A B S T R A C T

THE DESIGN AND USE OF A SIMULATION PACKAGE IN ALGOL-W

Nowadays, there are many problems which the system designer has to solve, which are extremely difficult, if not impossible, to solve using conventional analytical methods. Quite often, the experience of the designer has enabled him to solve the problem intuitively. However, a better approach would be preferred in solving these problems.

By using a computer to simulate a system by taking one activity at a time, obeying the rules for that activity and then carrying on with whatever comes next, it is possible to see what might happen to a system in practice. Simulation is now a recognised and widely used tool and a number of techniques have evolved whereby a system designer can set up a computer simulation.

Special purpose simulation languages were developed which provide the basic data-structures and control structure required to run a simulation system. However, many computers do not have compilers for special purpose languages and they are also difficult to learn for the average programmer. For this reason it is an advantage to use a commonly used language for a simulation process. My package is written in ALGOL-W, a variation of ALGOL-60 which includes the additional facility of data-structures.

The package includes the necessary declarations and procedures required to run a simulation including a "diary" which keeps a note of the various "events" in the system. Also included are "queues" and "facilities" which have predefined procedures acting on them and automatic collection of statistics. The package assumes the existence of "things" defined by the user, which represent objects moving through the system.

In general, the simulated system consists of a number of fixed objects (e.g. telephone box, cash desk, etc.) with a succession of transient objects that move through the system (e.g. people,cars or whatever).

When simulating any particular system it is useful to be actually able to see what is happening to the model during the simulation. For this reason I have introduced the facility which enables "snap-shots" of the system to be output. This is a picture of the simulation model, showing the position of "things" at that particular moment in simulated time.

The description of the simulation package, together with the additional facility of obtaining pictures is explained in this thesis as well as examples as to how the package is used with reference to particular systems.

Maths Library

THE DESIGN AND USE OF A SIMULATION PACKAGE IN ALGOL-W

by

Geoffrey H. Maunder

A thesis presented for the degree of Master of Science of the University of St. Andrews

1975

ACKNOWLEDGEMENTS

My sincere thanks to:

Professor A.J. Cole who provided the opportunity for this study.

Mr. D.A. Turner, my supervisor, whose help, stimulation and guidance has been invaluable.

Other members of the laboratory staff who helped with the running of my programs in some way.

CONTENTS

CHAPTER 1

I N T R O D U C T I O N

1.1 Why Simulation

Nowadays, there are many problems which the system designer has to solve, which are extremely difficult, if not impossible, to solve using conventional analytical methods. Quite often, the experiences of the designer has enabled him to solve the problem intuitively. However, a better approach would be preferred in solving these problems.

Examples such as traffic flow, where road designers are trying to decide whether streets should be one-way or two, what period of time to set traffic lights to green, whether a bus lane should be used on a particular stretch of road and so on as well as other examples such as deciding how many checkouts to build in a new supermarket are all problems which need to be solved.

Although many of these problems can be solved using various operational research formulae such as queueing theory, system behaviour prediction, job shop analysis, system effectiveness and management information, as a problem gets more complicated, so more complex calculations may have to be made or the problem may have to be over-simplified to make calculations easier.

Although computers can be used merely to evaluate these complex analytical formulae by substituting the appropriate parameters, actually using the computer to simulate a system will produce results without the need to use these formulae at all, but instead, by taking one activity at a time, obeys the rules for that activity and then carries on with whatever comes next. By this method, it is also possible to view the system at particular points during the simulation and see what is happening. By solving a formula, it is only possible to get some result rather than actually watching what happens in practice which is possible with simulation. Simulation can give the system designer the feeling, insight and opportunity to operate and manipulate a system plus the measure of confidence, while the system is still a paper concept.

Once a particular system has been set up inside the computer so that it produces a valid representation of the real world situation, then a series of simulations may be run with different parameters for some tasks and so a reasonable effect can be obtained for different representations of the same situation.

Typical problems which can be tackled using a computer simulation are those with a large random element (i.e. problems which are dependant on unpredictable conditions) and which vary with time.

Simulation is now a recognised and widely used tool and a number of techniques have evolved whereby a system designer can set up a computer simulation.[11]

1.2 Special Purpose Languages

For the last few years, we have progressed from communicating with the computer on its own terms (computer machine language) to using the computer to convert some easily understood instructions to machine language on a one-to-one basis (assembler) to the present day practice in which one instruction can become hundreds or even thousands of machine language instructions (compiler).

Improved computer languages have permitted the solution of increasingly complex problems. Higher level computer languages such as FORTRAN, ALGOL and COBOL have saved a large amount of money and brought the price of computing down rapidly. Almost all data processing today is based on some form of higher level programming language.

The basic data-structures and control structures provided by languages like FORTRAN, ALGOL and COBOL do not at first sight lend themselves very conveniently to simulation applications. Simulation requires a degree of parallelism (real or conceptual) whereas these languages have a strictly sequential flow of control and simulation also requires some special data-structures, such as "queues".

For these reasons, special purpose simulation languages were developed.

A simulation language provides descriptors for describing the state of a system, language statements for modifying the state of the system, procedures for controlling the running of the system and for observing the results of the system.

One such language is the General Purpose Simulation System[5] (GPSS) which has been described as 'highly structured, relatively easy for the non-programmer to learn and includes many convenient features'.[16] Another language is SIMSCRIPT[14] which is a less structured language than GPSS and requires greater programming competence demanding more effort from the user. GPSS was developed by Geoffrey Gordon and released in its primitive form in 1961 by IBM. SIMSCRIPT was developed by the RAND Corporation and released in 1962.

There are numerous other special purpose simulation languages of which only a few are mentioned here.[21] MILITRAN[4] represented an attempt to develop a conversational language for war gaming, but is far from a complete simulation language. SIMULA[10] 1 has many of the properties of SIMSCRIPT, is based on ALGOL, but has little in the way of convenience aids. SIMULA 67 is a more useful and general version of the language. GASP[15], SOL[12] and GSP[20] all have their merits, but little general usage. They are based on general purpose languages. Familiarity with FORTRAN, for example, would simplify GASP. These languages, however, do not provide the user with as many tools as GPSS and SIMSCRIPT.

We shall, however, illustrate an example using a FORTRAN structured language called Control and Simulation Language[3,9](CSL).

A CSL program is divided into sectors called activities preceded by a set of initialisation statements. These activities describe the changes in state which are to take place when that activity occurs. There is a simulated clock which is automatically advanced to the time of the next event which is due to happen. The clock may not be advanced within the scope of an activity so that each instance of an activity constitutes an event.

The activity may be imagined to be constantly trying to occur. Before each list of changes are a set of test statements which determine whether or not the activity can occur at that instant. The result of a test is a choice of destination. Success transfers control to the next statement, failure transfers control to the first statement in the next activity. Activities interact with one another by setting and testing appropriate status descriptors. The successful completion of the tests of an activity initiates an event. Event timing is introduced by the use of special status descriptors called T-Cells which contain time variables. Events are scheduled by providing tests of specific T-Cells.

At the start of the simulation, control is given to the first statement of the first activity (ACTIVITIES identifies the point in the program). Each activity is entered in succession and is either executed or bypassed depending upon the outcome of test statements. BEGIN denotes the start of each activity. After the last activity at the end of a cycle, all T-Cells are scanned, the clock advanced by the least T-Cell value and all T-Cell time values correspondingly reduced.

For example, suppose we wish to simulate the following situation: Customers arrive at intervals prescribed by some probability distribution and join a queue with one server. The service times are drawn from some other probability distribution. Thus the customers have two activities: waiting in the queue and being served. The server has two activities: serving and being idle.

The complete coding for the above example in CSL would be of the form:

```
CLASS SERVER.1 SET BUSY,IDLE                  no. of servers = 1

Q=0                                             sets queue to empty
MAXQ=0                                          max length of queue
STREAMA=13           initial random number for customer arrivals
STREAMB=17               initial random number for service times
T.CUSTOMER=2             the time the first customer will arrive

ACTIVITIES
BEGIN WAITING
T.CUSTOMER EQ 0                                 continues if true
Q=Q+1
```

```
    T.CUSTOMER=DEVIATE(STREAMA,1,7)              normal distribution
    IF (Q .GT. MAXQ) MAXQ=Q                         (mean 7,s.d. 1)

    BEGIN SERVING
    Q GT 0
    SERVER.1 IN IDLE
    SERVER.1 FROM IDLE
    SERVER.1 TO BUSY
    T.SERVICE=RANDOM(STREAMB,10)      random number between 1 and 10
    Q=Q-1

    BEGIN IDLING
    T.SERVICE EQ 0
    SERVER.1 FROM BUSY
    SERVER.1 TO IDLE

    BEGIN FINISH
    CLOCK GE 180
    WRITE (3,20)MAXQ,Q
 20 FORMAT ('MAXQ = ',I3/'Q= ',I3)
    STOP
    END
```

1.3 Nature of Package

Since many computers do not have compilers for special purpose languages, it is not usually possible to carry simulation programs from one computer to another. They are also difficult to learn for the average programmer who knows just Fortran, Algol or Cobol. So if a programmer wishes to carry out a simulation in this way, he will have to learn another completely different language. They are also rather slow because they are interpreted and use up a large area of store in the computer.

It is therefore an advantage to use a commonly used language for a simulation process which does not have any of the disadvantages mentioned above. This means that all the special operations needed by a simulation language such as a clock, schedule of future events, queues and statistics and so on have to be programmed since they are not built into the language. However, this programming need only be done once and incorporated in a package of standard declarations. Once this has been done, simulation will be as easy as using a special purpose language.

Indeed for problems of any degree of complexity it will be easier because the general programming capabilities available are much more powerful than those of special purpose languages.

A language such as Algol is considerably faster in both execution and compilation than most of the special purpose simulation languages and we have evidence that FORTRAN is faster than GPSS[18]. Practically all computers have an ALGOL compiler as well as plenty of programmers who know the language itself. Since ALGOL-W[6] was the main language used in this establishment, I have used it for my simulation package.

The Computing Laboratory at St. Andrews is equipped with an IBM 360/44 computer running 44MFT[1]. It has 256K bytes of core storage and programs run in full core with 200K bytes available, background with 112K bytes or foreground with 88K bytes.

The Input/Output includes card input and lineprinter output. In addition there is a multi-access system which can be used for programs of less than 96K bytes using 2260 video display terminals as well as batch.

The simulation package, of which details follow, provides a very general program which can be used to simulate practically anything. The examples include such things as a supermarket, a telephone box, a road junction and other traffic problems.

The package itself consists of a series of declarations providing the user with a set of procedures, variables and record classes which he can use in any simulation problem. These declarations consist of a deck of cards which the user inserts before his own program and is compiled at the same time in ALGOL-W.

As in CSL, the main part of the program following the initialisations consists of a set of tests which determines when the activities which describe changes in the state of the system are to take place. If the test is successful then the activities are executed otherwise control is transferred to the next statement. Event timing is introduced by means of a diary which contains the times events are scheduled to occur. After each cycle of scans of the activities, the command "tick" advances the value of the clock to the time the next event is due to happen and the list of events is recycled.

The example program illustrated in CSL in the previous section can be written in ALGOL-W using the simulation package as follows:

```
--deck of cards provided--

record thing(logical dummy);
reference(queue)q;                                          1 queue
reference(facility)server;                                  1 server
integer r1,r2,arrival,depature;
r1:=13;                initial random number for customer arrivals
r2:=17;                    initial random number for service times
arrival:=1;
departure:=2;                                           event codes
initialise;
setupq("QUEUE",q);                            sets queue to empty
setupf("SERVER",server,1);                     sets server to idle
note(arrival,2);           the time the first customer will arrive
while clock<180 do
begin
  if at(departure) then release(server);
  if again(arrival,normal(7,1,r1)) then enter(q);         same
  if ¬empty(q) and idle(server) then              distribution
  begin
    seize(server);
    note(departure,uniform(1,10,r2));
    next(q)
  end;
```

```
   tick
 end;
 outstats
 end.
```

Notes: The declaration of record in the first statement will be explained in detail later, but some declaration of "thing" MUST appear.

The event codes 1 and 2 referred to by the variables "arrival" and "departure" represent respectively the event when another customer is entering the shop and when a server has finished serving someone.

"Outstats" gives information about the queue and server. (maximum length of queue, etc.)

Other points about the language are explained later in the text.

It is also possible to follow a particular 'thing' through the simulation - for example follow a person through a supermarket from when they pick up a basket at the entrance until they leave the checkout. Similarly it is possible to watch a car turning right at a road junction.

As well as producing a number of statistics on the various simulations which are carried out, there is also the facility for producing snap-shots at each "tick" showing the position of 'things' in the simulation. This is particularly useful if one is able to work interactively with the system and watch a situation develop such as a traffic jam and then step in, such as a policeman might do in practice, and alter the process manually by over-riding the system. It was not possible to actually try this out, however, since the computer was not big enough to allow the program to be run on the multi-access system.

The first part of this thesis deals with the program as a simulation package producing a set of results at the end, such as queue lengths and usage facilities. It consists firstly of a user's guide for the package and then goes into more details describing how it actually works. The second part is an extension of this idea with the same statistical information being output, but also with modifications to enable a pictorial representation of the model at each clock tick to be output. It also includes an addition to the user's guide to show him how to use this facility.

CHAPTER 2

THE SIMULATION PACKAGE - THE USER'S GUIDE

2.1 Introduction

The flow of control in an ALGOL program is from beginning to end, i.e. starting at the beginning, each statement is executed in order until the end one is reached. In a simulation process, a number of statements are required to be executed in an order which is not known in advance and at a particular point in simulated time, a number of statements may have to be executed 'simultaneously'.

These statements may be divided into two classes - those that are scheduled to occur at a particular moment in simulated time (time-dependant) - and those that are waiting for the system to reach a certain state before they can proceed (state-dependant). The two cases can be expressed in ALGOL thus:

```
if <at a specific time for an event to occur>then<do some action>;
if <a condition of an entity is true> then <do some action>;
```

The first is controlled by the clock and diary mechanism and is some event that is due to happen at a particular point in time such as someone entering a shop. The clock is an integer variable which gives the value of the simulated time during the simulation. The diary is a housekeeping mechanism which will be explained later and contains information concerning the time events are due to occur (i.e. it is a 'future-events' chain). The second kind of statement is concerned with activities which can be altered by other condition statements such as testing if a queue is empty or a facility free and so on.

So as to execute these statements in parallel in ALGOL, they should be placed in a *while* loop so that all events which are to be carried out at a particular time can be done at the same simulated moment, i.e. the value of the clock is unchanged for all the events:

```
while <not finished> do
begin
  if at(<event>) then <action>;              }   time
  .                                          }   dependant
  .
  .
  .
  if empty(q3) then <action>;                }   state
  if idle(server) then <action>;             }   dependant
  .
  .
  .
  tick
end;
```

When control passes through the body of the *while* loop, only those time-dependant statements scheduled to occur at the current value of "clock" are executed. The action following the time condition of a time test may change the value of one of the entities such as queue length or the state of a facility and enable state-dependant statements to be executed immediately afterwards.

For this reason, it is good policy to place time-dependant events before state-dependant events inside the *while* loop so that any side effects on entities which result, can be carried out without a further loop round the *while* block.

It is usually possible to place the other tests in such an order that it is only necessary to scan the set of tests once and be sure that no resulting side effect of a true test will affect the condition of a previously scanned test.

The procedure "tick", which is the last thing to be executed in this moment of simulated time, sets the clock to the next time for which an event is scheduled.

In general, the simulated system consists of a number of fixed objects (e.g. telephone box, cash desk, etc.) with a succession of transient objects that move through the system (e.g. people,cars or whatever). The package provides two sorts of standard 'fixed' objects - "queues" and "facilities" which have predefined procedures for operating on them and automatic collection of statistics (such as average length for a queue). The package assumes the existence of a record class of "things" defined by the user, which are the objects moving through the system. The nature and properties of "thing" are completely up to the user. A "queue" is defined by the package to be a queue of things. The user can create any number of queues, all initially empty. Each queue is given a name by which it is identified when the statistics are output. Similarly, any number of facilities can be created by the user each with a name which is used during the output of statistics and a capacity "max" which is the maximum number of "things" which can be using it at one time. Initially a facility is "idle".

In addition to the statistics for queues and facilities, which are collected automatically, the user can collect any other statistic he requires, for example the transit times of "things" through the system, by creating an entity called a "table" in which he collects information during the simulation.

At the end of the simulation the single command "outstats" produces a listing of the statistical information accumulated in all queues, facilities and tables.

2.2 Declarations

The package consists of a series of declarations of records, variables and procedures. The user will follow these by some declarations of his own as follows:

<u>record</u> thing (......);

The fields in brackets are the properties of "things" which pass through the system such as sex, age, etc. for people. By this method it is possible to keep track as to where a particular "thing" is in the simulation at a given time and thus evaluate the time taken to go through the system being simulated.

e.g. <u>record</u> thing (<u>logical</u> sex,old);

N.B. Even if this is not required, a definition of "thing" MUST be made: e.g. <u>record</u> thing (<u>integer</u> dummy);

If more than one type of "thing" is required, then a definition of the form:

<u>record</u> thing (<u>reference</u>(car,person,fruit)it);

could be made where car,person,fruit are seperate record classes defined by the user. Alternatively one could have a single record class containing the fields appropriate to every type of thing, though this would be rather wasteful.

<u>reference</u>(queue)......;

In place of dots are put the names of the queues used in the simulation:

e.g. <u>reference</u>(queue)q1,q2;

<u>reference</u>(facility)......;

This is where facilities are defined. Such things as telephone boxes, checkouts or other servers are included:

e.g. <u>reference</u>(facility)server,basket,trolley;

<u>reference</u>(table).....;

If a table of statistics is required, then a <u>reference</u> to "table" must be made. For example, if the user wishes to know how long it takes for people to go through a supermarket, he will declare a <u>reference</u> to "table, where this information can be stored:

e.g. <u>reference</u>(table)total_time;

<u>integer</u> <random_numbers>;

The user must declare random numbers if he uses any of the statistical distributions in the package:

e.g. <u>integer</u> r1,r2;

2.3 Initialisations

The following procedures provided by the package must be called by the user before he can use the relevant entities:

<u>procedure</u> initialise;

This sets up the initial values of the clock, diary and other housekeeping used by the package in the simulation and MUST be called.

procedure setupq(_string_(20)_value_ s;_reference_(queue)_result_ q);

For each queue which has been defined in the _reference_ statement, a "setupq" call must appear which initialises each queue to _null_ and gives a name specified by up to 20 characters given in quotes:

e.g. setupq("queue for telephone",q);

procedure setupf(_string_(20)_value_ s; _reference_(facility)_result_ server; _integer_ _value_ max);

For each facility which has been defined in the _reference_ statement, a "setupf" call must appear which initialises each facility to be empty and sets the maximum number of "things" it can hold represented by "max". It also gives each a name of up to 20 characters given in quotes:

e.g. setupf ("telephone",telephone,1);

procedure setuptable(_string_(20)_value_ s; _reference_(table)_result_ t);

For each piece of information for which results are required at the end of the simulation, a table must be set up:

e.g. setuptable ("time spent in queue",queuet);

setuptable ("time to make a call",callt);

The following variables also have to be initialised if required:

<random_numbers>

Random numbers used must each be initialised to an integer between 1 and 2^{31}-1 (2147483647):

e.g. r:=437621;

logical zero;

Zero is a logical variable, defined in the package, which is initially set to _true_ so that statistical distributions can give a result of 0. If "zero" is set to _false_, then if the result of a statistical procedure is 0, the procedure will be called again until the result is not equal to 0. This is of particular use when some "thing" is not wanted to appear at time CLOCK+0 (i.e. at that instant). Since the list of activities is only meant to be scanned once in each clock tick, if a statistical distribution does produce the value 0 then that event will occur immediately and may produce the error: "SIMULTANEOUS EVENTS IN DIFFERENT CYCLES". For example, it does not matter if more than one person enters a bus queue at a given instant and so "zero" would be set to _true_ and the test would be of the form: _while_ again(join_queue,poisson(1,random)) _do_ <activity>; whereas it would be impossible for two or more cars to be on a particular stretch of road at the same time and so in that case "zero" would be set to _false_ and the test would be of the form:

if again(entering_road,poisson(1,random)) then <activity>;

2.4 Simulation Procedures

The following procedures, provided by the package, are used in the simulation program to make use of the diary, queues and facilities:

procedure note (_integer_ _value_ event,time);

Note puts the "event" in the diary so that it will happen in "time" ticks.

logical procedure at (integer value event);

At is a logical procedure which returns the answer true if it is time for "event" to happen and false if not. If the answer is true, then it also removes the entry from the diary.

logical procedure again (integer value event; integer time);

Again is a logical procedure which, if it is time for "event" to happen, produces the answer true and "notes" "event" in the diary to happen in "time" ticks. If the "event" is nowhere in the diary, it will "note" the "event" to happen in "time" ticks.

procedure enter (reference(queue)value q);

Enter puts some "thing" onto the queue "q".

procedure next (reference(queue)value q);

Next takes the next "thing" off the queue "q".

logical procedure empty (reference(queue)value q);

Empty is a logical procedure which returns the value true if the queue "q" is empty and false if not.

logical procedure jump(reference(queue)value q; logical favourite);

Jump is a logical procedure which is used when it is required for some "thing" in the queue to jump to the top. The type of "thing" in the queue, which is called by 'name', is a logical expression which is referring to "thething": e.g. basket(thething) has value true or false depending whether or not "thething" has a basket. "Thething" has had one of its fields set up in the record declaration statement as "logical basket". This means that the procedure goes through the queue searching for some "thing" which has basket true. If "thething" which is being jumped is in the queue, then it is jumped to the top of the queue and will return the value true. The item nearest the top is jumped if there is more than one of the required type in the queue. The value of "thething" following the procedure call will be null.

procedure seize (reference(facility)value server);

Seize seizes a facility and makes it busier.

procedure release (reference(facility)value server);

Release releases the facility "server" and makes it freer for use.

logical procedure busy (reference(facility)value server);

Busy returns true if the number of "things" being served by "server" is the maximum that it can hold.

logical procedure idle (reference(facility)value server);

Idle returns the result true if the "server" is not doing anything.

logical procedure ready (reference(facility)value server);

Ready returns the value true if the facility "server" has not been accessed yet at this moment of simulated time and the facility is "busy".

procedure tick;

Tick sets the CLOCK equal to the next time something will happen: i.e. the time of the next event in the DIARY. It also tests if the clock does not change between cycles by comparing its original value with the new one. If it hasn't changed then the message "SIMULTANEOUS EVENTS IN DIFFERENT CYCLES" is output and means that the list of activities is being recycled.

2.5 Random Number Generators

An important part of a simulation package is the statistical distributions. These can be used to help the user simulate non-uniform arrival or service times which may approximate to some statistical distribution. In each of these procedures, "r" is one of the random numbers provided by the user and after each call, its value is changed so that a different result will be calculated the next time that that random number is used.

The following are included in the package:

real _procedure_ random (_integer_ _value_ _result_ r);

Random returns a real number between 0 and 1.

logical _procedure_ perhaps (_real_ _value_ f; _integer_ _value_ _result_ r);

Perhaps returns the value _true_ f of the time and _false_ 1-f of the time. (i.e. f lies between 0 and 1)

integer _procedure_ uniform (_real_ _value_ min,max; _integer_ _value_ _result_ r);

Uniform produces a random number between "min" and "max" according to a uniform distribution.

<u>integer</u> <u>procedure</u> exponential (<u>real</u> <u>value</u> mean; <u>integer</u> <u>value</u> <u>result</u> r);

Exponential produces a random number according to an exponential distribution with mean "mean".

<u>integer</u> <u>procedure</u> normal (<u>real</u> <u>value</u> mean,sd; <u>integer</u> <u>value</u> <u>result</u> r);

Normal produces a random number according to a normal distribution with mean "mean" and standard deviation "sd". At the present, this procedure cannot produce negative numbers. If any are produced, the procedure is evaluated again and a new result obtained. This is to prevent events being put in the diary to occur at a time which has passed, since this procedure will be used mainly for times. The package could be modified to include a variable similar to "zero" which could control this.

<u>integer</u> <u>procedure</u> poisson (<u>real</u> <u>value</u> mean; <u>integer</u> <u>value</u> <u>result</u> r);

Poisson produces a random number according to a Poisson distribution with mean "mean".

The user can declare his own distributions as procedures but they must be called by different names if defined in the main block. The user can declare his own random number generator if he wishes by physically substituting it for the existing random number generator in the package.

2.6 Statistical Routines

These routines are concerned with the collection and output of statistics collected during the simulation program:

<u>procedure</u> collect (<u>reference</u>(table)<u>value</u> t; <u>integer</u> <u>value</u> n);

Collect is called when it is wanted for some value to be noted and then written out in tables at the end of the simulation. For example, one might want to know how long it takes some "thing" to go through the system, so one would use "collect" to collect this information and then work out the number, the maximum, the minimum, the average and the standard deviation of the numbers that had been collected for this particular table.

<u>procedure</u> tempstats;

Tempstats can be used at various points in the program to print out the values of various entities in the simulation which concern queues and facilities. It will print out the value of the clock and then for queues, their lengths and for facilities, the number of "things" which are using them.

<u>procedure</u> outstats;

Outstats prints out statistical information at the end of the program.
It prints for queues the average length, average waiting time of a "thing" in the queue, maximum length, length at the end of the simulation as well as the total number of "things" that have been through the queue.
For facilities, it prints the number of users, the average use and average time spent in the facility.
For tables, the number, maximum, minimum, average and standard deviation of the numbers which have been "collected" during the simulation are printed.

For example, if we want to collect the time taken for people to go through a shop, then we might write code as follows:

```
record thing (integer start);          "start" will contain the time
.                                      that "thing" entered the shop
.
.
reference(table)shoptime;
reference(facility)shop;
integer arrival,departure;
initialise;
setuptable ("total time in shop",shoptime);
setupf ("shop",shop,30);
.
.
.
arrival:=1;
departure:=2;
while <not finished> do
begin
  .
  .
  .
  if again (arrival,<time>) then
  begin
    thething:=thing(clock);
    seize(shop);
    .
    .
  end;
  .
  .
  .
  if at(departure) then
  begin
    release(shop);                               Collects the time
    collect(clock-start(thething),shoptime);       "thing" took to
    .                                                 pass through
  end;                                                    the shop
  .
  .
  .
  tick
end;
outstats
end.
```

The output obtained would be of the form:

```
           --statistics about queues and facilities--

                              TABLES

                        NUMBER   MAX   MIN   AVERAGE  STANDARD DEVIATION

TOTAL TIME IN SHOP        40      43    10     21.6          8.6
```

2.7 Things

A variable called "thething" which is a global reference to a record defined by the user called "thing" defines the "thing" which is being processed at the current time. That is, for example, if "next(q)" is being processed at the current time, then "thething" will point to the "thing" which is being taken off the queue. If the value of "thething" is set up before being put on the queue, then when "enter(q)" is called, it will remember that that particular "thing" has been put on the queue at that point.

Queues keep a pointer to the type of "thing" that is in the queue while it is in the queue. When a "thing" is in a facility whose capacity is more than 1, some entry will have been made in the diary to note the time when that "thing" will release that facility. It is here, in the diary, that a note is kept of what "thing" is being processed.

For each element in the diary, there is a pointer to "thing" which gives the reference to "thething" for that particular time.

2.8 Side effects on "things" by procedures

Initialise sets "thething" to null.

At puts "thething" equal to the kind of "thing" which the time is at for it to occur.
(The field in the diary which points to "thething" is called "kind" and has been defined as: reference(thing)kind.)

Note puts "thething" into the diary when it places in the diary the time that the event will happen.

Again will put "thething" into the diary if it "notes" the event. The value of "thething" will not be affected.

Tick sets "thething" to null ready for the next time when the next events will happen.

Since queues and facilities do not know in advance how long they are going to keep a "thing" in them, they must have the means of storing that information. So, for each "thing" in the queue or facility for which it is required to know the type of thing that is there, a queue is made with each member of the queue consisting of a pointer to the type of "thing" in the queue and a pointer to the next member of the queue and a facility has a pointer to the type of thing in it. If it is not required to know what is in a queue, then these extra pointers are eliminated and only the header of each queue is present. When they are required then "next" and "enter" respectively add "thething" to the queue or set "thething" equal to whatever is removed from the queue. Similarly facilities have a null pointer in their respective position if it is not required and "seize" and "release" respectively put "thething" into the facility and set "thething" equal to whatever was in the facility. If there are not types in the queue or the facility when "next" or "release" are used, then "thething" remains unchanged.

It is only possible, however, to note "things" in facilities with capacity equal to 1 since there is only one cell in which this information can be stored. If it is necessary to store information in facilities of capacity more than 1, then they must be defined as seperate facilities of capacity 1 each. However, since most facilities are finished with after a known period of time, although this may only be known indirectly, it is not necessary to store that information in the facility. For example, we don't know how long a person is going to be in a shop when he enters it, but we do know how long he is going to take to serve himself, because that is evaluated at the beginning according to some distribution, and after that time, the time taken to queue up, although it is not known, the "thing" is then added to the queue and so not lost. When "thething" reaches the checkout, the time taken to go through the checkout is evaluated when the "note" call is used. This means that the particular "thing" which is going through the shop, although not noted in one stage, is noted as it goes through each stage of the simulation - from entering the shop, queueing at the checkout and going through the checkout. This is firstly noted in the diary, then by the queue and finally by the diary again. Facilities of capacity more than 1 always have null in this cell.

2.9 Example run on the Computer

The system to be simulated which demonstrates the package is a telephone box. Users arrive at the box according to an exponential distribution with mean value 5 minutes. The time each caller spends in the telephone box is represented by a uniform distribution and depends on whether they are male or female. A male caller spends between 1 and 3 minutes making a call whereas a female caller spends between 1 and 9 minutes, The unit of time will be a minute and the system is to be simulated for 6 hours.

The program begins with the declaration of "thething" which will hold the information as to the sex of a person and also carries the time the person entered the queue so that information about the time the person spent in the system can be kept in a table and results obtained at the end. There are also declarations of the queue and telephone box as well as random numbers. The two events "entering" and "leaving" represented by the codes 1 and 2 respectively refer to entering the queue and leaving the telephone box.

When a person enters the queue, their sex is decided and a note of their arrival time is kept in "thething". On exit from the telephone box, the time spent altogether, queueing and making the call is collected in the table.

From the output results we can see that the average length of the queue was 0.59, the average waiting time in the queue was 3.29 minutes. The maximum length of the queue was 5, there were 65 people altogether queueing of which there were none left at the end. 64 people used the telephone, which was in use 66% of the time. The average time spent making a call was 3.73 minutes and there was one person left in the telephone box at the end. The total number of people who went through the whole system was 64 and the longest took 20 minutes and the shortest 1 minute. The average and standard deviation of these times was 7.02 and 4.07 respectively.

```
0310 --          COMMENT EXAMPLE 1: SIMULATION OF TELEPHONE BOX;
0310 --
0310 --        RECORD THING(LOGICAL MALE; INTEGER START);
0312 --        REFERENCE(QUEUE)Q;
0313 --        REFERENCE(FACILITY)TELEPHONE;
0314 --        REFERENCE(TABLE)T;
0315 --        INTEGER R1,R2,R3;
0316 --        INTEGER ENTERING,LEAVING;
0317 --        ENTERING:=1; LEAVING:=2;
0319 --        INITIALISE;
0320 --        ZERO:=FALSE;
0321 --        R1:=45631;
0322 --        R2:=564732;
0323 --        R3:=7632486;
0324 --        SETUPQ("QUEUE FOR TELEPHONE",Q);
0325 --        SETUPF("TELEPHONE",TELEPHONE,1);
0326 --        SETUPTABLE ("TOTAL TIME SPENT",T);
0327 --        NOTE(ENTERING,4+EXPONENTIAL(5,R3));
0328 --        WHILE CLOCK<361 DO
0328 2-        BEGIN
0329 --          IF AGAIN(ENTERING,EXPONENTIAL(5,R3)) THEN
0329 3-          BEGIN
0330 --            THETHING:=THING(PERHAPS(0.5,R1),CLOCK);
0331 --            ENTER(Q)
0331 -3          END;
0332 --          IF AT(LEAVING) THEN
0332 3-          BEGIN
0333 --            RELEASE(TELEPHONE);
0334 --            COLLECT(T,CLOCK-START(THETHING))
0334 -3          END;
0335 --          IF ¬EMPTY(Q) AND IDLE(TELEPHONE) THEN
0335 3-          BEGIN
0336 --            NEXT(Q);
0337 --            SEIZE(TELEPHONE);
0338 --            NOTE(LEAVING,UNIFORM(1,IF MALE(THETHING) THEN 3 ELSE 9,R2))
0338 -3          END;
0339 --          TICK
0339 -2        END;
0340 --        OUTSTATS
0340 -1      END.

EXECUTION OPTIONS: DEBUG,0 TIME=97369 PAGES=32767
```

```
THE VALUE OF THE CLOCK AT THE END OF THE SIMULATION IS   362

                                 QUEUE STATS

                        AV. LENGTH  AV. WAIT TIME   MAX       USERS   LENGTH AT END
QUEUE FOR TELEPHONE       0.59          3.29          5        65          0


                         FACILITY STATS

                          USERS     AVERAGE USE  AV.TIME SPENT  USERS AT END
TELEPHONE                  64          0.66         3.73            1


                              TABLES

                         NUMBER       MAX          MIN       AVERAGE STANDARD DEVIATION
TOTAL TIME SPENT           64          20            1          7.02         4.07

001.16 SECONDS IN EXECUTION
/&                                                                                20.27.37






********************    ST.ANDREWS UNIVERSITY COMPUTING LABORATORY - 44MFT/SPOOLER END-OF-JOB RECORD    **********
******************** JOB NAME : GHSIM       DATE : 75276      START TIME : 20.26.56   ELAPSED : 00.00.42 **********
********************    CPU TIME : 00.00.13      WAIT TIME : 00.00.19      SYSTEM OVERHEAD : 00.00.06    **********
********************    PAGES PRINTED :     21      CARDS PUNCHED :      0      CARDS READ :    549       **********
********************    A GRAND TOTAL OF        1912  INPUT/OUTPUT REQUESTS WERE HANDLED FOR THE JOB.   **********
********************    THE AVAILABLE CORE WAS : 112 K      OF WHICH AT LEAST    66 K WAS USED          **********
********************                    THE APPROXIMATE COST OF THIS JOB WAS  £  0.39                     **********
********************                 THIS JOB WAS EXECUTED IN THE BACKGROUND PARTITION                    **********
* THIS LINE PRINTED AT 20.45.43  ;
```

CHAPTER 3

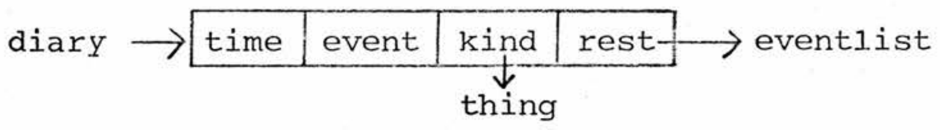
THE IMPLEMENTATION OF THE SIMULATION PACKAGE

This chapter describes how the housekeeping of the package is controlled.

3.1 The Diary

This is a record class internal to the housekeeping. Its fields need never be accessed explicitly by the user.

```
record eventlist (integer time,event; reference(thing)kind;
                                        reference (eventlist) rest);
reference (eventlist) diary;
diary:=null;
```

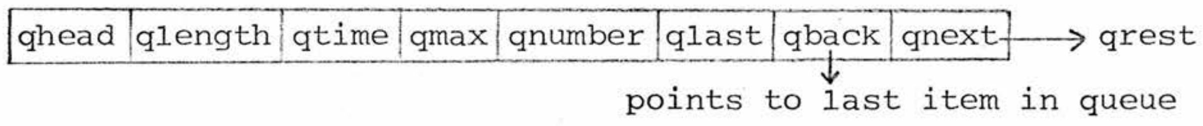


Symbol	Meaning
time	the time when the activity is completed.
event	the activity being performed.
kind	a pointer to the record class "thing" defined by the user, which gives information as to the "thing" in the activity being performed.
rest	a pointer to the next activity to be completed following the present one.
diary	a pointer to the first activity to be completed in the diary.

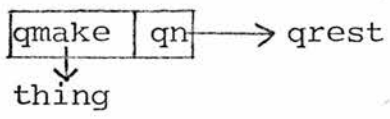
3.2 Queues

```
record queue (string(20)qhead; integer qlength,qtime,qmax,
     qnumber,qlast;  reference (qrest) qback,qnext);
record qrest (reference(thing)make; reference(qrest)qn);
reference (queue) SUPPLIED BY USER;
```

QUEUE



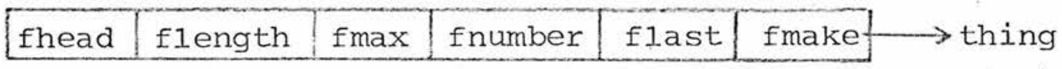
QREST



Symbol	Meaning
qhead	a piece of text, maximum length 20, which contains the name of the queue. It is used for error messages e.g. when something is wanted from the queue and it is empty, then a message:'"QHEAD" IS EMPTY' is output where "QHEAD" is replaced by the name of the queue. It is also used for outputting statistics.
qlength	the length of the queue at the moment.
qtime	the total time spent queueing in that queue so far.
qmax	the largest length of the queue so far.
qnumber	the total number of "things" which have entered the queue so far.
qlast	the time that the last thing happened to the queue.
qback	a pointer to the last item in the queue.
qnext	points to the first item in the queue.
qmake	points to the "thing" at that point in the queue.
qn	points to the next member of the queue.

3.3 Facilities

```
record facility (string(20)fhead; integer flength,ftime,fmax,
     fnumber,flast; reference(thing)fmake);
reference(facility) SUPPLIED BY USER;
```



Symbol	Meaning
fhead	a piece of text, maximum length 20, which contains the name of the facility. It is used for error messages and when statistics are output at the end of the program.
flength	the number of servers busy at the moment. (For example, if there are two checkouts at a super-market, then flength can be 0,1 or 2)
ftime	the total time things have spent in the facility so far.
fmax	the total number of servers that there are in the facility.
fnumber	the total number of "things" which have used the facility.
flast	the time that the last thing happened to the facility.
fmake	points to "thething" which is in the facility.

Examples: In a supermarket, the facility is 'checkout' and the servers are the people at the cash registers. For telephone boxes, the facility is 'telephone box' and the servers are each individual one.

All the variables used by queues, facilities and the diary CANNOT be defined by the user in his program in his main block.

3.4 How the random numbers are generated

3.4.1 Random

```
real procedure random (integer value result r);
begin bits j;
  bits procedure xor (bits value p,q);
  ¬ (p and q) and (p or q);
  j:=bitstring(r);
  j:=xor(j shl 18,j);
  r:=number(xor(j shr 13,j) and #7FFFFFFF);
  r/2147483647
end;
```

The routine calculates pseudo random numbers by the Tauseworth Generator[19] and this coding originated from the PL360 procedure 'random'[8]. This particular random number generator was used because it has been shown that it is fast, short and passes a large number of statistical tests[22].

The procedure produces random numbers from various juggling of bits of the integer which is initially entered into the routine.

Firstly, it shifts the number 18 bits left and does an exclusive 'or' on that number and the original number. It then does a shift right of 13 on that number and an exclusive 'or' with it. Finally the number which is left is 'and'ed to #7FFFFFFF (hexadecimal for 2^{31}) which makes it positive. (All the bits except the first are 1 and so all the bits of the final answer will be what they were before except for the top bit which will be 0.)

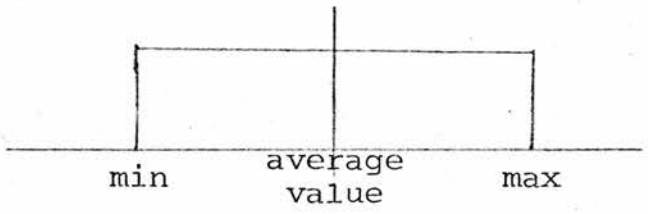
The word size on the IBM 360/44 is 32 bits with the top bit representing the sign. Thus the largest possible integer is 2^{31}-1 that is 2147483647. Since this routine calculates using all the bits, it will need a number put into it between 1 and 2^{31}-1 and will subsequently produce a number in the same range. However, for the purposes of the simulation, only a number in the range 0 to 1 is required and division by this number is performed at the end of the routine. The number that was produced in the range 1 to $2^{31}-1$ is returned via the <u>integer</u> <u>value</u> <u>result</u> parameter "r". So the procedure has the side effect that it not only returns with an answer between 0 and 1 as the real number of the type procedure, but it changes the random number which is put into it as a parameter. This means that the same random number can be used for each cell of the routine with the knowledge that it has been changed since the last call of the routine and will not produce the same value.

Although this means that the user of the package must define random numbers if he uses any of the statistical routines, it does mean that he can use different random number sequences for different parts of the program. For example he can use one random number sequence when working out the time between people entering a shop and a different sequence for the service times of people at the checkout. This also has the advantage that if he wishes to examine any particular occasion such as a particular sequence of service times at the checkout, he can keep the sequence of random numbers for the checkout constant whilst changing the sequence used for people entering the shop.

<u>3.4.2 Uniform</u>

```
integer procedure uniform (real value min,max;
                            integer value result r);
round(min+(max-min)*random(r));
```

This function will produce a random number between "min" and "max" (being an integer) with equal probability for all numbers.



In this and all remaining procedures, the value is calculated as a real number and then rounded to the nearest integer. This is because it is only necessary to have integer results since the clock operates in definite ticks and it is not possible to have a time which is a fraction of a tick. If it is required to have more accurate timing than one tick, then the time interval of the clock can be changed to a smaller interval.

3.4.3 Poisson

```
integer procedure poisson (real value mean; integer value
                                                  result r);
begin real r1,p,e; integer n;
  n:=0;
  r1:=random(r);
  p:=e:=exp(-mean);
  while r1>p do
  begin
    n:=n+1;
    p:=p+e*mean**n/
    (begin real f; comment this evaluates factorial(n);
      f:=1.0;
      for i:=2 until n do f:=f*i;
      f
    end)
  end;
  n
end;
```

Although this procedure may appear to rather long, its activity is relatively simple. The formula for the Poisson distribution is:

The probability that r or more events are contained in an interval when the average number of such events is "m" is given by:

$$p = \sum_{n=r}^{\infty} e^{-m} \frac{m^n}{n!}$$

This formula must be reversed so that n = some function(p) since a random number between 0 and 1 will be entered into the procedure and the result will be an integer value distributed according to a Poisson distribution with a mean of "m".

The procedure uses the formula to add to p according to the formula overleaf until the value of p equals the random number r1.

3.4.4 Normal

```
integer procedure normal (real value mean,sd;
                          integer value result r);

begin real y;
  y:=0.0;
  for i:=1 until 12 do y:=y+random(r);
  y:=y-6.0;
  y:=y*sd+mean;
  round(y)
end;
```

This procedure computes a normally distributed random number with a given mean and standard deviation ("sd"). It originated from the FORTRAN Scientific Subroutine Manual as SUBROUTINE GAUSS[2].

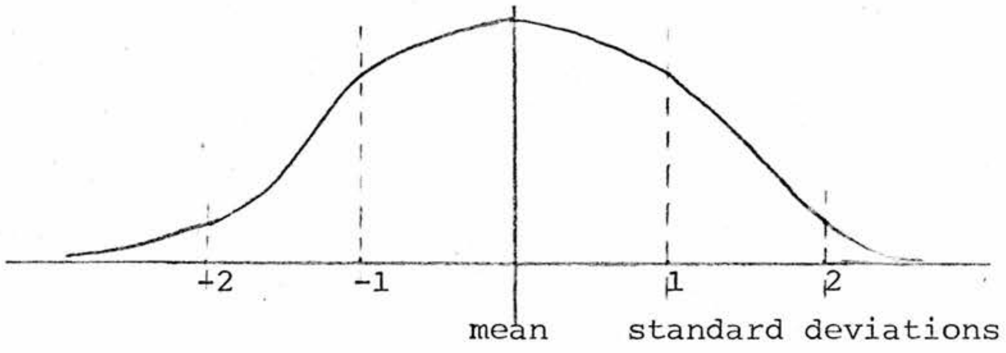
An approximation to normally distributed random numbers "y" can be found from a sequence of uniform random numbers using the formula:

$$y = \frac{\sum_{i=1}^{k} x_i - \frac{k}{2}}{\sqrt{\frac{k}{12}}}$$

where x_i is a uniformly distributed random number between 0 and 1; k is the number of values of x_i used.
y approaches a true normal distribution asymptotically as k - .
For this reason, k was chosen as 12 to reduce execution time.
Thus the simplified formula is:

$$y = \sum_{i=1}^{12} x_i - 6.0$$

The adjustment for the required mean and standard deviation is then: y':=y*s+am where y' is the required normally distributed random number; s is the required standard deviation and am is the required mean.



3.4.5 Exponential or Poisson process

```
integer procedure exponential (real value mean;
                               integer value result r);
round(-mean*ln(random(r)));
```

The negative exponential function is evaluated by the formula:

$$f(x) = \lambda e^{-\lambda x}$$

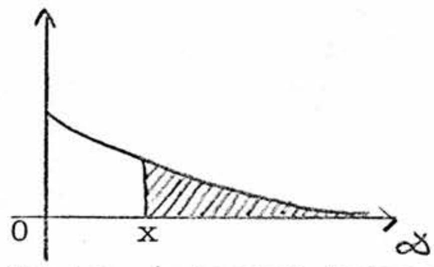
where the mean and standard deviation is $\frac{1}{\lambda}$.
This formula must be reversed so that x is in terms of f.

Since f(x) gives us the value of the function of x at a particular point, we need to integrate f(x) to get the total area under the curve. We then want to integrate this limit from 0 to x so that the value of the integral is a random number between 0 and 1.

If we say that F is the integral of f(x) then:

$$F = \int_{x=x}^{\infty} \lambda e^{-\lambda x}$$

$$= \left[\frac{\lambda e^{-\lambda x}}{-\lambda}\right]_x = e^{-\lambda x}$$



evaluates integral between x and ∞.

$$\log_e(F) = \log_e(e^{-\lambda x})$$
$$= -\lambda x$$
$$x = -\frac{\log_e(F)}{\lambda} \qquad \text{Now the mean} = \frac{1}{\lambda}$$
$$x = -\text{mean}*\log_e(F)$$

3.4.6 Perhaps

```
logical procedure perhaps (real value f; integer value result r);
random(r) < f;
```

This procedure computes a random number (which is between 0 and 1) and if this is less than f (which is also between 0 and 1), then _true_ is returned, otherwise _false_ is the result.

3.5 Tables

```
record table (string(20)thead; integer tmax,tmin,ttotal,tnumber,
                                                        ttotalsq);
```

Tables keep a note of statistics which the user requires to collect during the simulation. Such things as how long "things" take to go through the system, how many of a particular type there are, etc. The tables are set up using the "setuptable" procedure explained earlier and define the tables which the user wants. The initial value of tmax is set to 0 and tmin to 100000.

Information is put into the table using the procedure "collect" which was also explained earlier.

thead	tmax	tmin	ttotal	tnumber	ttotalsq

Symbol	Meaning
thead	A piece of text, maximum length 20, which contains the name of the table. It is used at the end of the program by describing which table the figures refer to.

tmax	The maximum value of the information collected.
tmin	The minimum value of the information collected.
ttotal	This is the total of the information collected.
tnumber	This is the total number of times collect has been called. (i.e. the number of numbers which have been collected)
ttotalsq	This is the total of (the numbers squared). (i.e. the numbers are squared and then added to ttotalsq)

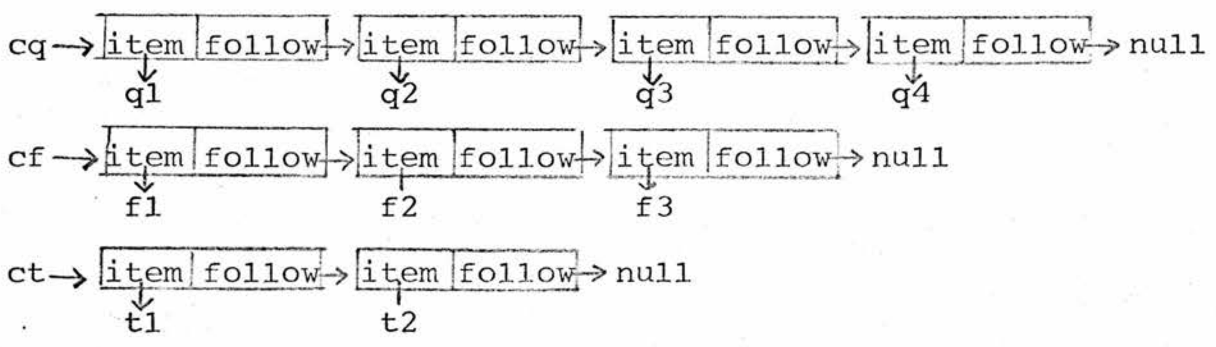
3.6 Chains

So that "outstats" (the procedure used at the end of the simulation to get the statistics output) need only be called once and does not need seperate calls to get information from individual queues, facilities and tables, a record "chain" is defined which links these all together:

```
record chain (reference(queue,facility,table) item;
                                        reference(chain)follow);
reference(chain)cq,cf,ct;
```

When "setupq", "setupf" and "setuptable" are used, they automatically set up links in the chain which point to the queues, facilities and tables.

Each are referred to seperately by pointers cq, cf and ct:



This means that when "outstats" is called, a pointer starts with "cq" and follows the pointer down all the queues until it reaches a <u>null</u>. It then traces "cf" and then "ct" until it finally finishes the outputting of the statistics.

CHAPTER 4
MODIFICATIONS TO THE USER'S GUIDE TO ENABLE DRAWINGS TO BE OUTPUT

4.1 Introduction

When simulating any particular system on a computer, or by any other method, it is useful to be actually able to see what is happening to the model during the simulation. Up to now, it has only been possible to obtain statistics of various entities at the end of the simulation run and, although this is very useful, it does not tell us what actually happens, just what happens on average.

For this reason, I have introduced the facility which enables 'snap-shots' of the system at each clock tick to be output. That is a picture of the simulation model, showing the position of the "things" at that particular moment in simulated time. The user can specify which part of the model he would like to see in diagramatic form.

In order to produce pictures of the system, it is necessary to introduce one or two changes to the previous package. The main one is the ability to show on the diagram "things" which travel at a constant speed through the system - enter at one end at a certain time and then come out of the other end a known time later. For example, a car driving down a stretch of road where nothing can affect its speed will be at the end of that stretch at a particular time. In the previous package we would probably have represented this total stretch of road as a facility which could, for example, contain up to 10 cars. Likewise, we may have just noted the event in the diary that a car would be ready to do something else a certain time later and in fact would be disappearing from the model for a given length of time while it is waiting for an "at" command to give it something else to do.

When a picture is being produced of the system, however, "things" should not be allowed to disappear from the system for a certain length of time as the user would wonder where they had disappeared to. So, instead, there must be a new entity which could represent "things" in the system moving at constant speed (say a square at a time) through the system. For this, I have introduced the record structure called a "belt" which when something is "put" on the belt, will move down the belt, one position at a time, until it is time for the "thing" to be taken off the belt and have some other operation done to it.

When something is "put" on a belt, there will automatically be a "note" put in the diary that there will be something coming off the belt at some time later dependant on the length of the belt. So, if the belt was four squares long, then there would be a note put in the diary that some "thing" was going to be entering the system again in four ticks.

"Things" on the drawing are represented by arrows depending on their direction at a particular point in time. This will be described further in the chapter as well as other modifications which have been made.

Since I thought about the need for pictures mainly in connection with simulations of traffic systems, I have also introduced the idea of "signals" which the user can use to represent traffic lights, etc. in his picture. Their value can be stop or go represented in the picture by @ and O respectively.

4.2 Requirements for drawing

"Things" on the drawing are represented by arrows according to which direction they are heading. Each square containing a "thing" must know which direction it is heading so that the appropriate symbol can be displayed on the drawing. If a direction is not necessary, a "*" can be used merely to symbolise that there is something at that point.

The following key is used to represent the direction of the "things" in the drawings:

Symbol	Meaning	Representation on Drawing
1	up	A
2	down	V
3	right	>
4	left	<
5	doesn't matter	*

4.3 Changes to facilities

So as to produce appropriate arrows for "thethings" inside facilities in the drawing, they must also have parameters which represent their direction and for this extra parameter is added to the facility named "fd". This means that the "setupf" call must also include a parameter indicating the direction of the facility and thus uses the key above.

"Setupf" now becomes: <u>procedure</u> setupf(<u>string</u>(20)<u>value</u> t; <u>reference</u>(facility)<u>value</u> <u>result</u> server; <u>integer</u> <u>value</u> max,direction);

If the direction of a facility can be more than one, for example at a crossroads intersection, then 0 can be input as the direction and will be initialised to 5 and will depend on the "thing" which happens to be in that facility at a particular time.

Because of this change to facilities, the procedures referring to facilities except "release" now need a further parameter representing their direction.

```
procedure seize(reference(facility)value server; integer value d);
```

This seizes facility "server" in the direction "d". If d=0 then the value of "fd(server)" will be unchanged, otherwise it will be set equal to "d".

```
logical procedure busy(reference(facility)value server;
                                          integer value d);
```

This returns the value true if the facility "server" is fully occupied (i.e. the number of "things" which the server is serving is the maximum which it can hold) and also either d=0 (which means that it doesn't matter which direction the facility is) or d is equal to the direction of the facility.

```
logical procedure idle(reference(facility)value server;
                                          integer value d);
```

This returns the value true if the "server" is not doing anything or $d \neq 0$ and the direction of the server equals d.

```
logical procedure ready (reference(facility)value server;
                                          integer value d);
```

This returns the value true if the facility has not been accessed so far at this moment of time and that there is something in the facility and that either d=0 or the direction of the facility. (This procedure is most useful if some "thing" is wanted to stay in a facility for just one tick only, so that the order of statements in the simulation program will not matter, since the statements following the "ready" test will not be executed at the same clock tick as the facility is seized, but will instead be done on the next scan of the routine.)

4.4 Belts

So as to represent some "thing" moving through the system at a constant speed - i.e. one square per tick - a new entity has been created which represents this called a "belt".

For each belt the user requires, a declaration must be made in the following way:

```
          reference(belt)......;
          e.g. reference(belt)b1,b2;
```

Each of these belts must be initialised by calling:

```
procedure setupbelt(reference(belt)result b; integer value s,d);
```

For each "belt", its length, "s", and direction, "d", are declared and set up inside the computer. If the length is 3 then this means that it will take 3 clock ticks for a "thing" to pass from one end to the other.

For example 'setupbelt(b,4,2);' sets up a belt 4 units long in a downward direction.

Procedures referring to "belts" which are used in the simulation are:

<u>procedure</u> put(<u>reference</u>(belt)<u>value</u> b; <u>integer</u> <u>value</u> event);

This procedure will put some "thing" onto the belt. If it is "full", then "things" will be queued on the front. It will also "note" the "event" that a "thing" will leave the belt in the diary.

<u>procedure</u> take(<u>reference</u>(belt)<u>value</u> b);

This procedure removes some "thing" from the belt.

<u>logical</u> <u>procedure</u> full(<u>reference</u>(belt)<u>value</u> b);

This procedure returns the value <u>true</u> if the belt has its first position full. That means that either the belt is full, or something has already been put on the belt this tick.

Each time "tick" is called, all "things" in belts are moved one place along the belt automatically. If there is nothing else to do, the simulation part of the program will not be scanned, but the "things" on the belts will be moved the appropriate number of steps and the appropriate number of pictures printed. This means that, as before, the statements in the program are not scanned if there is nothing to do.

4.5 Signals

To represent traffic lights or other signals in the system, an entity called "signal" is introduced. For each signal the user requires, a declaration must be made of the form:

<u>reference</u> (signal)......;

e.g. <u>reference</u>(signal)lights;

These are set up by calling the procedure:
procedure setupsignal (reference(signal)result s; logical value go);
which initialises the signal to the value of the logical "go"
which is either true (which means green) or false (red).
For example: 'setupsignal(lights,true);' will initialise the
signal "lights" to green.

Procedures referring to "signal" are:

procedure change (reference(signal)value s);

This procedure changes the value of the signal "s" from either green to red or red to green depending upon its value when called.

logical procedure red (reference(signal)value s);

This is a logical procedure which returns the value true if the signal "s" is red and false if it is green.
Since the parameter of signal is "green", then if green(s) is called, it will refer directly to the record and return true if the signal is green and false if it is red.

4.6 Data required for drawing

At the start of the package, there is a 'read' statement which reads in free format four integers: Height, Width, Page height and Page width. These refer to the height followed by the width of an individual picture and the actual height and width of a computer page. This enables the initial picture size to be declared as:

```
string(3)array pic(1::height,1::width);
```

which will be used to read in the initial picture from cards, where each square of the picture is represented by three columns of the card. A further array declared as:

```
string(1)array picout(1::pagew,1::hheight);
```

where hheight=height+4, is used to display the pictures on the line-printer. The height "hheight" enables spare lines to be kept between pictures on a page.

As stated previously, each square of the picture is represented by three columns on the data card. The first of these columns represents the type of square that particular one represents, i.e. queue, facility, signal or belt, or any other symbol which will be reproduced exactly on the output picture. For example "_" could be used as a border. It can consist of the following specific symbols:

	B	which represents a belt;
	Q	which represents a queue;
	F	which represents a facility
and	S	which represents a signal.

The second and third columns of the square depend on what the first column was. For belts, the second column represents the number of the belt in the order 1,2,3,4,5,6,7,8,9,A,B,C,....,Z. (i.e. 35 belts can be represented) The third column indicates the direction of the belt starting from 1 and continuing to Z. (up to 35 squares per belt). If this third number is 0 (zero) then this square is used to stack "things" when the belt is full and is usually at the edge of pictures where the correct number of arrows indicating "things" cannot all be represented on the drawing and so a number are stacked at the edge, for example: 4>>> would be short for >>>>>>>.

These squares would be represented on the data card of the picture as: B10B11B12B13B14.

For queues, signals and facilities, both the second and third columns are combined to give a number of the queue, facility or signal between 1 and 99. However, the first queue is represented by Q1 and not Q 1 and similarly for all queues, signals and facilities numbered 1 to 9.

The number of the queue, facility, signal or belt is dependant upon the order in which they are declared in the "setup" calls. Thus in the following "setup" statements:

```
setupq ("queue",q1);
setupq ("another",q);
setupq ("one",q2);
```

Q2 would be the queue "q" on the data card used for the picture and so on.

4.7 Other changes which the user will need to make

In the previous package, one of the begin statements was included in the package and required a corresponding end. card at the end of the user's program. In this package, two begins are included and so the user will need to end his program with end end. in order to match these two begins.

Since the package includes the facility for printing a picture at each tick, the procedure "tempstats" as provided in the previous package has been removed.

4.8 Example involving drawings

The system to be simulated to demonstrate the use of obtaining pictures is the same telephone box system that was described at the end of Chapter 2.

The system consists of a telephone box with a path up to it which can hold 4 people and a path away from it of capacity 2. The paths are represented as "belts" of lengths 4 and 2 and are declared and setup initially at the start of the program.

Since it takes 4 ticks for a person to walk up the path, ("things" move up a belt one position at a time) the delay of 4 ticks in the initial "note" statement has been removed. "Things" are placed on the belt according to the same distribution as the first example and when they have walked up the path, enter the queue as before. "Things" are not removed from the "belt" until they can enter the telephone, otherwise they would disappear from the picture. When a "thing" finishes a call, it enters a path of length 2 before finally leaving the system.

The events coded 1 to 4 are in this case entering, queueing, finishing and leaving which refer to entering the path, reaching the queue, finishing the call and leaving the system.

Although there is not enough space to show all the pictures for the 6 hours, a few have been shown including the same statistics as before.

The data cards provided for this program were as follows:

```
6  4  44  111

         -  -  -
_  _  _     F1    _
B10B11B12B13B14B21B22
-  -  -  -  -  -  -
```

You may notice that the last picture shows two "things" outside the telephone box. These are not shown as being in the "LENGTH AT END" of the queue because they have not entered the queue yet, but have merely reached the top of the path. They do not enter the queue until the next "tick".

```
    COMMENT EXAMPLE 1: SIMULATION OF TELEPHONE BOX;

RECORD THING(LOGICAL MALE; INTEGER START);
REFERENCE(QUEUE)Q;
REFERENCE(FACILITY)TELEPHONE;
REFERENCE(TABLE)T;
REFERENCE(BELT)B,B1;
INTEGER R1,R2,R3;
INTEGER LEAVING,ENTERING,QUEUEING,FINISHING;
LEAVING:=2;  ENTERING:=4;  QUEUEING:=1;  FINISHING:=3;
R1:=45631;
R2:=564732;
R3:=7632486;
INITIALISE;
ZERO:=FALSE;
SETUPQ("QUEUE FOR TELEPHONE",Q);
SETUPBELT (B,4,5);
SETUPBELT(B1,2,3);
SETUPF("TELEPHONE",TELEPHONE,1,0);
SETUPTABLE ("TOTAL TIME SPENT",T);
WHILE CLOCK<361 DO
BEGIN
  IF AT(LEAVING) THEN TAKE(B1);
  IF AGAIN(ENTERING,EXPONENTIAL(5,R3)) THEN PUT(B,1);
  IF AT(QUEUEING) THEN
  BEGIN
    THETHING:=THING(PERHAPS(0.5,R1),CLOCK);
    ENTER(Q)
  END;
  IF AT(FINISHING) THEN
  BEGIN
    RELEASE(TELEPHONE);
    PUT(B1,LEAVING);
    COLLECT(T,CLOCK-START(THETHING))
  END;
  IF ¬EMPTY(Q) AND IDLE(TELEPHONE,0) THEN
  BEGIN
    TAKE(B);
    NEXT(Q);
    SEIZE(TELEPHONE,0);
    NOTE(FINISHING,UNIFORM(1,IF MALE(THETHING) THEN 3 ELSE 9,R2))
  END;
```

```
0505 --          TICK
0505 -3        END;
0506 --        OUTSTATS
0506 -2      END
0506 -1      END.

EXECUTION OPTIONS: DEBUG,0 TIME=97369 PAGES=32767
```

```
#### 1 ######  #### 2 ######  #### 3 ######  #### 4 ######  #### 5 ######  #### 6 ######  #### 7 ######
#############  #############  #############  #############  #############  #############  #############
##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##
## ___| |_ ##  ## ___| |_ ##  ## ___| |_ ##  ## ___| |_ ##  ## ___| |_ ##  ## ___| |_ ##  ## ___|*|_ ##
##         ##  ##         ##  ##  *      ##  ##   *     ##  ##    *    ##  ##     *   ##  ##  *      ##
## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##
##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##
#############  #############  #############  #############  #############  #############  #############
#############  #############  #############  #############  #############  #############  #############

#### 8 ######  #### 9 ######  #### 10 #####  #### 11 #####  #### 12 #####  #### 13 #####  #### 14 #####
#############  #############  #############  #############  #############  #############  #############
##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##
## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##
##  **     ##  ##   **    ##  ##    **   ##  ##    **   ##  ##  * **   ##  ##   ***   ##  ##   ***   ##
## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##
##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##
#############  #############  #############  #############  #############  #############  #############
#############  #############  #############  #############  #############  #############  #############

#### 15 #####  #### 16 #####  #### 17 #####  #### 18 #####  #### 19 #####  #### 20 #####  #### 21 #####
#############  #############  #############  #############  #############  #############  #############
##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##
## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##
##    **>  ##  ##    ** > ##  ##    **   ##  ##  *   *> ##  ##  **  >> ##  ##  ***  > ##  ##   ***   ##
## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##
##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##
#############  #############  #############  #############  #############  #############  #############
#############  #############  #############  #############  #############  #############  #############

#### 22 #####  #### 23 #####  #### 24 #####  #### 25 #####  #### 26 #####  #### 27 #####  #### 28 #####
#############  #############  #############  #############  #############  #############  #############
##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##
## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##
##   ***   ##  ##   ***   ##  ##   ***   ##  ##   ***   ##  ##   ***   ##  ##    **>  ##  ##    ** > ##
## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##
##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##
#############  #############  #############  #############  #############  #############  #############
#############  #############  #############  #############  #############  #############  #############
```

```
#### 29 #####   #### 30 #####   #### 31 #####   #### 32 #####   #### 33 #####   #### 34 #####   #### 35 #####
#############   #############   #############   #############   #############   #############   #############
##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##
## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##
##     *>  ##   ##     * > ##   ##  *  *   ##   ##   * *   ##   ##    **   ##   ##    **   ##   ##    **   ##
## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##
##         ##   ##         ##   ##         ##   ##         ##   ##         ##   ##         ##   ##         ##
#############   #############   #############   #############   #############   #############   #############
#############   #############   #############   #############   #############   #############   #############

#### 36 #####   #### 37 #####   #### 38 #####   #### 39 #####   #### 40 #####   #### 41 #####   #### 42 #####
#############   #############   #############   #############   #############   #############   #############
##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##
## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##
##    **   ##   ##  *  *>  ##   ##   * * > ##   ##    **   ##   ##  * **   ##   ##   ***   ##   ##   ***   ##
## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##
##         ##   ##         ##   ##         ##   ##         ##   ##         ##   ##         ##   ##         ##
#############   #############   #############   #############   #############   #############   #############
#############   #############   #############   #############   #############   #############   #############

#### 43 #####   #### 44 #####   #### 45 #####   #### 46 #####   #### 47 #####   #### 48 #####   #### 49 #####
#############   #############   #############   #############   #############   #############   #############
##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##
## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##
##   ***   ##   ##    **>  ##   ##  * ** > ##   ##   ***   ##   ##  ****   ##   ##  ****   ##   ##  ****   ##
## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##
##         ##   ##         ##   ##         ##   ##         ##   ##         ##   ##         ##   ##         ##
#############   #############   #############   #############   #############   #############   #############
#############   #############   #############   #############   #############   #############   #############

#### 50 #####   #### 51 #####   #### 52 #####   #### 53 #####   #### 54 #####   #### 55 #####   #### 56 #####
#############   #############   #############   #############   #############   #############   #############
##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##   ##    ___  ##
## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##   ## ___|*|_ ##
##  ****   ##   ##   ***>  ##   ##   *** > ##   ##    **>  ##   ##     *>> ##   ##  *  * > ##   ##  ** *   ##
## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##   ## ------- ##
##         ##   ##         ##   ##         ##   ##         ##   ##         ##   ##         ##   ##         ##
#############   #############   #############   #############   #############   #############   #############
#############   #############   #############   #############   #############   #############   #############
```

```
#### 337 ####  #### 338 ####  #### 339 ####  #### 340 ####  #### 341 ####  #### 342 ####  #### 343 ####
#############  #############  #############  #############  #############  #############  #############
##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##
## ___| |_ ##  ## ___| |_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##
##  * *  > ##  ##   * *   ##  ##    *    ##  ##     *   ##  ##     *   ##  ##     *   ##  ##     *   ##
## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##
##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##
#############  #############  #############  #############  #############  #############  #############
#############  #############  #############  #############  #############  #############  #############

#### 344 ####  #### 345 ####  #### 346 ####  #### 347 ####  #### 348 ####  #### 349 ####  #### 350 ####
#############  #############  #############  #############  #############  #############  #############
##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##
## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##
##     *   ##  ##  *  *   ##  ##   *  >  ##  ##    *  > ##  ##     *   ##  ##      >  ##  ##       > ##
## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##
##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##
#############  #############  #############  #############  #############  #############  #############
#############  #############  #############  #############  #############  #############  #############

#### 351 ####  #### 352 ####  #### 353 ####  #### 354 ####  #### 355 ####  #### 356 ####  #### 357 ####
#############  #############  #############  #############  #############  #############  #############
##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##
## ___|*|_ ##  ## ___| |_ ##  ## ___| |_ ##  ## ___| |_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##
##  *      ##  ##   *  >  ##  ##  * *  > ##  ##   * *   ##  ##    *    ##  ##     *   ##  ##     *   ##
## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##
##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##  ##         ##
#############  #############  #############  #############  #############  #############  #############
#############  #############  #############  #############  #############  #############  #############

#### 358 ####  #### 359 ####  #### 360 ####  #### 361 ####  #### 362 ####
#############  #############  #############  #############  #############
##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##  ##    ___  ##
## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##  ## ___|*|_ ##
##     *   ##  ##  *  *   ##  ##  ** *   ##  ##   ** >  ##  ##    ** > ##
## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##  ## ------- ##
##         ##  ##         ##  ##         ##  ##         ##  ##         ##
#############  #############  #############  #############  #############
#############  #############  #############  #############  #############
```

```
 THE VALUE OF THE CLOCK AT THE END OF THE SIMULATION IS  362

                                   QUEUE STATS

                         AV. LENGTH  AV. WAIT TIME   MAX       USERS   LENGTH AT END
QUEUE FOR TELEPHONE        0.59          3.29          5         65          0


                          FACILITY STATS

                           USERS     AVERAGE USE  AV.TIME SPENT  USERS AT END
TELEPHONE                   64          0.66          3.73          1


                                TABLES

                          NUMBER       MAX          MIN        AVERAGE STANDARD DEVIATION
TOTAL TIME SPENT            64          20            1          7.02          4.07

064.22 SECONDS IN EXECUTION
/&                                                                                 09.01.17




********************     ST.ANDREWS UNIVERSITY COMPUTING LABORATORY - 44MFT/SPOOLER END-OF-JOB RECORD     *********
******************** JOB NAME : GHSIM       DATE : 75276       START TIME : 08.58.54   ELAPSED : 00.02.24 *********
********************     CPU TIME : 00.01.23     WAIT TIME : 00.00.24      SYSTEM OVERHEAD : 00.00.21     *********
********************     PAGES PRINTED :     42     CARDS PUNCHED :      0     CARDS READ :    815        *********
********************     A GRAND TOTAL OF          3126  INPUT/OUTPUT REQUESTS WERE HANDLED FOR THE JOB.  *********
********************     THE AVAILABLE CORE WAS : 200 K      OF WHICH AT LEAST    75 K WAS USED          *********
********************                    THE APPROXIMATE COST OF THIS JOB WAS  £  1.19                    *********
********************                 THIS JOB WAS EXECUTED IN THE BACKGROUND PARTITION                   *********
* THIS LINE PRINTED AT 09.07.03  ;
```

CHAPTER 5

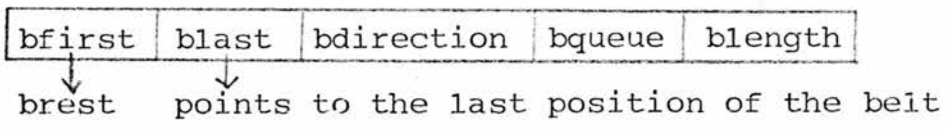
THE IMPLEMENTATION OF THESE REVISIONS

This chapter demonstrates how the additional housekeeping is controlled.

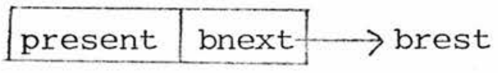
5.1 Belts

```
record belt (reference(brest)bfirst,blast; integer bdirection,
     bqueue,blength);
record brest (logical present; reference(brest)bnext);
reference (belt) SUPPLIED BY USER;
```

BELT



BREST



Symbol	Meaning
bfirst	a pointer to the first place on the belt.
blast	a pointer to the last place on the belt.
bdirection	the direction that the belt is going.
bqueue	the number of "things" stacked on the front of the belt.
blength	the length of the belt.
present	a logical which indicates whether a particular place on the belt is occupied by a "thing".
bnext	a pointer to the next place on the belt.

5.2 Signals

<u>record</u> signal (<u>logical</u> green);

<u>reference</u> (signal) SUPPLIED BY USER;

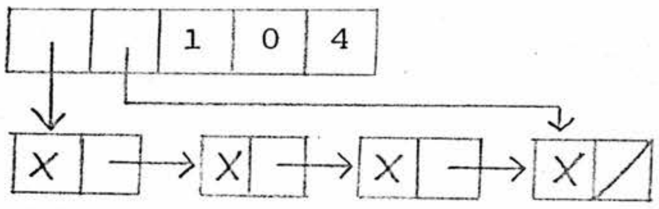
<u>Symbol</u>	<u>Meaning</u>
green	a logical which is <u>true</u> if the signal is green and <u>false</u> if it is red.

Although signals are very small and are relevant particularly to traffic systems, I have included then in the package so that it is possible to get their condition shown on the pictures.

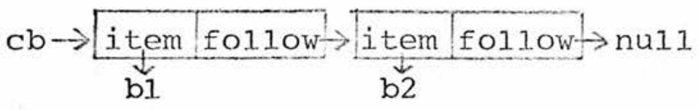
All the variables used by belts and signals <u>cannot</u> be defined by the user in the main block of his program as well as those mentioned previously.

5.3 How the belts work

When the "setupbelt" procedure is called, a belt is set up as a group of records with a header of type "belt" followed by a number of records of type "brest" depending on the length of the belt. Thus 'setupbelt(B,4,1);' would create the following group of records:



It would also set up the appropriate pointers in the "chain" as follows:



This is used during the printing of the pictures when a pointer works down the chain the number of times corresponding to the number in the second position of the square. Another pointer then works its way down the belts the corresponding number of times depending on the third number of the three of each square.

When the procedure "put" is called, the value of "bqueue" of that particular belt is incremented by 1 and a "note" is made in the diary that a "thing" will be leaving the belt in a time depending on the length of the belt.

Each time "tick" is called, all "things" in the belt are moved along by one place in the belt. If the last position of the belt is full, then this means that something was not taken from the belt during the last tick and so other "things" on the belt will queue up behind it and move as close as they can without actually filling in the same square. After this has been done, the picture is printed.

5.4 How the pictures are printed

Pictures are stored until either the end of the simulation is reached, or a line of pictures is filled, whichever is the sooner. Each picture is seperated by 6 columns and 4 lines and so if the picture width was 25 then 4 pictures could be fitted on one width of output paper of size width 120. If the height of the picture was 10, then 3 pictures could be fitted downwards on a page of length 44, thus 12 pictures can be printed on a page of size 25 by 10 each.

At each "tick" of the clock, each member of the '<u>string</u>(3) <u>array</u> pic' are scanned and if the first character of the 3-character string is Q,B,S or F, then a further investigation evaluates the appropriate symbol to place in the '<u>string</u>(1) <u>array</u> picout' used for the output. Any other symbol is copied straight into the output array and the second and third columns are ignored.

CHAPTER 6

IMPROVEMENTS WHICH COULD HAVE BEEN MADE

As has been shown in this thesis, it is possible to write a simulation package which can use all the advantages of a high level language and at the same time give a reasonable performance in simulating various systems. Obviously there are always improvements which can be made to any package which the designer may not realise until after the package has been written or which are impossible to implement due to machine capacity and so on.

One particular defect with this package is the inability to be able to converse with the system as one is able to do on a multi-access system. Instead of running a program 6 times, each time changing a card, it would be much easier merely to edit a file on the computer and then rerun the program and discover straightaway what difference this change has made. In particular, with regard to interactions with the computer, I would have liked very much to have been able actually to change various entities of the program during the execution such as traffic lights on a cross-roads.

Another improvement which could have been made concerns the actual language in which the package is written. During my period of writing this package, I have discovered a little about ALGOL-68 and how it is possible to store procedures in records[17]. Thus by using this facility, it would be possible to store procedures in the diary with the "note" procedure and then "tick" would execute these procedures when it was time for that activity to be done. This would then eliminate the need for event codes and the procedure "at".

Another change which could be made, although not necessarily an improvement, would be to change "clock" from integer to real. Since the clock jumps ahead each time "tick" is called to the time when the next event is to occur, this time would just as well be a fraction of a tick. In some cases this is more realistic since in real life things do not happen in exact units of time, but at any time. However, with an integer clock, the simulation is more machine independant and there is more precise control over the time.

At the start of each simulation run, the system is completely empty with none of its facilities busy. In actual life we want to know the results, such as the average time some "thing" takes to go through the shop, when it is at its normal workload and so one may not want information of this sort to be collected until the simulation has been going a while so that the initial case does not produce peculiar results. To produce these results a procedure resetstats; could be introduced which sets all the statistics back to their original values and the clock back to 0, but keeping the values of "things" actually in the system unchanged.

It should also be possible to restrict the output of pictures, so that only at certain times pictures would be output instead of at each tick. Thus it would be possible to skip a few ticks in the simulation and see what is happening, say 50 ticks later. This would not only save paper, but also enable longer simulations to be run with pictures.

The diary could be organised as a leftist tree[13] which is a special type of binary tree structure which allows easy implementation of priority queues. Although this method makes "note" easier and quicker, it makes "at" and "again" more complicated. This is because for each "time" in the diary are associated a number of "events". Thus if something is to be removed from the diary, this may not necessarily be the node element, but another element with the same "time" value. If this method was to be used, there would have to be some way of ordering the cycle of events such that the required event was always at the node of the tree.

Some of the identifiers mentioned in Appendix 1 which the user would not normally use should have had more outlandish names so that the user does not accidently use the same name for some of his own identifiers.

Vavious other examples with and without pictures are given in Appendices 2 and 3 which follow.

APPENDIX 1

A LIST OF IDENTIFIERS USED IN THE PACKAGE

This list does not include a list of ALGOL-W reserve words which can be found elsewhere[7]. If the user wishes to use any of these identifiers he must include an additional *begin* *end* block before declaring them.

Again	Facility	Perhaps	Release
At	*Fd	*Pic	Rest
*Bdirection	Fhead	*Pich	Seize
*Belt	Flast	*Pichs	*Setupbelt
*Bfirst	Flength	*Picout	Setupf
*Blast	Fmake	*Picw	Setupq
*Blength	Fmax	*Picws	*Setupsignal
*Bnext	Fnumber	Poisson	Setuptable
*Bqueue	Follow	*Present	*Signal
*Brest	Ftime	*Put	Table
Busy	*Full	Qback	*Take
*Cb	*Green	Qhead	+Tempstats
Cf	*Height	Qlast	Thead
Chain	*Hheight	Qlength	Thething
*Change	Idle	Qmake	Thing
Clock	Initialise	Qmax	Tick
Collect	Item	Qn	Time
Cq	Jump	Qnext	Tmax
*Cs	Kind	Qnumber	Tmin
Ct	Next	Qrest	Tnumber
Diary	Normal	Qtime	Ttotal
Empty	Note	Queue	Ttotalsq
Enter	Outstats	Random	Uniform
Event	*Pageh	Ready	*Width
Eventlist	*Pagew	*Red	Zero
Exponential			

*only included in the package involving drawings.

+only included in the package not involving drawings.

APPENDIX 2

EXAMPLES WITHOUT PICTURES

Example 1

A supermarket manager is trying to decide whether he should change the number of checkouts that he has in his supermarket at the present time. He also wishes to make sure that he has enough baskets and trolleys for his customers to use.

People who use the supermarket are divided into various classes depending upon who they are. 30% of the customers are old people, 30% of the customers take a trolley, 50% take a basket, 5% take both and the rest don't take anything.

Customers enter the shop according to a Poisson distribution with a mean time of 1 minute. The average time people take to do their shopping is 8 minutes normally distributed with a standard deviation of 3 minutes. If the customer is old or has a trolley, then he takes on average 12 minutes to be served at the checkout otherwise this number is 5½ minutes. Both are distributed according to a Uniform distribution with a minimum of 1 minute. Sometime between every 10 minutes and 3 hours one of the store's VIPs reaches the checkout and he jumps to the head of the queue provided that he has a basket.

How many servers should the supermarket have and what is the minimum number of baskets and trolleys required so as not to run out of them?

Each tick represents 1 minute.

By having a more than sufficient number of baskets and trolleys initially, it was possible to discover the number of servers required to cope with the people so that the queue at the checkout does not gradually get longer. It can be seen from Table 1 that 13 servers is the minimum number of servers required to cope with the customers.

Using this value, the program was then rerun for different numbers of baskets and trolleys. From Table 2 it is possible to see that 19 baskets and 16 trolleys was the minimum number that prevented a shortage.

This particular combination of baskets, trolleys and servers is used in the program output following the tables.

The % utilisation of the facilities was obtained by dividing the utilisation provided by the output statistics by the number of facilities present in that simulation. The value for baskets and trolleys may appear unusually low but this is because at the start of the simulation when the shop opens, all baskets and trolleys are "idle" and only become "busy" as customers enter the shop.

<u>TABLE 1 - To discover the minimum number of servers required</u>

	servers	customers served	average time spent in shop	utilisation of servers		queue at checkout average length	max length	length at end
run 1	11	272	21.31	8.68	79%	4.37	25	13
run 2	11	279	17.71	8.28	75%	0.70	11	0
run 3	11	305	20.24	9.47	86%	3.45	17	4
average	11	285	19.75	8.81	80%	2.84	18	6
run 1	12	281	19.98	8.98	75%	2.64	19	3
run 2	12	279	17.36	8.29	69%	0.37	9	0
run 3	12	305	18.13	9.51	79%	1.25	11	3
average	12	288	18.49	8.93	74%	1.42	13	2
run 1	13	286	18.79	9.14	70%	1.25	13	0
run 2	13	279	17.16	8.29	64%	0.18	8	0
run 3	13	306	17.51	9.56	74%	0.58	10	1
average	13	290	17.82	9.00	69%	0.67	10	0

With more servers, the number of customers served was the same as for 13 servers, as was the utilisation (not the % value) of servers, thus all customers had been served. These simulations were run with 25 baskets and 20 trolleys. The same set of random numbers were used for each set of results.

TABLE 2 - To discover the minimum number of baskets and trolleys required

	baskets	trolleys	utilisation of baskets		utilisation of trolleys		short
run 1	17	14	9.01	53%	7.60	54%	2 trolleys 1 basket
run 2	17	14	8.72	51%	6.12	44%	
run 3	17	14	8.03	47%	6.48	46%	2 baskets 2 trolleys
average	17	14	8.59	51%	6.73	48%	
run 1	18	15	9.18	51%	8.09	54%	1 trolley
run 2	18	15	8.72	48%	6.12	41%	
run 3	18	15	8.92	50%	6.92	46%	1 basket 1 trolley
average	18	15	8.94	50%	7.04	47%	
run 1	19	16	9.18	48%	8.38	52%	
run 2	19	16	8.72	46%	6.12	38%	
run 3	19	16	9.79	51%	7.34	46%	
average	19	16	9.23	49%	7.28	45%	

With more baskets and trolleys, the utilisation of them (not the % value) remained the same, thus the required number of baskets and trolleys have been provided. These simulations were run with 13 servers at the checkouts. The same set of random numbers were used for each set of results.

```
0310 --          COMMENT SIMULATION OF SUPERMARKET;
0310 --
0310 --          RECORD THING(LOGICAL BASK,TROLL,MALE,OLD; INTEGER START);
0312 --          REFERENCE (QUEUE)Q;
0313 --          REFERENCE (FACILITY) BASKET,TROLLEY,SHOPPING,SERVER;
0314 --          REFERENCE (TABLE) T;
0315 --          INTEGER R1,R2,R3,R4,NB,NT,NS;
0316 --          INTEGER ENTERING,QUEUEING,FINISHING,VIP;
0317 --          ENTERING:=1;  QUEUEING:=2;  FINISHING:=3;  VIP:=4;
0321 --          READ(R1,R2,R3,R4);
0322 --          WHILE R1¬=0 DO
0322 2-          BEGIN
0323 --            READ(NB,NT,NS);
0324 --            IOCONTROL(3);
0325 --            WRITE("SIMULATION OF SUPERMARKET WITH",NB,"BASKETS,",NT,"TROLLEYS AN
0325 --        D",NS,"SERVERS");
0326 --            WRITE ("% THE RANDOM NUMBERS ARE ",R1,R2,R3,R4);
0327 --            INITIALISE;
0328 --            SETUPQ ("QUEUE AT CHECKOUT",Q);
0329 --            SETUPF ("BASKETS",BASKET,NB);
0330 --            SETUPF("TROLLEYS",TROLLEY,NT);
0331 --            SETUPF("SHOP",SHOPPING,500);
0332 --            SETUPF("CHECKOUT SERVER",SERVER,NS);
0333 --            SETUPTABLE ("TOTAL TIME IN SHOP",T);
0334 --            WHILE CLOCK < 300 DO
0334 3-            BEGIN
0335 --              WHILE AT(FINISHING) DO COMMENT FINISHED AT CHECKOUT;
0335 4-              BEGIN
0336 --                RELEASE(SERVER);
0337 --                COLLECT(T,CLOCK-START(THETHING));
0338 --                IF BASK(THETHING) THEN RELEASE(BASKET);
0339 --                IF TROLL(THETHING) THEN RELEASE(TROLLEY)
0339 -4              END;
0340 --              WHILE AGAIN(ENTERING,POISSON(1,R1)) DO
0340 4-              BEGIN COMMENT ENTER ANOTHER PERSON IN THE SYSTEM;
0341 --                IF PERHAPS (0.3,R3) THEN
0341 5-                BEGIN
0342 --                  THETHING:=THING(FALSE,TRUE,FALSE,FALSE,CLOCK);
0343 --                  SEIZE(TROLLEY)
0343 -5                END ELSE
0343 --                IF PERHAPS (5/7,R3) THEN
0343 5-                BEGIN
```

```
                 THETHING:=THING(TRUE,FALSE,FALSE,FALSE,CLOCK);
                 SEIZE(BASKET)
               END ELSE
               IF PERHAPS (0.25,R3) THEN
               BEGIN
                 THETHING:=THING(TRUE,TRUE,FALSE,FALSE,CLOCK);
                 SEIZE(BASKET);
                 SEIZE(TROLLEY)
               END ELSE
               THETHING:=THING(FALSE,FALSE,FALSE,FALSE,CLOCK);
               IF PERHAPS(0.5,R3) THEN MALE(THETHING):=TRUE;
               IF PERHAPS(0.3,R3) THEN OLD(THETHING):=TRUE;
               SEIZE(SHOPPING);
               NOTE(QUEUEING,NORMAL(8,3,R2))
             END;
             WHILE AT(QUEUEING) DO
             BEGIN COMMENT FINISHED SHOPPING SO QUEUE FOR CHECKOUT;
               RELEASE(SHOPPING);
               ENTER(Q)
             END;
             WHILE ¬BUSY(SERVER) AND ¬EMPTY(Q) DO COMMENT SERVE CUSTOMER;
             BEGIN
               NEXT(Q);
               SEIZE(SERVER);
               IF OLD(THETHING) OR TROLL(THETHING) THEN
               NOTE(FINISHING,UNIFORM(4,20,R4)) ELSE NOTE(3,UNIFORM(1,10,R4))
             END;
             IF AGAIN(VIP,UNIFORM(10,180,R1)) AND JUMP(Q,BASK(THETHING)) THEN
             WRITE("JUMPED OK AT CLOCK TIME = ",CLOCK);
             TICK
           END;
           OUTSTATS;
           READ(R1,R2,R3,R4)
         END
       END.

EXECUTION OPTIONS: DEBUG,0 TIME=97369 PAGES=32767
```

```
SIMULATION OF SUPERMARKET WITH      19   BASKETS,       18   TROLLEYS AND      13   SERVERS
 THE RANDOM NUMBERS ARE       873   65432170        6382   893046732
JUMPED OK AT CLOCK TIME =    160
 THE VALUE OF THE CLOCK AT THE END OF THE SIMULATION IS    300


                                  QUEUE STATS

                        AV. LENGTH  AV. WAIT TIME    MAX        USERS   LENGTH AT END
QUEUE AT CHECKOUT          0.58          0.54         10          319          1


                         FACILITY STATS

                          USERS      AVERAGE USE  AV.TIME SPENT  USERS AT END
BASKETS                   174          9.79         16.82          10
TROLLEYS                  103          7.34         21.17           7
SHOP                      320          8.58          8.02           8
CHECKOUT SERVER           306          9.56          9.34          13


                               TABLES

                         NUMBER      MAX         MIN        AVERAGE STANDARD DEVIATICN
TOTAL TIME IN SHOP        306         39           4         17.51         6.09

046.80 SECONDS IN EXECUTION
/&                                                                                 21.20.34




********************    ST.ANDREWS UNIVERSITY COMPUTING LABORATORY - 44MFT/SPOOLER END-OF-JOB RECCRD    **********
******************** JOB NAME : GHSIM      DATE : 75276       START TIME : 21.19.02   ELAPSED : 00.01.33 **********
********************    CPU TIME : 00.01.00     WAIT TIME : 00.00.19     SYSTEM CVERHEAD : 00.00.09     **********
********************    PAGES PRINTED :    24    CARDS PUNCHED :     0     CARDS READ :    594          **********
********************    A GRAND TOTAL OF        2064  INPUT/OUTPUT REQUESTS WERE HANDLED FOR THE JOB.   **********
********************    THE AVAILABLE CORE WAS : 112 K     OF WHICH AT LEAST   68 K WAS USED            **********
********************                THE APPROXIMATE COST OF THIS JOB WAS  £  0.82                       **********
********************             THIS JOB WAS EXECUTED IN THE BACKGROUND PARTITION                      **********
* THIS LINE PRINTED AT 21.22.08  ;
```

Example 2

An import company operating its own fleet of ships considers the construction of its port facilities. It has to be decided whether one, two or three berths are to be built and an unloading installation has to be selected from three available types, A, B and C. If more than one berth is built, then all will be equipped with the same type of installation.

The installations are characterised by the following parameters:

Type of installation	Average unloading capacity ton/day	Fixed cost per day £ units	Operating cost per day £ units
A	4000	500	500
B	6000	800	750
C	8000	1000	900

Operating costs are incurred only during the time of actual use of the installation while the fixed cost applies to all days. Ships carry 10000 tons each and their arrival can be considered as Poisson with a mean arrival rate of one ship per day. Service - unloading - time can be considered exponential with mean equal to the given average unloading capacity.

The time spent by the ship in the port (waiting and unloading time) costs approximately £1000 per ship per day. How many berths should be constructed and what type of installation chosen to minimise overall cost?

Since the following cases are not feasible: A1, A2, B1 and C1, since they cannot cope with the number of ships arriving and will make the queue longer all the time, we need only simulate this example with the cases A3, B2, B3, C2 and C3.

When this problem was evaluated theoretically, similar results were obtained.

Each tick is equivalent to $\frac{1}{10}$th of a day (2.4 hours).

Event 'entering' is a ship arriving at the port and event 'leaving' represents a ship leaving the port having been unloaded.

The table below shows the results obtained. For each type of berth, 3 simulations were run with different random numbers and their average results used.

The output included after the listing is one of the solutions.

The results shown below all represent the average cost per day.

Type of installation	Run 1	Run 2	Run 3	Average
A3	£719	£615	£666	£667
B2	£681	£578	£628	£629
B3	£539	£539	£548	£542
C2	£450	£454	£456	£453
C3	£529	£534	£539	£534

The results shown on the output sheet should be interpreted as follows:

The fixed cost is for all the berths and so this value will be the fixed cost shown in the table on the previous page multiplied by 2 or 3 depending on the number of berths.

Both fixed cost and operating cost are shown for each time interval ($\frac{1}{10}$th day) and so are $\frac{1}{10}$th of the values shown in the first table.

To obtain the average cost per day divide the total cost by the time at the end of the simulation.

Thus two installations of type C should be built for minimum cost.

```
COMMENT SIMULATION OF DOCKS;

RECORD THING(INTEGER COST,TIM);
REFERENCE(QUEUE)Q;
REFERENCE(FACILITY)BERTH;
REFERENCE(TABLE)T,T1;
INTEGER R1,R2,NB,FC,OC;
REAL UT;
INTEGER ENTERING,LEAVING;
ENTERING:=1; LEAVING:=2;
READ(R1,R2);
WHILE R1¬=0 DO
BEGIN
IF R2 =0 THEN
BEGIN
  READ(NB,FC,OC,UT);
  READ(R1,R2)
END;
ICCONTROL(3);
WRITE ("DOCKS WITH",NB,"BERTHS WITH FIXED COST",FC,",OPERATING COST",
OC);
WRITE(" AND AVERAGE UNLOADING TIME OF",UT);
WRITE(" THE RANDOM NUMBERS ARE",R1,R2);
INITIALISE;
ZERO:=FALSE;
SETUPQ("WAITING",Q);
SETUPF("BERTH",BERTH,NB);
SETUPTABLE("SHIP COSTS",T);
SETUPTABLE ("OPERATING COSTS",T1);
WHILE CLOCK <= 10000 DO
BEGIN
  IF AGAIN(ENTERING,POISSON(10,R1)) THEN
  BEGIN
    THETHING:=THING(0,CLOCK);
    ENTER(Q)
  END;
  WHILE AT(LEAVING) DO
  BEGIN
    COLLECT (T1,CLOCK-COST(THETHING));
    COLLECT (T,CLOCK-TIM(THETHING));
    RELEASE(BERTH)
  END;
```

```
0342 --          WHILE ¬EMPTY(Q) AND ¬BUSY(BERTH) DO
0342 4-          BEGIN
0343 --            NEXT(Q);
0344 --            COST(THETHING):=CLOCK;
0345 --            SEIZE(BERTH);
0346 --            NCTE(LEAVING,EXPONENTIAL(UT,R2))
0346 -4          END;
0347 --          TICK
0347 -3        END;
0348 --        OUTSTATS;
0349 --        WRITE("TOTAL COSTS ARE £",CLOCK*FC+TTOTAL(T)*100+TTOTAL(T1)*OC);
0350 --        READ(R1,R2)
0350 -2      END
0350 -1      END.

EXECUTION OPTIONS: DEBUG,0 TIME=97369 PAGES=32767
```

```
DOCKS WITH          2  BERTHS WITH FIXED COST          200  OPERATING COST          90
 AND AVERAGE UNLOADING TIME OF       12.00000
 THE RANDCM NUMBERS ARE        4376129             1385
 THE VALUE OF THE CLOCK AT THE END CF THE SIMULATION IS 10002


                                  QUEUE STATS

                         AV. LENGTH  AV. WAIT TIME   MAX       USERS   LENGTH AT END
WAITING                     0.22         2.26          6        993           0

                          FACILITY STATS

                           USERS    AVERAGE USE  AV.TIME SPENT  USERS AT END
BERTH                      993        1.20         12.07           0

                               TABLES

                          NUMBER       MAX         MIN        AVERAGE STANDARD DEVIATICN
SHIP COSTS                 993         70           1          14.33         12.70
OPERATING COSTS            993         70           1          12.07         11.10
TOTAL CCSTS ARE £ 4502430
```

APPENDIX 3

EXAMPLES WITH PICTURES

Example 1

A city council is trying to decide whether a bus lane should be made in a particular road in the city. The stretch of road has 4 bus stops on it. The projected bus lane would be between the first stop and the fourth. A traffic census has been taken and at the busiest time of day when a bus lane would be of most use, vehicles appear approximately every 10 seconds. For the purpose of the simulation, the road is divided into squares which take a car 5 seconds to cross. Each tick represents 5 seconds of real time.

Approximately 15% of the vehicles are buses and of the remaining vehicles 60% of the drivers do not give way to buses. At bus stop 1 the probability of having to stop is 60% when the bus is stopped for 5 ticks. Bus stops 2 and 3 have the probability of 30% of a bus stopping when it will stop for 4 ticks. At bus stop 4, the probability of stopping is 40% when the bus is stopped for 5 ticks.

At the end of the stretch of road is a set of traffic lights which change between red and green every 15 ticks.

The pictures are input in the following format where P represents a bus stop in the picture. With the bus lane:

```
         P                 P                 P                 P
      +  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  +  S1
-  -  +  F2 B21B22B23B24B25F3 B31B32B33B34B35F4 B41B42B43B44B45F5 +  -
B10B11F1 B51B52B53B54B55B56B57B58B59B5AB5BB5CB5DB5EB5FB5GB5HB5IB5JB61B62
-  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -
```

Without the bus lane, B5 is removed and just one lane of the road is shown. Extra vehicles that cannot be shown on the diagram are indicated by B20, B30 and B40 below the road:

```
    P                    P                  P                  P
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
B10B11F1 F2 B21B22B23B24B25F3 B31B32B33B34B35F4 B41B42B43B44B45F5 B51B52
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
            B20                B30                B40
```

As can be seen from the output, both cars and buses move along the stretch of road slower when there is no bus lane.

Only one listing of the program is included and that is the one which includes a bus lane. Only a few pictures of the simulation have been included both with and without a bus lane.

Since there are only 5 squares between each bus stop, when a queue builds up and more than 5 vehicles are occupying the space between bus stops, the extra number of vehicles is denoted by the number below that particular stretch of road it is referring to.

From these simulations, it can be seen that bus lanes are an advantage both to cars and buses, since cars can pass buses at bus stops and buses can pass cars when they are stuck in traffic jams waiting for the car in front to move.

```
0468 --          COMMENT SIMULATION OF BUS LANE;
0468 --
0468 --          RECORD THING(LOGICAL BUS,UNKIND; INTEGER START);
0470 --          REFERENCE(TABLE)TBUS,TCAR;
0471 --          REFERENCE(SIGNAL)S;
0472 --          REFERENCE(QUEUE)Q1,Q2,Q3,Q4,Q5,Q6,Q7;
0473 --          REFERENCE(FACILITY)F1,F2,F3,F4,F5;
0474 --          REFERENCE(BELT)B1,B2,B3,B4,B5,B6;
0475 --          INTEGER R;
0476 --          INTEGER ENTERING,STOP1,STARTBUSLANE,ENTERSTOP2,ENDBUSLANE,STOP2,
0476 --          ENTERSTOP3,STOP3,ENTERSTOP4,STOP4,LIGHTS,CHANGING;
0477 --          ENTERING:=1; STOP1:=2; STARTBUSLANE:=3; ENTERSTOP2:=4; ENDBUSLANE:=5;
0482 --          STOP2:=6; ENTERSTOP3:=7; STOP3:=8; ENTERSTOP4:=9; STOP4:=10;
0487 --          LIGHTS:=11; CHANGING:=12;
0489 --          R:=432567201;
0490 --          INITIALISE;
0491 --          SETUPQ("ENTER BUS LANE ",Q1);
0492 --          SETUPQ("ENTER BUS STOP 2",Q2);
0493 --          SETUPQ("ENTER BUS STOP 3",Q3);
0494 --          SETUPQ("ENTER BUS STOP 4",Q4);
0495 --          SETUPQ("LEAVING BUS LANE",Q5);
0496 --          SETUPQ("CAR QUEUE",Q6);
0497 --          SETUPQ("TRAFFIC LIGHTS",Q7);
0498 --          SETUPF(" START OF BUS LANE",F1,1,0);
0499 --          SETUPF("BUS STOP 1",F2,1,3);
0500 --          SETUPF("BUS STOP 2",F3,1,3);
0501 --          SETUPF("BUS STOP 3",F4,1,3);
0502 --          SETUPF("BUS STOP 4",F5,1,2);
0503 --          SETUPBELT(B1,1,3);
0504 --          SETUPBELT(B2,5,3);
0505 --          SETUPBELT(B3,5,3);
0506 --          SETUPBELT(B4,5,3);
0507 --          SETUPBELT(B5,19,3);
0508 --          SETUPBELT(B6,2,3);
0509 --          SETUPTABLE("TOTAL TIME BUS TAKES",TBUS);
0510 --          SETUPTABLE("TOTAL TIME CAR TAKES",TCAR);
0511 --          SETUPSIGNAL(S,TRUE);
0512 --          WHILE CLOCK < 400 DO
0512 3-          BEGIN
0513 --            IF AT(LIGHTS) THEN ENTER(Q7);
0514 --            IF ¬EMPTY(Q7) AND GREEN(S) THEN
0514 4-            BEGIN
```

```
0515 --            TAKE(B6);
0516 --            NEXT(Q7);
0517 --           COLLECT(IF BUS(THETHING) THEN TBUS ELSE TCAR,CLOCK-START(THETHING))
0517 -4          END;
0518 --          IF AT(ENDBUSLANE) THEN ENTER(Q6);
0519 --          IF ¬EMPTY(Q6) AND ¬FULL(B6) AND (EMPTY(Q5) OR UNKIND(QMAKE(QNEXT(Q6)
0519 --          ))) THEN
0519 4-          BEGIN
0520 --            TAKE(B5);
0521 --            NEXT(Q6);
0522 --            PUT(B6,LIGHTS)
0522 -4          END;
0523 --          IF BUSY(F1,3) AND ¬FULL(B5) THEN
0523 4-          BEGIN
0524 --            RELEASE(F1);
0525 --            PUT(B5,ENDBUSLANE)
0525 -4          END;
0526 --          IF AT(STOP4) THEN
0526 4-          BEGIN
0527 --            RELEASE(F5);
0528 --            SEIZE(F5,2);
0529 --            ENTER(Q5)
0529 -4          END;
0530 --          IF ¬EMPTY(Q5) AND ¬FULL(B6) THEN
0530 4-          BEGIN
0531 --            NEXT(Q5);
0532 --            RELEASE(F5);
0533 --            PUT(B6,LIGHTS)
0533 -4          END;
0534 --          IF AT(ENTERSTOP4) THEN ENTER(Q4);
0535 --          IF ¬EMPTY(Q4) AND IDLE(F5,0) THEN
0535 4-          BEGIN
0536 --            TAKE(B4);
0537 --            NEXT(Q4);
0538 --            SEIZE(F5,3);
0539 --            IF BUS(THETHING) AND PERHAPS(0.4,R) THEN NOTE(STOP4,5) ELSE
0539 --            NOTE(STOP4,1)
0539 -4          END;
0540 --          IF AT(STOP3) THEN
0540 4-          BEGIN
0541 --            RELEASE(F4);
0542 --            PUT(B4,ENTERSTOP4)
```

```
-4         END;
--         IF AT(ENTERSTOP3) THEN ENTER(Q3);
--         IF ¬EMPTY(Q3) AND IDLE(F4,0) THEN
4-         BEGIN
--           TAKE(B3);
--           NEXT(Q3);
--           SEIZE(F4,0);
--           IF BUS(THETHING) AND PERHAPS(0.3,R) THEN NOTE(STOP3,4) ELSE
--           NOTE(STOP3,1)
-4         END;
--         IF AT(STOP2) THEN
4-         BEGIN
--           RELEASE(F3);
--           PUT(B3,ENTERSTOP3)
-4         END;
--         IF AT(ENTERSTOP2) THEN ENTER(Q2);
--         IF ¬EMPTY(Q2) AND IDLE(F3,0) THEN
4-         BEGIN
--           TAKE(B2);
--           NEXT(Q2);
--           SEIZE(F3,0);
--           IF BUS(THETHING) AND PERHAPS(0.3,R) THEN NOTE(STOP2,4) ELSE
--           NOTE(STOP2,1)
-4         END;
--         IF AT(STOP1) THEN
4-         BEGIN
--           RELEASE(F2);
--           PUT(B2,ENTERSTOP2)
-4         END;
--         IF BUSY(F1,1) AND IDLE(F2,0) THEN
4-         BEGIN
--           RELEASE(F1);
--           SEIZE(F2,0);
--           IF BUS(THETHING) AND PERHAPS(0.6,R) THEN NOTE(STOP1,5) ELSE
--           NOTE(STOP1,1)
-4         END;
--         IF AT(STARTBUSLANE) THEN ENTER(Q1);
--         IF ¬EMPTY(Q1) AND IDLE(F1,0) THEN
4-         BEGIN
--           TAKE(B1);
--           NEXT(Q1);
--           SEIZE(F1,IF BUS(THETHING) THEN 1 ELSE 3)
```

```
            END;
            IF AGAIN(ENTERING,2) THEN
            BEGIN
              IF PERHAPS(0.15,R) THEN THETHING:=THING(TRUE,FALSE,CLOCK)
              ELSE IF PERHAPS(0.4,R) THEN THETHING:=THING(FALSE,FALSE,CLCCK)
              ELSE THETHING:=THING(FALSE,TRUE,CLCCK);
              PUT(B1,STARTBUSLANE)
            END;
            IF AGAIN(CHANGING,15) THEN CHANGE(S);
            TICK
          END;
          CUTSTATS
        END
        END.

EXECUTION OPTIONS: DEBUG,0 TIME=97369 PAGES=32767
```

```
############# 1 ##############  ############# 2 ##############  ############# 3 ##############
##############################  ##############################  ##############################
##    P     P     P     P   ##  ##    P     P     P     P   ##  ##    P     P     P     P   ##
##   +___________________+O ##  ##   +___________________+O ##  ##   +___________________+O ##
## __+                   +_ ##  ## __+                   +_ ##  ## __+                   +_ ##
##                          ##  ##                          ##  ##  >                       ##
## ------------------------ ##  ## ------------------------ ##  ## ------------------------ ##
##############################  ##############################  ##############################
##############################  ##############################  ##############################

############# 4 ##############  ############# 5 ##############  ############# 6 ##############
##############################  ##############################  ##############################
##    P     P     P     P   ##  ##    P     P     P     P   ##  ##    P     P     P     P   ##
##   +___________________+C ##  ##   +___________________+O ##  ##   +___________________+O ##
## __+                   +_ ##  ## __+                   +_ ##  ## __+                   +_ ##
##   >                      ##  ##  > >                     ##  ##   > >                    ##
## ------------------------ ##  ## ------------------------ ##  ## ------------------------ ##
##############################  ##############################  ##############################
##############################  ##############################  ##############################

############# 7 ##############  ############# 8 ##############  ############# 9 ##############
##############################  ##############################  ##############################
##    P     P     P     P   ##  ##    P     P     P     P   ##  ##    P     P     P     P   ##
##   +___________________+C ##  ##   +___________________+O ##  ##   +___________________+O ##
## __+                   +_ ##  ## __+                   +_ ##  ## __+>                  +_ ##
##  > > >                   ##  ##   A > >                  ##  ##  >   > >                 ##
## ------------------------ ##  ## ------------------------ ##  ## ------------------------ ##
##############################  ##############################  ##############################
##############################  ##############################  ##############################

############# 10 #############  ############# 11 #############  ############# 12 #############
##############################  ##############################  ##############################
##    P     P     P     P   ##  ##    P     P     P     P   ##  ##    P     P     P     P   ##
##   +___________________+C ##  ##   +___________________+O ##  ##   +___________________+C ##
## __+ >                 +_ ##  ## __+  >                +_ ##  ## __+   >               +_ ##
##   >   > >                ##  ##  > >   > >               ##  ##   > >   > >              ##
## ------------------------ ##  ## ------------------------ ##  ## ------------------------ ##
##############################  ##############################  ##############################
##############################  ##############################  ##############################
```

```
############ 385 ############   ############ 386 ############   ############ 387 ############
#############################   #############################   #############################
##    P     P     P     P  ##   ##    P     P     P     P  ##   ##    P     P     P     P  ##
##   +__________________+@ ##   ##   +__________________+@ ##   ##   +__________________+@ ##
## __+                  +_ ##   ## __+                  +_ ##   ## __+                  +_ ##
##  > > > > > >>>>>>>>>>>> ##   ##   > > > > >>>>>>>>>>>>> ##   ##  > > > > >>>>>>>>>>>>>> ##
## ----------------------- ##   ## ----------------------- ##   ## ----------------------- ##
#############################   #############################   #############################
#############################   #############################   #############################

############ 388 ############   ############ 389 ############   ############ 390 ############
#############################   #############################   #############################
##    P     P     P     P  ##   ##    P     P     P     P  ##   ##    P     P     P     P  ##
##   +__________________+@ ##   ##   +__________________+@ ##   ##   +__________________+@ ##
## __+                  +_ ##   ## __+>                 +_ ##   ## __+>                 +_ ##
##   A > > >>>>>>>>>>>>>>> ##   ##  >   > >>>>>>>>>>>>>>>> ##   ##   >   >>>>>>>>>>>>>>>>> ##
## ----------------------- ##   ## ----------------------- ##   ## ----------------------- ##
#############################   #############################   #############################
#############################   #############################   #############################

############ 391 ############   ############ 392 ############   ############ 393 ############
#############################   #############################   #############################
##    P     P     P     P  ##   ##    P     P     P     P  ##   ##    P     P     P     P  ##
##   +__________________+O ##   ##   +__________________+O ##   ##   +__________________+C ##
## __+>                 +_ ##   ## __+>                 +_ ##   ## __+>                 +_ ##
##  > >  >>>>>>>>>>>>>>>>> ##   ##   > > >>>>>>>>>>>>>>>>> ##   ##  > > > >>>>>>>>>>>>>>>> ##
## ----------------------- ##   ## ----------------------- ##   ## ----------------------- ##
#############################   #############################   #############################
#############################   #############################   #############################

############ 394 ############   ############ 395 ############   ############ 396 ############
#############################   #############################   #############################
##    P     P     P     P  ##   ##    P     P     P     P  ##   ##    P     P     P     P  ##
##   +__________________+O ##   ##   +__________________+O ##   ##   +__________________+C ##
## __+ >                +_ ##   ## __+  >               +_ ##   ## __+   >              +_ ##
##   > > > >>>>>>>>>>>>>>> ##   ##  > > > > >>>>>>>>>>>>>> ##   ##   A > > > >>>>>>>>>>>>> ##
## ----------------------- ##   ## ----------------------- ##   ## ----------------------- ##
#############################   #############################   #############################
#############################   #############################   #############################
```

```
############ 397 ############   ############ 398 ############   ############ 399 ############
##############################   ##############################   ##############################
##    P     P     P     P   ##   ##    P     P     P     P   ##   ##    P     P     P     P   ##
##   +___________________+0 ##   ##   +___________________+0 ##   ##   +___________________+0 ##
## __+>   >              +_ ##   ## __+ >   >            +_ ##   ## __+  >   >            +_ ##
##  >   > > > >>>>>>>>>>>>> ##   ##   >   > > > >>>>>>>>>>>> ##   ##  > >   > > > >>>>>>>>>>> ##
## ------------------------ ##   ## ------------------------ ##   ## ------------------------ ##
##############################   ##############################   ##############################
##############################   ##############################   ##############################

############ 400 ############
##############################
##    P     P     P     P   ##
##   +___________________+0 ##
## __+   >  >            +_ ##
##   > >   > > > >>>>>>>>>> ##
## ------------------------ ##
##############################
##############################
```

```
THE VALUE OF THE CLOCK AT THE END OF THE SIMULATION IS  400


                                    QUEUE STATS

                          AV. LENGTH  AV. WAIT TIME   MAX        USERS   LENGTH AT END
ENTER BUS LANE               0.02         0.05          2         199          0
ENTER BUS STOP 2             0.01         0.19          1          26          0
ENTER BUS STOP 3             0.02         0.28          1          25          0
ENTER BUS STOP 4             0.23         3.36          3          25          0
LEAVING BUS LANE             0.31         4.64          1          25          0
CAR QUEUE                    2.48         6.21          9         159          5
TRAFFIC LIGHTS               1.32         2.88          3         182          0


                          FACILITY STATS

                           USERS    AVERAGE USE   AV.TIME SPENT  USERS AT END
 START OF BUS LANE         198         0.52           1.05           1
BUS STOP 1                  27         0.23           3.37           0
BUS STOP 2                  25         0.12           1.84           1
BUS STOP 3                  25         0.12           1.72           0
BUS STOP 4                  50         0.44           3.30           0


                               TABLES

                          NUMBER       MAX         MIN        AVERAGE STANDARD DEVIATION
TOTAL TIME BUS TAKES        25          69          27         42.00      11.53
TOTAL TIME CAR TAKES       157          41          23         31.61       5.31

185.52 SECONDS IN EXECUTION
/&                                                                                  10.11.41


*********************    ST.ANDREWS UNIVERSITY COMPUTING LABORATCRY - 44MFT/SPCCLER ENC-CF-JCB RECCRC    **********
********************* JOB NAME : GHSIM      DATE : 75276      START TIME : 10.C7.08   ELAPSEC : CO.04.35 **********
*********************    CPU TIME : 00.03.26     WAIT TIME : 00.00.24     SYSTEM OVERHEAD : 00.CC.26     **********
*********************    PAGES PRINTED :    65     CARDS PUNCHED :     0     CARDS READ :    908         **********
*********************    A GRAND TOTAL OF        4213  INPUT/OUTPUT REQUESTS WERE HANCLEC FCR THE JCB.   **********
*********************    THE AVAILABLE CORE WAS : 200 K     OF WHICH AT LEAST   79 K WAS USED            **********
*********************                THE APPROXIMATE COST OF THIS JOB WAS  £  2.44                       **********
*********************             THIS JOB WAS EXECUTED IN THE BACKGROUND PARTITICN                      **********
* THIS LINE PRINTED AT 10.22.06  ;
```

```
############# 1 ##############   ############# 2 ##############   ############# 3 ##############
##############################   ##############################   ##############################
##    P     P     P     P O ##   ##    P     P     P     P O ##   ##    P     P     P     P O ##
## ------------------------ ##   ## ------------------------ ##   ## ------------------------ ##
##                          ##   ##                          ##   ## >                        ##
## ------------------------ ##   ## ------------------------ ##   ## ------------------------ ##
##                          ##   ##                          ##   ##                          ##
##############################   ##############################   ##############################
##############################   ##############################   ##############################

############# 4 ##############   ############# 5 ##############   ############# 6 ##############
##############################   ##############################   ##############################
##    P     P     P     P O ##   ##    P     P     P     P O ##   ##    P     P     P     P O ##
## ------------------------ ##   ## ------------------------ ##   ## ------------------------ ##
##  >                       ##   ##  > >                     ##   ##   > >                    ##
## ------------------------ ##   ## ------------------------ ##   ## ------------------------ ##
##                          ##   ##                          ##   ##                          ##
##############################   ##############################   ##############################
##############################   ##############################   ##############################

############# 7 ##############   ############# 8 ##############   ############# 9 ##############
##############################   ##############################   ##############################
##    P     P     P     P O ##   ##    P     P     P     P O ##   ##    P     P     P     P O ##
## ------------------------ ##   ## ------------------------ ##   ## ------------------------ ##
## > > >                    ##   ##   > > >                  ##   ## > > > >                  ##
## ------------------------ ##   ## ------------------------ ##   ## ------------------------ ##
##                          ##   ##                          ##   ##                          ##
##############################   ##############################   ##############################
##############################   ##############################   ##############################

############# 10 #############   ############# 11 #############   ############# 12 #############
##############################   ##############################   ##############################
##    P     P     P     P O ##   ##    P     P     P     P O ##   ##    P     P     P     P O ##
## ------------------------ ##   ## ------------------------ ##   ## ------------------------ ##
##  > > > >                 ##   ##  > > > > >               ##   ##   > > > > >              ##
## ------------------------ ##   ## ------------------------ ##   ## ------------------------ ##
##                          ##   ##                          ##   ##                          ##
##############################   ##############################   ##############################
##############################   ##############################   ##############################
```

```
############ 385 ###########   ############ 386 ###########   ############ 387 ###########
############################   ############################   ############################
##    P    P     P    P @ ##   ##    P    P     P    P @ ##   ##    P    P     P    P @ ##
## ______________________ ##   ## ______________________ ##   ## ______________________ ##
##  > >>>>>   >>>>>>>>>>>> ##   ##   > >>>>>    >>>>>>>>>>> ##   ##  > > >>>>>   >>>>>>>>>> ##
## ---------------------- ##   ## ---------------------- ##   ## ---------------------- ##
##                19       #   ##                20       #   ##                21       #
############################   ############################   ############################
############################   ############################   ############################

############ 388 ###########   ############ 389 ###########   ############ 390 ###########
############################   ############################   ############################
##    P    P     P    P @ ##   ##    P    P     P    P @ ##   ##    P    P     P    P @ ##
## ______________________ ##   ## ______________________ ##   ## ______________________ ##
##   > > >>>>>   >>>>>>>>> ##   ##  > > > >>>>>   >>>>>>>> ##   ##   >> > >>>>>   >>>>>>>> ##
## ---------------------- ##   ## ---------------------- ##   ## ---------------------- ##
##                22       #   ##                23       #   ##                23       #
############################   ############################   ############################
############################   ############################   ############################

############ 391 ###########   ############ 392 ###########   ############ 393 ###########
############################   ############################   ############################
##    P    P     P    P C ##   ##    P    P     P    P O ##   ##    P    P     P    P C ##
## ______________________ ##   ## ______________________ ##   ## ______________________ ##
##  >>>   > >>>>> >>>>>>>> ##   ##  >>>    > >>>>> >>>>>>>> ##   ## 1>>>     > >>>>>>>>>>>>> ##
## ---------------------- ##   ## ---------------------- ##   ## ---------------------- ##
##                23       #   ##                23       #   ##                22       #
############################   ############################   ############################
############################   ############################   ############################

############ 394 ###########   ############ 395 ###########   ############ 396 ###########
############################   ############################   ############################
##    P    P     P    P O ##   ##    P    P     P    P O ##   ##    P    P     P    P C ##
## ______________________ ##   ## ______________________ ##   ## ______________________ ##
##  >>>>    > >>>>>>>>>>>> ##   ##  >>>>>    > >>>>>>>>>>> ##   ##   >>>>>    > >>>>>>>>>> ##
## ---------------------- ##   ## ---------------------- ##   ## ---------------------- ##
##                22       #   ##                22       #   ##                22       #
############################   ############################   ############################
############################   ############################   ############################
```

```
############ 397 ###########  ############ 398 ###########  ############ 399 ###########
############################  ############################  ############################
##   P    P    P     P O ##  ##   P    P    P     P O ##  ##   P    P    P     P C ##
## ______________________ ##  ## ______________________ ##  ## ______________________ ##
##  > >>>>>     > >>>>>>>>> ##  ##   >> >>>>     > >>>>>>>> ##  ##  >>>  >>>>     >>>>>>>>> ##
## ---------------------- ##  ## ---------------------- ##  ## ---------------------- ##
##              22        #  ##              22        #  ##              21        #
############################  ############################  ############################
############################  ############################  ############################

############ 400 ###########
############################
##   P    P    P     P O ##
## ______________________ ##
##  >>>   >>>>    >>>>>>>> ##
## ---------------------- ##
##              21        #
############################
############################
```

```
THE VALUE OF THE CLOCK AT THE END OF THE SIMULATION IS  400


                          QUEUE STATS

                   AV. LENGTH  AV. WAIT TIME   MAX      USERS   LENGTH AT END
ENTER BUS LANE        0.15         0.30          3       198          1
ENTER BUS STOP 2      0.17         0.34          4       194          0
ENTER BUS STOP 3      0.17         0.35          6       192          0
ENTER BUS STOP 4     12.96        31.16         28       166         22
LEAVING BUS LANE      0.40         0.96          1       165          0
TRAFFIC LIGHTS        1.26         3.07          3       163          0


                    FACILITY STATS

                     USERS    AVERAGE USE  AV.TIME SPENT  USERS AT END
 START OF BUS LANE   197         0.61         1.22           1
BUS STOP 1           196         0.64         1.29           1
BUS STOP 2           193         0.53         1.09           1
BUS STOP 3           192         0.51         1.06           0
BUS STOP 4           165         0.87         2.11           1


                          TABLES

                     NUMBER       MAX         MIN       AVERAGE STANDARD DEVIATION
TOTAL TIME BUS TAKES   21          79          27        54.81      15.26
TOTAL TIME CAR TAKES  142          81          23        56.99      16.38

186.92 SECONDS IN EXECUTION
/&                                                                                    09.40.04



********************     ST.ANDREWS UNIVERSITY COMPUTING LABORATORY - 44MFT/SPOOLER END-OF-JOB RECORD    *********
******************** JOB NAME : GHSIM      DATE : 75276      START TIME : 09.35.28   ELAPSED : 00.04.37 *********
********************     CPU TIME : 00.03.27     WAIT TIME : 00.00.27     SYSTEM OVERHEAD : 00.00.25     *********
********************     PAGES PRINTED :    64     CARDS PUNCHED :      0     CARDS READ :    889         *********
********************     A GRAND TOTAL OF         4169  INPUT/OUTPUT REQUESTS WERE HANDLED FOR THE JOB.   *********
********************     THE AVAILABLE CORE WAS : 200 K     OF WHICH AT LEAST   79 K WAS USED           *********
********************                THE APPROXIMATE COST OF THIS JOB WAS  £  2.44                         *********
********************               THIS JOB WAS EXECUTED IN THE BACKGROUND PARTITION                     *********
* THIS LINE PRINTED AT 09.47.05  ;
```

Example 2

A busy crossroads exists in a town and it is wanted to know whether traffic lights would improve the traffic flow and what time intervals to set the lights at red and green.

On the main road, vehicles arrive every 2 ticks and on the side road every 4 ticks.

The picture, which includes traffic lights, used for data is as follows:

```
            |  B75|  B30|
            |  B74|  B31|
            |  B73|  B32|
            |  B72|  B33|
- -  - -  - +  B71.  F13+  - -  - -  -
B20B21B22B23F1 F2 F3 F4 B81B82B83B84B85
- -  - -  - .  F5 F6 F7 .  - -  - -
B65B64B63B62B61F8 F9 F10F11B43B42B41B40
- -  - -  - +  F12.  B51+  - -  - -  -
            |  B13|  B52|
            |  B12|  B53|
            |  B11|  B54|
            |  B10|  B55|
```

When all the signals are set at green, as in the second computer output example, that indicates that there are no lights at all and the results without traffic lights can be obtained.

From Table 3 it can be seen that traffic flow moves quicker without lights than with them, but if lights were installed then the sequence of 8 ticks with the main road green, 4 ticks with the side road green and all the lights red for 2 ticks between each change.

The listing of the program with lights is shown, together with output for some of the simulation both with and without lights.

TABLE 3 - The time vehicles take in relation to the sequence of the traffic lights

Traffic Light Sequence				The time vehicles take between points											
All red	Main Green	Side Green	Total	NE	NS	NW	SN	SE	SW	EN	ES	EW	WN	WE	WS
2	8	4	16	18.97	15.60	17.89	17.93	20.33	17.00	18.00	12.12	15.85	14.20	21.00	15.34
2	10	6	20	16.45	18.00	20.11	17.50	17.29	14.56	18.83	13.62	16.66	13.60	19.56	16.69
2	13	7	24	18.61	17.80	23.75	19.11	20.63	18.00	18.83	13.50	16.95	14.00	23.00	16.70
2	16	8	28	21.45	22.40	22.37	21.96	25.00	17.67	26.00	13.25	17.03	15.90	18.50	17.01
2	18	10	32	20.68	16.00	21.57	20.52	17.57	20.33	28.33	14.75	21.00	15.40	27.14	17.68
2	24	12	40	24.29	22.00	30.57	24.67	23.86	25.00	20.67	17.25	19.87	16.70	26.00	18.44
No lights				13.19	10.40	15.11	13.07	13.22	10.33	14.67	10.00	12.51	10.00	13.67	12.41

```
COMMENT SIMULATION OF CROSS ROADS;

RECORD THING(INTEGER DIRECTION,ORIGIN,S);
REFERENCE(QUEUE)Q1,Q2,Q3,Q4;
REFERENCE(FACILITY)F1,F2,F3,F4,F5,F6,F7,F8,F9,F10,F11,F12,F13;
REFERENCE(BELT)B1,B2,B3,B4,B5,B6,B7,B8;
REFERENCE(SIGNAL)S1,S2;
REFERENCE(TABLE)T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12;
INTEGER R;
LOGICAL RERUN,SECOND;
INTEGER TALLRED,TS2GREEN,TS1GREEN,S2GREEN,S2RED,S1GREEN,S1RED;
INTEGER ENTERSOUTH,SOUTHLIGHTS,ENTERWEST,WESTLIGHTS,ENTERNORTH,
NORTHLIGHTS,ENTEREAST,EASTLIGHTS,LEAVESOUTH,LEAVEWEST,LEAVENORTH,
LEAVEEAST;
ENTERSOUTH:=1; SOUTHLIGHTS:=2; ENTERWEST:=3; WESTLIGHTS:=4;
ENTERNORTH:=5; NORTHLIGHTS:=6; ENTEREAST:=7; EASTLIGHTS:=8;
LEAVESOUTH:=9; LEAVEWEST:=10; LEAVENORTH:=11; LEAVEEAST:=12;
S2GREEN:=13;  S2RED:=14;  S1GREEN:=15;  S1RED:=16;
SECOND:=FALSE;
R:=45673201;
INITIALISE;
READ(TALLRED,TS2GREEN,TS1GREEN);
SETUPQ("Q1",Q1);
SETUPQ("Q2",Q2);
SETUPQ("Q3",Q3);
SETUPQ("Q4",Q4);
SETUPF("F1",F1,1,3);
SETUPF("F2",F2,1,0);
SETUPF("F3",F3,1,0);
SETUPF("F4",F4,1,0);
SETUPF("F5",F5,1,0);
SETUPF("F6",F6,1,0);
SETUPF("F7",F7,1,0);
SETUPF("F8",F8,1,0);
SETUPF("F9",F9,1,0);
SETUPF("F10",F10,1,0);
SETUPF("F11",F11,1,4);
SETUPF("F12",F12,1,1);
SETUPF("F13",F13,1,2);
SETUPBELT(B1,3,1);
SETUPBELT(B2,3,3);
SETUPBELT(B3,3,2);
SETUPBELT(B4,3,4);
SETUPBELT(B5,5,2);
SETUPBELT(B6,5,4);
SETUPBELT(B7,5,1);
SETUPBELT(B8,5,3);
SETUPSIGNAL(S1,FALSE);
SETUPSIGNAL(S2,TRUE);
SETUPTABLE("NORTH TO SOUTH",T1);
SETUPTABLE("NORTH TO EAST",T2);
SETUPTABLE("NORTH TO WEST",T3);
SETUPTABLE("SOUTH TO NORTH",T4);
SETUPTABLE("SOUTH TO EAST",T5);
SETUPTABLE("SOUTH TO WEST",T6);
SETUPTABLE("EAST TO NORTH",T7);
SETUPTABLE("EAST TO SOUTH",T8);
SETUPTABLE("EAST TO WEST",T9);
SETUPTABLE("WEST TO NORTH",T10);
SETUPTABLE("WEST TO SOUTH",T11);
SETUPTABLE("WEST TO EAST",T12);
NOTE(S2RED,3);
WHILE CLOCK <=200 DO
BEGIN INTEGER I;
  RERUN:=TRUE;
  IF AT(LEAVESOUTH) THEN
  BEGIN
    TAKE(B5);
    CASE ORIGIN(THETHING) OF
    BEGIN
      COLLECT(T1,CLOCK-S(THETHING)); ;
      COLLECT(T8,CLOCK-S(THETHING));
      COLLECT(T11,CLOCK-S(THETHING))
    END
  END;
  IF AT(LEAVEWEST) THEN
  BEGIN
    TAKE(B6);
    CASE ORIGIN(THETHING) OF
    BEGIN
      COLLECT(T3,CLOCK-S(THETHING));
      COLLECT(T6,CLOCK-S(THETHING));
      COLLECT(T9,CLOCK-S(THETHING))
    END
```

```
0553 -4          END;
0554 --          IF AT(LEAVENORTH) THEN
0554 4-          BEGIN
0555 --            TAKE(B7);
0556 --            CASE ORIGIN(THETHING) OF
0556 5-            BEGIN
0557 --              ;
0558 --              COLLECT(T4,CLOCK-S(THETHING));
0559 --              COLLECT(T7,CLOCK-S(THETHING));
0560 --              COLLECT(T10,CLOCK-S(THETHING))
0560 -5            END
0560 -4          END;
0561 --          IF AT(LEAVEEAST) THEN
0561 4-          BEGIN
0562 --            TAKE(B8);
0563 --            CASE ORIGIN(THETHING) OF
0563 5-            BEGIN
0564 --              COLLECT(T2,CLOCK-S(THETHING));
0565 --              COLLECT(T5,CLOCK-S(THETHING)); ;
0567 --              COLLECT(T12,CLOCK-S(THETHING))
0567 -5            END
0567 -4          END;
0568 --          IF BUSY(F4,3) THEN
0568 4-          BEGIN
0569 --            RELEASE(F4);
0570 --            PUT(B8,LEAVEEAST)
0570 -4          END;
0571 --          IF BUSY(F2,1) THEN
0571 4-          BEGIN
0572 --            RELEASE(F2);
0573 --            PUT(B7,LEAVENORTH)
0573 -4          END;
0574 --          IF BUSY(F8,4) THEN
0574 4-          BEGIN
0575 --            RELEASE(F8);
0576 --            PUT(B6,LEAVEWEST)
0576 -4          END;
0577 --          IF BUSY(F10,2) THEN
0577 4-          BEGIN
0578 --            RELEASE(F10);
0579 --            PUT(B5,LEAVESOUTH)
0579 -4          END;
```

```
WHILE RERUN DO
BEGIN INTEGER J;
  RERUN:=FALSE;
  IF READY(F3,3) AND IDLE(F4,0) AND (SECOND OR IDLE(F13,0)) THEN
  BEGIN
    RELEASE(F3);
    SEIZE(F4,3)
  END;
  IF READY(F7,2) AND IDLE(F10,0) AND (SECOND OR IDLE(F11,0)) THEN
  BEGIN
    RELEASE(F7);
    SEIZE(F10,2)
  END;
  IF READY(F9,4) AND IDLE(F8,0) AND (SECOND OR IDLE(F12,0)) THEN
  BEGIN
    RELEASE(F9);
    SEIZE(F8,4)
  END;
  IF READY(F5,1) AND IDLE(F2,0) AND (SECOND OR IDLE(F1,0)) THEN
  BEGIN
    RELEASE(F5);
    SEIZE(F2,1)
  END;
  IF READY(F8,1) AND IDLE(F5,0) THEN
  BEGIN
    RELEASE(F8);
    IF ¬SECOND THEN RERUN:=TRUE;
    SEIZE(F5,DIRECTION(THETHING))
  END;
  IF READY(F2,3) AND IDLE(F3,0) THEN
  BEGIN
    RELEASE(F2);
    IF ¬SECOND THEN RERUN:=TRUE;
    SEIZE(F3,DIRECTION(THETHING))
  END;
  IF READY(F4,2) AND IDLE(F7,0) THEN
  BEGIN
    RELEASE(F4);
    IF ¬SECOND THEN RERUN:=TRUE;
    SEIZE(F7,DIRECTION(THETHING))
  END;
  IF READY(F10,4) AND IDLE(F9,0) THEN
```

```
BEGIN
  RELEASE(F10);
  IF ¬SECOND THEN RERUN:=TRUE;
  SEIZE(F9,DIRECTION(THETHING))
END;
IF READY(F6,3) AND IDLE(F3,0) THEN
BEGIN
  RELEASE(F6);
  SEIZE(F3,3)
END;
IF READY(F6,2) AND IDLE(F7,0) THEN
BEGIN
  RELEASE(F6);
  SEIZE(F7,2)
END;
IF READY(F6,4) AND IDLE(F9,0) THEN
BEGIN
  RELEASE(F6);
  SEIZE(F9,4)
END;
IF READY(F6,1) AND IDLE(F5,0) THEN
BEGIN
  RELEASE(F6);
  SEIZE(F5,1)
END;
IF READY(F5,3) AND IDLE(F3,0) THEN
BEGIN
  RELEASE(F5);
  IF ¬SECOND THEN RERUN:=TRUE;
  SEIZE(F3,3)
END ELSE
IF READY(F5,3) AND IDLE(F6,0) THEN
BEGIN
  RELEASE(F5);
  IF ¬SECOND THEN RERUN:=TRUE;
  SEIZE(F6,3)
END;
IF READY(F3,2) AND IDLE(F7,0) THEN
BEGIN
  RELEASE(F3);
  IF ¬SECOND THEN RERUN:=TRUE;
  SEIZE(F7,2)
```

```
END ELSE
IF READY(F3,2) AND IDLE(F6,0) THEN
BEGIN
  RELEASE(F3);
  IF ¬SECOND THEN RERUN:=TRUE;
  SEIZE(F6,2)
END;
IF READY(F7,4) AND IDLE(F9,0) THEN
BEGIN
  RELEASE(F7);
  IF ¬SECOND THEN RERUN:=TRUE;
  SEIZE(F9,4)
END ELSE
IF READY(F7,4) AND IDLE(F6,0) THEN
BEGIN
  RELEASE(F7);
  IF ¬SECOND THEN RERUN:=TRUE;
  SEIZE(F6,4)
END;
IF READY(F9,1) AND IDLE(F5,0) THEN
BEGIN
  RELEASE(F9);
  IF ¬SECOND THEN RERUN:=TRUE;
  SEIZE(F5,1)
END ELSE
IF READY(F9,1) AND IDLE(F6,0) THEN
BEGIN
  RELEASE(F9);
  IF ¬SECOND THEN RERUN:=TRUE;
  SEIZE(F6,1)
END;
IF READY(F12,0) AND IDLE(F8,0) THEN
BEGIN
  IF IDLE(F5,3) OR IDLE(F6,0) OR IDLE(F3,0) THEN
  BEGIN
    RELEASE(F12);
    SEIZE(F8,IF DIRECTION(THETHING)=4 THEN 4 ELSE 1)
  END ELSE
  IF ¬SECOND THEN RERUN:=TRUE
END;
IF READY(F1,0) AND IDLE(F2,0) THEN
BEGIN
```

```
0656 --            IF IDLE(F3,2) OR IDLE(F6,0) OR IDLE(F7,0) THEN
0656 6-            BEGIN
0657 --              RELEASE(F1);
0658 --              SEIZE(F2,IF DIRECTION(THETHING)=1 THEN 1 ELSE 3)
0658 -6            END ELSE
0658 --            IF ¬SECOND THEN RERUN:=TRUE
0658 -5          END;
0659 --          IF READY(F13,0) AND IDLE(F4,0) THEN
0659 5-          BEGIN
0660 --            RELEASE(F13);
0661 --            SEIZE(F4,IF DIRECTION(THETHING)=3 THEN 3 ELSE 2)
0661 -5          END ELSE
0661 --          IF READY(F3,3) AND IDLE(F4,0) THEN RERUN:=TRUE;
0662 --          IF READY(F11,0) AND IDLE(F10,0) THEN
0662 5-          BEGIN
0663 --            RELEASE(F11);
0664 --            SEIZE(F10,IF DIRECTION(THETHING)=2 THEN 2 ELSE 4)
0664 -5          END ELSE
0664 --          IF READY(F7,2) AND IDLE(F10,0) THEN RERUN:=TRUE;
0665 --          SECOND:=RERUN
0665 -4        END;
0666 --        IF AT(SOUTHLIGHTS) THEN ENTER(Q1);
0667 --        IF AT(WESTLIGHTS) THEN ENTER(Q2);
0668 --        IF AT(NORTHLIGHTS) THEN ENTER(Q3);
0669 --        IF AT(EASTLIGHTS) THEN ENTER(Q4);
0670 --        IF GREEN(S1) AND ¬EMPTY(Q1) AND IDLE(F12,0) THEN
0670 4-        BEGIN
0671 --          NEXT(Q1);
0672 --          SEIZE(F12,0);
0673 --          TAKE(B1)
0673 -4        END;
0674 --        IF GREEN(S2) AND ¬EMPTY(Q2) AND IDLE(F1,0) THEN
0674 4-        BEGIN
0675 --          NEXT(Q2);
0676 --          SEIZE(F1,3);
0677 --          TAKE(B2)
0677 -4        END;
0678 --        IF GREEN(S1) AND ¬EMPTY(Q3) AND IDLE(F13,0) THEN
0678 4-        BEGIN
0679 --          NEXT(Q3);
0680 --          SEIZE(F13,2);
0681 --          TAKE(B3)
```

```
END;
IF GREEN(S2) AND ¬EMPTY(Q4) AND IDLE(F11,0) THEN
BEGIN
  NEXT(Q4);
  SEIZE(F11,0);
  TAKE(B4)
END;
IF AGAIN(ENTERSOUTH,4) THEN
BEGIN
  THETHING:=THING(IF PERHAPS(0.68,R) THEN 1
                      ELSE IF PERHAPS(0.5,R) THEN 4 ELSE 3,2,CLOCK);
  PUT(B1,SOUTHLIGHTS)
END;
IF AGAIN(ENTERWEST,2) THEN
BEGIN
  THETHING:=THING(IF PERHAPS(0.8,R) THEN 3
                      ELSE IF PERHAPS(0.5,R) THEN 1 ELSE 2,4,CLOCK);
  PUT(B2,WESTLIGHTS)
END;
IF AGAIN(ENTERNORTH,4) THEN
BEGIN
  THETHING:=THING(IF PERHAPS(0.68,R) THEN 2
                      ELSE IF PERHAPS(0.5,R) THEN 4 ELSE 3,1,CLCCK);
  PUT(B3,NORTHLIGHTS)
END;
IF AGAIN(ENTEREAST,2) THEN
BEGIN
  THETHING:=THING(IF PERHAPS(0.8,R) THEN 4
                      ELSE IF PERHAPS(0.5,R) THEN 1 ELSE 2,3,CLOCK);
  PUT(B4,EASTLIGHTS)
END;
IF AT(S2RED) THEN
BEGIN
  CHANGE(S2);
  NCTE(S1GREEN,TALLRED)
END;
IF AT(S1GREEN) THEN
BEGIN
  CHANGE(S1);
  NOTE(S1RED,TS1GREEN)
END;
IF AT(S1RED) THEN
```

```
NUMAC  ALGOL W  (01JULY72)                                                   03 CCTOBER 1975  @  09:09

0704 4-         BEGIN
0705 --           CHANGE(S1);
0706 --           NOTE(S2GREEN,TALLRED)
0706 -4         END;
0707 --         IF AT(S2GREEN) THEN
0707 4-         BEGIN
0708 --           CHANGE(S2);
0709 --           NOTE(S2RED,TS2GREEN)
0709 -4         END;
0710 --         TICK
0710 -3       END;
0711 --       OUTSTATS
0711 -2     END
0711 -1     END.

EXECUTION OPTIONS: DEBUG,0 TIME=97369 PAGES=32767
```

```
###### 1 #######   ###### 2 #######   ###### 3 #######   ###### 4 #######   ###### 5 #######
################   ################   ################   ################   ################
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | |V|   ##
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##
##   O| | |@  ##   ##   O| | |@  ##   ##   O| | |@  ##   ##   @| | |@  ##   ##   @| | |@  ##
##____+ . +___##   ##____+ . +___##   ##____+ . +___##   ##____+ . +___##   ##____+ . +___##
##            ##   ##            ##   ## >          ##   ##  >         ##   ## > >        ##
##----.   .---##   ##----.   .---##   ##----.   .---##   ##----.   .---##   ##----.   .---##
##            ##   ##            ##   ##           <##   ##          < ##   ##         < <##
##____+ . +___##   ##____+ . +___##   ##____+ . +___##   ##____+ . +___##   ##____+ . +___##
##   @| | |O  ##   ##   @| | |O  ##   ##   @| | |O  ##   ##   @| | |@  ##   ##   @| | |@  ##
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    |A| |   ##
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##
################   ################   ################   ################   ################
################   ################   ################   ################   ################

###### 6 #######   ###### 7 #######   ###### 8 #######   ###### 9 #######   ###### 10 ######
################   ################   ################   ################   ################
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | |V|   ##   ##    | | |   ##
##    | |V|   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | |V|   ##
##   @| | |O  ##   ##   @| |V|O  ##   ##   @| | |O  ##   ##   @| | |O  ##   ##   @| | |@  ##
##____+ . +___##   ##____+ . +___##   ##____+ .V+___##   ##____+ . +___##   ##____+ . +___##
##  >>        ##   ## >>>        ##   ## >>>        ##   ##1>>>   V    ##   ##1>>>        ##
##----.   .---##   ##----.   .---##   ##----.   .---##   ##----.   .---##   ##----.A V.---##
##         << ##   ##         <<<##   ##         <<<##   ##     A  <<<1##   ##        <<<1##
##____+ . +___##   ##____+ . +___##   ##____+A. +___##   ##____+ . +___##   ##____+ . +___##
##   O| | |@  ##   ##   O|A| |@  ##   ##   O| | |@  ##   ##   O| | |@  ##   ##   @| | |@  ##
##    |A| |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    |A| |   ##
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    |A| |   ##   ##    | | |   ##
##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##   ##    | | |   ##
################   ################   ################   ################   ################
################   ################   ################   ################   ################
```

```
###### 181 ######  ###### 182 ######  ###### 183 ######  ###### 184 ######  ###### 185 ######
#################  #################  #################  #################  #################
##    | |1|    ##  ##    | |1|    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##    | |V|    ##  ##    | |V|    ##  ##    | |V|    ##  ##    | | |    ##  ##    | |V|    ##
##    | |V|    ##  ##    | |V|    ##  ##    | |V|    ##  ##    | |V|    ##  ##    | | |    ##
##   @| |V|@   ##  ##   @| |V|O   ##  ##   @| |V|O   ##  ##   @| |V|O   ##  ##   @| |V|C   ##
##____+ . +____##  ##____+ . +____##  ##____+ .V+____##  ##____+ .V+____##  ##____+ .V+____##
## > > >>>>>>> ##  ##  >>  >>>>>>>##  ## >>>   >>>>>>##  ## >>>   V>>>>>##  ##1>>>   V >>>>##
##----.  V.----##  ##----.   .----##  ##----.   .----##  ##----.   .----##  ##----.A V.----##
##<<<<<<<< < < ##  ##<<<<<<<V <<  ##  ##<<<<<<   <<< ##  ##<<<<<A   <<< ##  ##<<<< A   <<<1##
##____+ . +____##  ##____+ . +____##  ##____+A.V+____##  ##____+A. +____##  ##____+A. +____##
##   @|A| |@   ##  ##   O|A| |@   ##  ##   O|A| |@   ##  ##   C|A|V|@   ##  ##   O|A| |@   ##
##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    | |V|    ##
##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    | | |    ##  ##    |A| |    ##
##    |1| |    ##  ##    |1| |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
#################  #################  #################  #################  #################
#################  #################  #################  #################  #################
```

```
###### 186 ######  ###### 187 ######  ###### 188 ######  ###### 189 ######  ###### 190 ######
#################  #################  #################  #################  #################
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | |V|    ##  ##    |A| |    ##
##    | |V|    ##  ##    | | |    ##  ##    | | |    ##  ##    |A| |    ##  ##    |A|V|    ##
##   @| | |@   ##  ##   @| |V|@   ##  ##   O|A|V|@   ##  ##   O|A|V|@   ##  ##   C| |V|@   ##
##____+ .V+____##  ##____+A. +____##  ##____+A. +____##  ##____+ . +____##  ##____+ . +____##
##1>>> A V  >>>##  ##2>>> A V   >>##  ##2>>>  >     >##  ##2>>>> >>     ##  ##1>>>>> >>    ##
##----.A V.----##  ##----.> <.----##  ##----.> <.----##  ##----.   .----##  ##----.   .----##
##<<<  A V <<<1##  ##<<   A V <<<2##  ##<     <  <<<2##  ##     << <<<<2##  ##    << <<<<<1##
##____+A. +____##  ##____+ .V+____##  ##____+ .V+____##  ##____+ . +____##  ##____+ . +____##
##   @| | |@   ##  ##   @|A| |@   ##  ##   @|A|V|O   ##  ##   @|A|V|O   ##  ##   @|A| |C   ##
##    |A| |    ##  ##    | | |    ##  ##    | | |    ##  ##    | |V|    ##  ##    |A|V|    ##
##    | |V|    ##  ##    | | |    ##  ##    | | |    ##  ##    |A| |    ##  ##    | |V|    ##
##    | | |    ##  ##    | |V|    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
#################  #################  #################  #################  #################
#################  #################  #################  #################  #################
```

```
###### 191 ######  ###### 192 ######  ###### 193 ######  ###### 194 ######  ###### 195 ######
#################  #################  #################  #################  #################
##    |A| |    ##  ##    |A| |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##    |A| |    ##  ##    | | |    ##  ##    | |V|    ##  ##    | |V|    ##  ##    |A|V|    ##
##    | |V|    ##  ##    | |V|    ##  ##    | |V|    ##  ##    |A|V|    ##  ##    | |V|    ##
##   O| |V|@   ##  ##   O| |V|@   ##  ##   O|A|V|@   ##  ##   C| |V|@   ##  ##   O| |V|@   ##
##____+ . +____##  ##____+A. +____##  ##____+ . +____##  ##____+ . +____##  ##____+ . +____##
##1>>>>AV >>   ##  ## >>>>>   >>  ##  ## >>>>>>   >> ##  ##  >>>>>>   >>##  ## > >>>>>>   >##
##----.   .----##  ##----.  V.----##  ##----.  V.----##  ##----.  V.----##  ##----.  V.----##
##   << <<<<<<1##  ##  << <<<<<<< ##  ## << <<<<<<<< ##  ##<< <<<<<<<<  ##  ##< <<<<<<<< < ##
##____+ . +____##  ##____+ . +____##  ##____+ . +____##  ##____+ . +____##  ##____+ . +____##
##   @|A| |O   ##  ##   @|A| |O   ##  ##   @|A| |O   ##  ##   @|A| |O   ##  ##   @|A| |O   ##
##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##
##    | |V|    ##  ##    | | |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##
##    | |V|    ##  ##    | |V|    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
#################  #################  #################  #################  #################
#################  #################  #################  #################  #################

###### 196 ######  ###### 197 ######  ###### 198 ######  ###### 199 ######  ###### 200 ######
#################  #################  #################  #################  #################
##    |A| |    ##  ##    | |1|    ##  ##    | |1|    ##  ##    | | |    ##  ##    | | |    ##
##    | |V|    ##  ##    | |V|    ##  ##    | |V|    ##  ##    | |V|    ##  ##    | | |    ##
##    | |V|    ##  ##    | |V|    ##  ##    | |V|    ##  ##    | |V|    ##  ##    | |V|    ##
##   @| |V|@   ##  ##   @| |V|@   ##  ##   @| |V|O   ##  ##   @| |V|C   ##  ##   @| |V|C   ##
##____+ . +____##  ##____+ . +____##  ##____+ . +____##  ##____+ .V+____##  ##____+A.V+____##
##  > >>>>>>   ##  ## > > >V>>>>  ##  ##  >>  > >>>> ##  ## >>> A > >>>>##  ## >>>   V> >>>##
##----.  V.----##  ##----.  V.----##  ##----.A V.----##  ##----.   .----##  ##----.   .----##
## <<<<<<<< <  ##  ##<<<<<AV  < < ##  ##<<<<<  V <<  ##  ##<<<<   V <<< ##  ##<<<  A   <<< ##
##____+ . +____##  ##____+ . +____##  ##____+ .V+____##  ##____+A.V+____##  ##____+A.V+____##
##   @|A| |@   ##  ##   @|A| |@   ##  ##   O|A| |@   ##  ##   C|A|V|@   ##  ##   C|A|V|@   ##
##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    |A|V|    ##
##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    |A| |    ##  ##    | | |    ##
##    | | |    ##  ##    |1| |    ##  ##    |1| |    ##  ##    | | |    ##  ##    | | |    ##
#################  #################  #################  #################  #################
#################  #################  #################  #################  #################
```

```
###### 201 ######
#################
##    | | |    ##
##    | |V|    ##
##    | | |    ##
##   @|A|V|O   ##
##____+ .V+____##
##1>>>   V > >>##
##----.A V.----##
##<<   A   <<<1##
##____+A. +____##
##   O|A|V|@   ##
##    | |V|    ##
##    |A|V|    ##
##    | | |    ##
#################
#################
```

```
THE VALUE OF THE CLOCK AT THE END OF THE SIMULATION IS  201

                              QUEUE STATS

                      AV. LENGTH  AV. WAIT TIME   MAX        USERS   LENGTH AT END
Q1                       1.55         6.48          3          48          1
Q2                       1.70         3.52          4          96          2
Q3                       1.55         6.48          3          48          1
Q4                       1.70         3.52          4          96          2


                       FACILITY STATS

                        USERS      AVERAGE USE  AV.TIME SPENT  USERS AT END
F1                        96          0.49         1.00           0
F2                       130          0.65         1.00           0
F3                        95          0.49         1.02           0
F4                       131          0.65         1.00           1
F5                        43          0.26         1.21           1
F6                         4          0.05         2.00           0
F7                        50          0.38         1.52           1
F8                       135          0.67         1.00           1
F9                        96          0.51         1.05           0
F10                      137          0.70         1.01           0
F11                       96          0.50         1.02           0
F12                       47          0.23         1.00           1
F13                       47          0.23         1.00           1


                             TABLES

                       NUMBER        MAX          MIN       AVERAGE STANDARD DEVIATION
NORTH TO SOUTH           31           23           12        18.97        3.47
NORTH TO EAST             5           18           12        15.60        2.94
NORTH TO WEST             9           25           15        17.89        3.35
SOUTH TO NORTH           27           23           12        17.93        3.47
SOUTH TO EAST             9           25           15        20.33        3.20
SOUTH TO WEST             9           21           12        17.00        3.74
EAST TO NORTH             6           25           13        18.00        3.74
EAST TO SOUTH             8           16           10        12.12        2.09
EAST TO WEST             78           19           12        15.85        2.21
WEST TO NORTH            10           17           11        14.20        2.14
WEST TO SOUTH             8           25           16        21.00        3.50
WEST TO EAST              73          19          12          15.34         2.29

123.32 SECONDS IN EXECUTION
/&                                                                                     C9.13.04
```

```
********************    ST.ANDREWS UNIVERSITY COMPUTING LABORATORY - 44MFT/SPOOLER END-OF-JOB RECCRD    **********
******************** JOB NAME : GHSIM      DATE : 75276      START TIME : 09.09.10   ELAPSED : 00.03.55 **********
********************    CPU TIME : 00.02.29     WAIT TIME : 00.00.31     SYSTEM OVERHEAD : 00.CC.30     **********
********************    PAGES PRINTED :     58     CARDS PUNCHED :      0     CARDS READ :  1127        **********
********************    A GRAND TOTAL OF         4073  INPUT/OUTPUT REQUESTS WERE HANDLED FOR THE JOB.  **********
********************    THE AVAILABLE CORE WAS : 200 K      OF WHICH AT LEAST   89 K WAS USED           **********
********************                THE APPROXIMATE COST OF THIS JOB WAS  £  1.91                       **********
********************             THIS JOB WAS EXECUTED IN THE BACKGROUND PARTITION                      **********
* THIS LINE PRINTED AT 09.19.22  ;
```

```
###### 1 ########  ###### 2 ########  ###### 3 ########  ###### 4 ########  ###### 5 ########
#################  #################  #################  #################  #################
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | |V|    ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##   O| | |O   ##  ##   O| | |O   ##  ##   O| | |O   ##  ##   O| | |O   ##  ##   O| | |O   ##
##____+ . +____##  ##____+ . +____##  ##____+ . +____##  ##____+ . +____##  ##____+ . +____##
##             ##  ##             ##  ## >           ##  ##  >          ##  ## > >         ##
##----.   .----##  ##----.   .----##  ##----.   .----##  ##----.   .----##  ##----.   .----##
##             ##  ##             ##  ##          < ##  ##         <  ##  ##         < < ##
##____+ . +____##  ##____+ . +____##  ##____+ . +____##  ##____+ . +____##  ##____+ . +____##
##   O| | |O   ##  ##   O| | |O   ##  ##   O| | |O   ##  ##   O| | |O   ##  ##   O| | |O   ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    |A| |    ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
#################  #################  #################  #################  #################
#################  #################  #################  #################  #################

###### 6 ########  ###### 7 ########  ###### 8 ########  ###### 9 ########  ###### 10 #######
#################  #################  #################  #################  #################
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | |V|    ##  ##    | | |    ##
##    | |V|    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | |V|    ##
##   O| | |O   ##  ##   O| |V|O   ##  ##   O| | |O   ##  ##   O| | |O   ##  ##   O| | |O   ##
##____+ . +____##  ##____+ . +____##  ##____+ .V+____##  ##____+ . +____##  ##____+ . +____##
##  > >        ##  ## > > >       ##  ##  > > >      ##  ## > > >>V     ##  ##  > > >>     ##
##----.   .----##  ##----.   .----##  ##----.   .----##  ##----.   .----##  ##----.A V.----##
##        < <  ##  ##       < < < ##  ##      < < <  ##  ##     A<< < < ##  ##     << < <  ##
##____+ . +____##  ##____+ . +____##  ##____+A. +____##  ##____+ . +____##  ##____+ . +____##
##   O| | |O   ##  ##   O|A| |O   ##  ##   O| | |O   ##  ##   O| | |O   ##  ##   O| | |O   ##
##    |A| |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    |A| |    ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    |A| |    ##  ##    | | |    ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
#################  #################  #################  #################  #################
#################  #################  #################  #################  #################
```

```
###### 181 ######   ###### 182 ######   ###### 183 ######   ###### 184 ######   ###### 185 ######
#################   #################   #################   #################   #################
##    |A| |    ##   ##    | | |    ##   ##    | | |    ##   ##    | | |    ##   ##    |A| |    ##
##    | |V|    ##   ##    | | |    ##   ##    | | |    ##   ##    |A| |    ##   ##    | |V|    ##
##    | | |    ##   ##    | |V|    ##   ##    |A| |    ##   ##    | | |    ##   ##    | | |    ##
##   O| | |O   ##   ##   O|A| |O   ##   ##   O| |V|O   ##   ##   O| | |O   ##   ##   O| | |O   ##
##____+A. +____##   ##____+ . +____##   ##____+ . +____##   ##____+ .V+____##   ##____+ .V+____##
## > > >>V >>  ##   ##  > > >>  >> ##   ## > > >>>>  >>##   ##  > > V>>>  >##   ## > > A  >>>  ##
##----.   .----##   ##----.> <.----##   ##----.   .----##   ##----.   .----##   ##----.  V.----##
##  << A<< < < ##   ## <<  << < <  ##   ##<<  <<<< < < ##   ##<  <<<< < <  ##   ##  <<<A<< < < ##
##____+ .V+____##   ##____+ . +____##   ##____+ . +____##   ##____+A. +____##   ##____+ . +____##
##   O| | |O   ##   ##   O| |V|O   ##   ##   O|A| |O   ##   ##   O| | |O   ##   ##   O| | |O   ##
##    | | |    ##   ##    |A| |    ##   ##    | |V|    ##   ##    | | |    ##   ##    | | |    ##
##    |A| |    ##   ##    | | |    ##   ##    | | |    ##   ##    | |V|    ##   ##    |A| |    ##
##    | |V|    ##   ##    | | |    ##   ##    | | |    ##   ##    | | |    ##   ##    | |V|    ##
#################   #################   #################   #################   #################
#################   #################   #################   #################   #################

###### 186 ######   ###### 187 ######   ###### 188 ######   ###### 189 ######   ###### 19C ######
#################   #################   #################   #################   #################
##    | | |    ##   ##    | | |    ##   ##    | | |    ##   ##    | | |    ##   ##    |A| |    ##
##    | | |    ##   ##    | | |    ##   ##    | | |    ##   ##    |A|V|    ##   ##    | | |    ##
##    | |V|    ##   ##    | | |    ##   ##    |A| |    ##   ##    | | |    ##   ##    | |V|    ##
##   O| | |O   ##   ##   O|A|V|O   ##   ##   O| | |O   ##   ##   O| | |O   ##   ##   O| | |O   ##
##____+A. +____##   ##____+ . +____##   ##____+ .V+____##   ##____+ . +____##   ##____+ . +____##
##  > >  V >>> ##   ## > > >>   >>>##   ##  > > >>   >>##   ## > > >>V>   >##   ##  > > >> >   ##
##----.>  .----##   ##----.  <.----##   ##----. < .----##   ##----. < .----##   ##----.A<V.----##
## <<< <<V< <  ##   ##<<< << < < < ##   ##<< << < < <  ##   ##< << A<< < < ##   ## <<  << < <  ##
##____+ . +____##   ##____+ .V+____##   ##____+A. +____##   ##____+ . +____##   ##____+ . +____##
##   O| | |O   ##   ##   O|A| |O   ##   ##   O| |V|O   ##   ##   O| | |O   ##   ##   O| | |O   ##
##    |A| |    ##   ##    | | |    ##   ##    | | |    ##   ##    | |V|    ##   ##    |A| |    ##
##    | | |    ##   ##    | | |    ##   ##    | | |    ##   ##    |A| |    ##   ##    | |V|    ##
##    | | |    ##   ##    | | |    ##   ##    | | |    ##   ##    | | |    ##   ##    | | |    ##
#################   #################   #################   #################   #################
#################   #################   #################   #################   #################
```

```
###### 191 ######  ###### 192 ######  ###### 193 ######  ###### 194 ######  ###### 195 ######
#################  #################  #################  #################  #################
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##    | | |    ##  ##    | | |    ##  ##    | |V|    ##  ##    | | |    ##  ##    | | |    ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | |V|    ##  ##    |A| |    ##
##   O| |V|O   ##  ##   C| | |O   ##  ##   O| | |O   ##  ##   C|A| |O   ##  ##   O| |V|O   ##
##____+ . +____##  ##____+ .V+____##  ##____+A. +____##  ##____+ . +____##  ##____+ . +____##
## > > > >> >  ##  ##  > >A> >> > ##  ## > > >>V >> >##  ##  > > >>  >> ##  ## > > >>>>  >>##
##----.A V.----##  ##----.   .----##  ##----.   .----##  ##----.> V.----##  ##----.  V.----##
##<<  <<<< < < ##  ##<  <<<<V< <  ##  ##  <<<A<< < < ##  ## <<< << < <  ##  ##<<< << < < < ##
##____+ . +____##  ##____+A. +____##  ##____+ .V+____##  ##____+ . +____##  ##____+ . +____##
##   O|A| |O   ##  ##   C| | |O   ##  ##   O| | |O   ##  ##   C| |V|O   ##  ##   C|A| |C   ##
##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    |A| |    ##  ##    | |V|    ##
##    | | |    ##  ##    | | |    ##  ##    |A| |    ##  ##    | | |    ##  ##    | | |    ##
##    | |V|    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
#################  #################  #################  #################  #################
#################  #################  #################  #################  #################

###### 196 ######  ###### 197 ######  ###### 198 ######  ###### 199 ######  ###### 2CC ######
#################  #################  #################  #################  #################
##    | | |    ##  ##    |A| |    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##    |A| |    ##  ##    | |V|    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
##    | | |    ##  ##    | | |    ##  ##    | |V|    ##  ##    | | |    ##  ##    | | |    ##
##   O| | |O   ##  ##   O| | |O   ##  ##   O| | |O   ##  ##   C| |V|O   ##  ##   O|A| |O   ##
##____+ .V+____##  ##____+ . +____##  ##____+ . +____##  ##____+A. +____##  ##____+ .V+____##
##  > > V>>>  >##  ## > > > V>>>  ##  ##  > >A>  >>> ##  ## > > > >  >>>##  ##  > >A> >  >>##
##----.   .----##  ##----.A V.----##  ##----.  <.----##  ##----.A  .----##  ##----.   .----##
##<< << AV< <  ##  ##< <<   V < < ##  ## <<  A V< <  ##  ##<<    << < < ##  ##<    << < <  ##
##____+A. +____##  ##____+A.V+____##  ##____+ .V+____##  ##____+ .V+____##  ##____+A. +____##
##   O| | |O   ##  ##   O| | |O   ##  ##   O| |V|O   ##  ##   C|A|V|O   ##  ##   O| |V|O   ##
##    | | |    ##  ##    | | |    ##  ##    |A| |    ##  ##    | |V|    ##  ##    | |V|    ##
##    | |V|    ##  ##    |A| |    ##  ##    | | |    ##  ##    | | |    ##  ##    | |V|    ##
##    | | |    ##  ##    | |V|    ##  ##    | | |    ##  ##    | | |    ##  ##    | | |    ##
#################  #################  #################  #################  #################
#################  #################  #################  #################  #################
```

```
###### 201 ######
#################
##    | | |    ##
##    | |V|    ##
##    |A| |    ##
##   O| | |O   ##
##____+A. +____##
## > > >>V >  >##
##----.   .----##
##    <A<< < < ##
##____+ . +____##
##   O| | |O   ##
##    | |V|    ##
##    |A|V|    ##
##    | |V|    ##
#################
#################
```

```
THE VALUE OF THE CLOCK AT THE END OF THE SIMULATION IS  201


                            QUEUE STATS

                    AV. LENGTH  AV. WAIT TIME   MAX      USERS   LENGTH AT END
Q1                     0.00         0.00          1        49          0
Q2                     0.00         0.00          1        98          0
Q3                     0.00         0.00          1        49          0
Q4                     0.00         0.00          1        98          0


                     FACILITY STATS

                     USERS     AVERAGE USE  AV.TIME SPENT  USERS AT END
F1                     98         0.49         1.00           0
F2                    133         0.66         1.00           1
F3                     96         0.64         1.33           1
F4                    134         0.67         1.00           1
F5                     46         0.39         1.70           0
F6                     10         0.12         2.20           0
F7                     53         0.43         1.62           0
F8                    138         0.69         1.01           1
F9                     97         0.70         1.43           1
F10                   140         0.71         1.01           1
F11                    98         0.49         1.00           0
F12                    49         0.28         1.14           0
F13                    49         0.31         1.27           0


                          TABLES

                    NUMBER       MAX        MIN      AVERAGE STANDARD DEVIATION
NORTH TO SOUTH        32          17         13       13.19        0.77
NORTH TO EAST          5          11         10       10.40        0.49
NORTH TO WEST          9          20         13       15.11        2.51
SOUTH TO NORTH        28          15         13       13.07        0.37
SOUTH TO EAST          9          15         13       13.22        0.63
SOUTH TO WEST          9          11         10       10.33        0.47
EAST TO NORTH          6          15         13       14.67        0.75
EAST TO SOUTH          8          10         10       10.00        0.00
EAST TO WEST          80          13         12       12.51        0.50
WEST TO NORTH         10          10         10       10.00        0.00
WEST TO SOUTH          9          15         13       13.67        0.94
WEST TO EAST           75         13         12         12.41          0.49

123.34 SECONDS IN EXECUTICN
/8                                                                                 09.50.41
```

```
********************     ST.ANDREWS UNIVERSITY COMPUTING LABORATORY - 44MFT/SPCCLER ENC-CF-JCB RECCRC     **********
******************** JOB NAME : GHSIM      DATE : 75276      START TIME : C9.47.15   ELAPSED : CO.C3.28 **********
********************     CPU TIME : 00.02.30     WAIT TIME : 00.00.25     SYSTEM OVERHEAD : 00.00.19     **********
********************     PAGES PRINTED :     57     CARDS PUNCHED :       0     CARDS READ :  1107     **********
********************     A GRAND TOTAL OF          4027  INPUT/OUTPUT REQUESTS WERE HANDLED FOR THE JCB.     **********
********************     THE AVAILABLE CORE WAS : 200 K      OF WHICH AT LEAST    89 K WAS USED     **********
********************                 THE APPROXIMATE COST OF THIS JCB WAS  £  1.90     **********
********************              THIS JOB WAS EXECUTED IN THE BACKGROUND PARTITICN     **********
* THIS LINE PRINTED AT 09.59.37  ;
```

References

1. "Multiprogramming System for 360 Model 44", IBM Corporation S/360-61 CPL 360 D-05.2.012.
2. "Scientific Subroutine Package System /360", IBM Corporation, GH20-0205.4.
3. "Control and Simulation Language Reference Manual", Esso Petroleum Co. Ltd. and IBM Corporation United Kingdom Ltd., 1963.
4. "MILITRAN Programming Manual", Prepared for the Office of Naval Research, Washington by Systems Research Group, Inc., New York, 1964.
5. "General Purpose Simulation System/360: Introductory User's Manual", IBM Corporation 1967.
6. "Introduction to Algol-W Programming", University of St. Andrews Technical Report No. CS/72/3, 1972.
7. "Algol-W Language Reference Manual", University of St. Andrews Technical Report No. CL/72/8, 1973.
8. "PL360 Programmers' Reference Manual", University of St. Andrews Technical Report No. CL/72/6, 1974.
9. Buxton, J.N. and Laski, J.G., "Control and Simulation Language", Computer Journal, 5, 3:194-199, 1962.
10. Dahl, O.J. and Nygaard, K., "SIMULA, A language for Programming and Description of Discrete Event Systems: Introduction and User's Manual", Norwegian Computer Centre, 1965
11. Hollingdale, S.H., "Digital Simulation", A conference under the aegis of the Scientific Affairs Division of N.A.T.O., The English Universities Press, 1967.

12. Knuth, D.E., and McNeley, J.L., "SOL - A symbolic language for General-Purpose Systems Simulation", IEEE Transactions on Electronic Computation, Vol. EC-13, No. 4 August 1964.

13. Knuth, D.E., "The Art of Computer Programming", Vol. 3, Addison-Wesley Publishing Company, Inc., 1973 (p 151-152).

14. Markowitz, H., Hausner, B. and Karr, H., "SIMSCRIPT: A Simulation Programming Language", Prentice-Hall, Inc., New Jersey, 1963.

15. Pritsker, A.A.B. and Kiviat, P.J., "Simulation with GASP II: A FORTRAN based Simulation Language", Prentice-Hall, 1969.

16. Reitman, Julian, "Computer Simulation Applications", John Wiley and Sons, 1971.

17. Shearn, D.C.S., "Discrete Event Simulation in ALGOL-68", Software Practice and Experience 1975, Vol. 5, No. 3, p 279.

18. Stravrakis, C., "A Comparison of Certain Simulation Systems", A thesis presented for the degree of Master of Science, University of St. Andrews, 1973.

19. Tauseworth, Robert C., "Random numbers generated by linear recurrence modulo-two", Mathematics of Computation 19, 1965.

20. Tocher, K.D. and Hopkins, D.A., "Handbook of the General Simulation Program", Department of O.R. and Cybernetics, United Steel Companies Ltd., Sheffield.

21. Tocher, K.D., "Review of Simulation Languages", Operational Research Quarterly, Vol 16, No. 2, 1965 (p 189-217).

22. Whittlesay, J.R.B., "A Comparison of the Correlational Behavior of Random Number Generators for the IBM 360", Communications of the ACM 11, 9, September 1968.