO
BEGIN
IF DAYSTOWORK(I) LEQ (I = R) THEN BASE(I) = A(K) + DAYSTOWORK(I)
ELSE BASE(I) = A(K) + DAYSTOWORK(I) - (6-X)
IF BASE(I) GT A(J) THEN A(J) = BASE(I)
IF EMPTY (ROLE(K)) THEN
BEGIN
IF MASON(R) GEQ MAS AND PLUMBER(R) GEQ PLUMS AND FUNDS GEQ COST THEN
BEGIN
MASON(R) = MASON(R) - MAS
PLUMBER(R) = PLUMBER(R) - PLUMS
IF R LEQ 5 THEN
COST = COST + MAS * RATE(1) + PLUMS * RATE(1) + DRY'S * RATE(3)
+ LABS * RATE(4) ELSE COST = (COST + (3 * MAS * RATE(1) + PLUMS * RATE(1))
+ DRY'S * RATE(3) + LABS * RATE(4)) / 2
DAYS WORKED(I) = DAYS WORKED(I) + 1
DAYS TO WORK(I) = DAYS TO WORK(I) + 1
IF DAYS TO WORK(I) EQL 0 THEN
BEGIN
REMOVED CURRENT
BASE(I) = 75
TERMINATE CURRENT
END
END
GO TO L11
L11 HOLD(I)
A(K) = A(K) + 1
END
END
ACTIVITY LIMAS BEGIN
INTEGER T, LABS, CHECK, BETA, UT, UB
READ (CARDS, I, J, K, CHECK, DAYS WORKED(I), DAYS TO WORK(I), LABS, BETA,
UT, UB)
IF CHECK EQL 2 THEN
BEGIN
BASE(I) = 75
TERMINATE CURRENT
END
INCLUDE CURRENT ROLE(J)
IF DAYS WORKED(I) EQL 0 AND CHECK EQL 0 THEN
BEGIN
IF BETA EQL 2 THEN DAYS TO WORK(I) = HISTD(B2, UT) ELSE
IF BETA EQL 3 THEN DAYSTOWORK(1) = HISTD(3, U7) ELSE
DAYSTOWORK(1) = HISTD(3, U7)
END

IF CHECK EQL 1 THEN
BEGIN
IF BETA EQL 1 THEN DAYSTOWORK(1) = 1 ELSE
IF BETA EQL 3 THEN DAYSTOWORK(1) = HISTD(3, U6) ELSE
DAYSTOWORK(1) = HISTD(3, U6)
END

FOR T = (1, 1, X) DO
BEGIN
IF DAYSTOWORK(I) LEQ (X - 1 - T) THEN BASE(I) = A(I, 3) + DAYSTOWORK(I) ELSE BASE(I) = A(K) + DAYSTOWORK(I) + (6 - X)
IF BASE(I) GTR A(I, T) THEN A(I) = BASE(I)
IF EMPTY (ROLE(K)) THEN
BEGIN
IF FUNDS GEQ COST THEN
BEGIN
IF T LEQ 5 THEN
COST = COST + LABS RATE(4) ELSE
COST = COST + (3 * LABS RATE(4)) / 2
DAYSTOWORKED(I) = DAYSTOWORKED(I) + 1
DAYSTOWORK(I) = DAYSTOWORK(I) - 1
IF DAYSTOWORK(I) EQL 0 THEN
BEGIN
REMOVE (CURRENT)
BASE(I) = 75
TERMINATE (CURRENT)
END
END
END
GO TO L12
END
L12: HOLD (1)
A(K) = A(K) + 1
END
END
ACTIVITY MIKES BEGIN
INTEGER Y, CARPS, PAINS, CHECK, BETAP, U7, U8
READ (CARDS, 10) K, CHECK, DAYSWORKED(I), DAYSWORKED(I), CARPS, PAINS, BETAP, U7, U8

IF CHECK EQU 2 THEN
BEGIN
BASE(I) = 75
TERMINATE (CURRENT)
END

INCLUDE (CURRENT, ROLE(J))

IF DAYSWORKED(I) EQU 0 AND CHECK EQU 0 THEN
BEGIN
IF BETA EQU 4 THEN DAYSWORKED(I) = HIST0(B6, U7) ELSE DAYSWORKED(I) = HIST0(B6, U7)
END

IF CHECK EQU 1 THEN
BEGIN
IF BETA EQU 1 THEN DAYSWORKED(I) = 1 ELSE IF BETA EQU 3 THEN DAYSWORKED(I) = HISTD(B3, U8) ELSE IF BETA EQU 5 THEN DAYSWORKED(I) = HISTD(B5, U8)
END

FOR Y = (1, 10) DO
BEGIN
IF DAYSWORKED(I) LEQ X + 1 THEN BASE(I) = A(K) + DAYSWORKED(I)
ELSE BASE(I) = A(K) + DAYSWORKED(I) + (6 - X)
IF BASE(I) GTR A(J) THEN A(J) = BASE(I)
IF EMPTY (ROLE(K)) THEN
BEGIN
IF CARPENTER(Y) GEQ CARPS AND PAINTER(Y) GEQ PAINS AND FUNDS
BEGIN
COST = CARPENTER(Y) = CARPS
PAINTER(Y) = PAINTER(Y) = PAINS
IF Y LEQ 5 THEN
COST = COST + CARPS*RATE(3) + PAINS*RATE(3) ELSE
COST = COST + (3*(CARPS*RATE(3) + PAINS*RATE(3)))/2
DAYSWORKED(I) = DAYSWORKED(I) + 1
END
END
END

END
DAYSTOWORK(I) = DAYSTOWORK(I) + 1
IF DAYSTOWORK(I) EQ 0 THEN
BEGIN
REMOVE (CURRENT)
BASE(I) = 75
TERMINATE (CURRENT)
END
GO TO L13
END

L13
HOLD (I)
A(K) = A(K) + 1
END

ACTIVITY NOLAN'S BEGIN
INNER Y, PAINS, CHECK, BETA, UT, U8
READ (CARDS, I, K, CHECK, DAYWORKED(I), DAYSTOWORK(I), PAINS, BETA, UT, U8)
END

IF CHECK EQ 2 THEN
BEGIN
BASE(I) = 75
TERMINATE (CURRENT)
END

INCLUDE (CURRENT, ROLE(J))

IF DAYWORKED(I) EQ 0 AND CHECK EQ 0 THEN
BEGIN
IF BETA EQ 3 THEN DAYSTOWORK(I) = HISTD(B3, U7) ELSE
DAYSTOWORK(I) = HISTD(B3, U7)
END

IF CHECK EQ 1 THEN
BEGIN
IF BETA EQ 1 THEN DAYSTOWORK(I) = 1 ELSE
DAYSTOWORK(I) = 2
END

FOR Y = (I+1, X) DO
BEGIN
IF DAYSTOWORK(I) LEQ (X+1-Y) THEN BASE(I) = A(K) + DAYSTOWORK(I)
ELSE \( \text{BASE}(i) = A(k) + \text{DAYSTOWORK}(i) + (6 - x) \)

IF \( \text{BASE}(i) \text{ GTR A}(j) \) THEN \( A(j) = \text{BASE}(i) \)

IF \( \text{EMPTY (ROLE}(j)) \) THEN

BEGIN

IF \( \text{PAINTER}(y) \text{ GEQ PAINS AND FUNDS GEQ COST} \) THEN

BEGIN

\( \text{PAINTER}(y) = \text{PAINTER}(y) - \text{PAINS} \)

IF \( y \text{ LEQ 5} \) THEN

\( \text{COST} = \text{COST} + \text{PAINS*RATE(3)} \) ELSE

\( \text{COST} = \text{COST} + (3*(\text{PAINS*RATE(3)}))/2 \)

\( \text{DAYWORKED}(i) = \text{DAYWORKED}(i) + 1 \)

\( \text{DAYSTOWORK}(i) = \text{DAYSTOWORK}(i) - 1 \)

IF \( \text{DAYSTOWORK}(i) \text{ EQL 0} \) THEN

BEGIN

REMOVE \( \text{CURRENT} \)

\( \text{BASE}(i) = 75 \)

TERMINATE \( \text{CURRENT} \)

END

END

END

GO TO L14

END

L14: HOLD \( (i) \)

\( A(k) = A(k) + 1 \)

END

ACTIVITY OSCARS BEGIN

INTEGER \text{CHECK}

READ (\text{CARDS} \text{I}, \text{J}, \text{K}, \text{CHECK}, \text{DAYWORKED}(i), \text{DAYSTOWORK}(i))

IF \text{CHECK} \text{ EQL 2} \) THEN

BEGIN

\( \text{BASE}(i) = 75 \)

TERMINATE \( \text{CURRENT} \)

END

INCLUDE \( \text{CURRENT}, \text{ROLE}(j) \)

L15: IF \( \text{EMPTY (ROLE}(k)) \) THEN

BEGIN

IF \( A(k) \text{ GTR A}(j) \) THEN \( A(j) = A(k) \)
CARPENTER(V) = HISTD(CA\textsubscript{U1})
MASON(V) = HISTD(MA\textsubscript{U2})
ELECTRICIAN(V) = HISTD(EL\textsubscript{U3})
PLUMBER(V) = HISTD(PL\textsubscript{U4})
PAINTER(V) = HISTD(PA\textsubscript{U5})
WX(V) = RANDINT(STPF/PU\textsubscript{U6})

END

WRITE (FM2,PFOR V = (1,1,6) DO CARPENTER(V),FOR V = (1,1,6) DO
MASON(V),FOR V = (1,1,6) DO ELECTRICIAN(V),FOR V = (1,1,6) DO
PLUMBER(V),FOR V = (1,1,6) DO PAINTER(V))

WRITE (FM3,WE)

RATE(1) = 36$
RATE(2) = 40$
RATE(3) = 32$
RATE(4) = 20$

FOR V = (1,1,6) DO

BEGIN
READ (CARDS, DIRECTION)
IF DIRECTION EQL 1 THEN ACTIVATE NEW ALPHA ELSE
IF DIRECTION EQL 2 THEN ACTIVATE NEW BAKER ELSE
IF DIRECTION EQL 3 THEN ACTIVATE NEW CHARLIE ELSE
IF DIRECTION EQL 4 THEN ACTIVATE NEW DELTA ELSE
IF DIRECTION EQL 5 THEN ACTIVATE NEW ECHO ELSE
IF DIRECTION EQL 6 THEN ACTIVATE NEW FOX ELSE
IF DIRECTION EQL 7 THEN ACTIVATE NEW GEORGE ELSE
IF DIRECTION EQL 8 THEN ACTIVATE NEW HOTEL ELSE
IF DIRECTION EQL 9 THEN ACTIVATE NEW INDIA ELSE
IF DIRECTION EQL 10 THEN ACTIVATE NEW JULIET ELSE
IF DIRECTION EQL 11 THEN ACTIVATE NEW KILO ELSE
IF DIRECTION EQL 12 THEN ACTIVATE NEW LIMA ELSE
IF DIRECTION EQL 13 THEN ACTIVATE NEW MIKE ELSE
IF DIRECTION EQL 14 THEN ACTIVATE NEW NOLAN ELSE
ACTIVATE NEW OSCAR

END

HOLD (X+1)
NEWFUNDS = NORMAL(SEG, EN, U6)

WRITE (FM3,WE)

WRITE (FM2,PFOR V = (1,1,6) DO CARPENTER(V),FOR V = (1,1,6) DO
MASON(V),FOR V = (1,1,6) DO ELECTRICIAN(V),FOR V = (1,1,6) DO
PLUMBER(V),FOR V = (1,1,6) DO PAINTER(V))
WRITE (FM4)
E = 315 F = 95 G = 08
FOR V = (1,1,31) DO
BEGIN
IF V EQL 10 THEN
L17: BEGIN
E = E + 15 F = F + 15
IF BASE(E) EQL 75 THEN
WRITE (FM9,F,DAYSWORKED(E),DAYSTOWORK(E)) ELSE
IF BASE(E) GTR 30 THEN
WRITE (FM6,F,DAYSWORKED(E),DAYSTOWORK(E),(BASE(E) = 30)) ELSE
WRITE (FM5,F,DAYSWORKED(E),DAYSTOWORK(E),BASE(E))
GO TO L18
END
IF V EQL 17 THEN GO TO L17
IF V EQL 24 THEN GO TO L17
L18: IF V EQL 24 AND G EQL 0 THEN
BEGIN
G = 1
GO TO L17
END
IF BASE(V) EQL 75 THEN
WRITE (FM9,V,DAYSWORKED(V),DAYSTOWORK(V)) ELSE
IF BASE(V) GTR 30 THEN
WRITE (FM6,V,DAYSWORKED(V),DAYSTOWORK(V),(BASE(V) = 30)) ELSE
WRITE (FM5,V,DAYSWORKED(V),DAYSTOWORK(V),BASE(V))
END
WRITE (FM7,COST)
WRITE (FM8,(WE + 1),NEWFUNDS)
WRITE (FM10,FOR V = (1,1,22) DO A(V))
END.
LITERATURE CITED


OTHER REFERENCES


