AN APPROACH TO TEACHING COMPUTER SIMULATION

By
S.O'BROIN, B.Sc., M.Sc.

A doctoral thesis submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of the Loughborough University of Technology, 1985

Supervisors : Professor A.C. Bajpai
Director of CAMET and Head of Department

and Dr. H.W.F. English
Department of Engineering Mathematics
Loughborough University of Technology

ABSTRACT

AN APPROACH TO TEACHING COMPUTER SIMULATION

BY S.O'Broin

CAMET

(Centre for Advancement of Mathematical Education in Technology)

Loughborough University of Technology

The thesis proposes a method for teaching computer simulation. The method involves the use of a large-scale real-life project which had been carried out by the author on a consultancy basis. This project has been developed into a teaching package which is intended for a wide spectrum of students, thus little knowledge of mathematics is assumed. This method has been found to be the most successful one by the author in over twenty years of experience with different types of students and this particular package has been tried successfully with a group of students in Hong Kong.

The real system and the relevant problems involved are first described and considered. From this an extremely simplified version is then extracted. Solution methods are considered and the usefulness of simulation demonstrated. The tools required for a simulation are then introduced, the simulation language used being GPSS which is considered by the author to be the most appropriate on the basis of ease of learning and future usefulness. In a series of steps realistic complications are added to this simple system while in parallel the required extra elements of the simulation language are introduced building up to the final simulation of the real system by the students who will by then have a thorough grasp of GPSS. They will also have had the experience of developing a full-scale simulation model. At this stage the students will be in a position to apply their knowledge and experience to problems in other fields and the author hopes that the lecturer will be motivated to develop similar projects for teaching in other areas.

Key words : Education, GPSS, Simulation, Operations Research, Case-Study, Computers, Transport, Queueing

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Professor A.C. Bajpai and Dr. H. English for their help and encouragement and their patience in the tedious task of reading and rereading successive drafts and for their numerous suggestions for improvements. I would like to thank my head of department Dr. I. Tang for his encouragement and understanding particularly inview of the time which I had to devote to work on the thesis and Hong Kong Polytechnic for generously granting study leave. Thanks are also due to our typists Fanny and Betty for their patience in typing and retyping so many times.

Finally I would like to thank my wife, Anne without whose encouragement I might not have embarked on this project in the first place.

CONTENTS

APPENDICES

CHAPTER 1

INTRODUCTION

1.1 SIMULATION

1.1.1 Definition of Simulation

There are several possible definitions. A list of definitions is given by Pritsker (1979). One useful one is that of Shannon (1975) who defines it as "the process of designing a model of a real system and conducting experiments with this model for the purpose either of understanding the behaviour of the system or of evaluating various strategies for the operation of the system." However it is not necessarily clear what is meant by the terms "real system" and "model". One starts with a system which is the subject of investigation. This system need not be "real" in the sense of physically existing but at least a description of it must exist. Let us call this the "original system". Suppose that it was not convenient to experiment with this original system then one might construct a secondary system the operation of which mimiced the operation of the original system but which is easier to manipulate. Provided that the secondary system mimics the original system closely enough it can be called a model of the original system. The procedure involved in constructing and experimenting on this secondary system is called SIMULATION. In computer simulation this secondary system or model is a computer program or programs. In this thesis simulation will always mean "Computer Simulation for which the following definition is proposed".

Definition :

A Computer Simulation is the process of designing and operating a computer model of some system, which behaves in a way analogous to the operation of the original system. The purpose of such a computer simulation is to acquire an understanding of the behaviour of the original system under various conditions and/or to evaluate the effect of certain strategies on the operation of the system usually with a view to selecting the optimum strategy.

A "Simulationist" is to be understood to be a person who uses simulation as defined above for any of the stated purposes. The purpose of the teaching package will be to impart the necessary skills and some experience.

It should be noted that a simulation can be undertaken for either of two purposes. An economist might simulate an economic system in order to experiment with it and to acquire an understanding of how it works. This type of simulation can be used in order to teach a subject such as economics. It is not strictly necessary that the student actually writes the simulation although it would be very useful if he could do so or at least manipulate it.

An Operations Research analyst (or indeed anyone else) might write a simulation in order to solve a specific problem such as the study of Carroll and Laurin (1982) who wanted to assign police patrol zones in an efficient manner. Students who may use simulation as a tool in this way must of course be able to write their own simulations.

1.1.2 Reasons for Teaching Simulation

Simulation is a technique which is useful in almost all disciplines. The list of applications on the following page includes examples from such diverse fields as Employment, Population Control, Police Work, Management, Health Care, Real Estate, Law, Journalism and Probability. Bork (1979) lists four reasons for teaching it.

1. It is an important research tool. A survey of young O.R. workers carried out by J.E. Beasley and G. Whitchurch (1984) revealed that 31% of respondents frequently used simulation - this was much higher than for instance linear programming (5%)
2. Simulations have a strong dramatic component which capture the imagination of the individual and thus motivate one to learn about the system which is being simulated.
3. Simulations provide students with experiences that may be impossible or difficult to obtain in everyday life e.g. to experiment with the economic system.
4. The experience of simulating develops insight and intuition.

An additional reason is that simulation draws from many fields and integrates diverse knowledge and skills. Thus if an economic system is being simulated the student uses and learns economics, probability, statistics, computer language and skills, modelling in addition to simulation per se. Of course the extent to which these topics are or can be utilised will depend on the students' background but at the very least he will acquire skill in modelling, computers and simulation.

1.1.3 Examples of Applications

To indicate the wide applicability of simulation a number of recent diverse applications are listed below

1. Can shorter Workweek Reduce Unemployment?

   J. Henize (1981)

2. Mathematical Simulation of Impact of Birth Control Policies on Indian Population System

   Patil, Janahanlal, Ghista (1983)

3. Using Simulation to Assign Police Patrol Zones

   Carroll and Laurin (1982)

4. Using Computer Simulation to Develop Optimal Inventory Policies

   Gaither (1983)

5. Implementation of Computer Simulation Projects in Health Care

   Tunnicliffe Wilson (1981)

6. A Simulation Approach to Real Estate Investment Analysis

   Markland (1979)

7. Making Better Use of Jurors Through Computer Simulation

   Greilich (1979)

8. A Network Model and Simulation Analysis of a Journal Review Process

   Moore, Taylor and Cattanach (1980)

9. Using Simulation to Resolve Probability Paradoxes

   Reinhardt and Loftsgaarden (1978)

1.2. EXISTING TEACHING METHODS

1.2.1 Traditional Method

The traditional method of teaching simulation is to start with a discussion of modelling in general, then develop the necessary statistics, then consider simulation in general terms and finally introduce a special simulation language which can be used for some small case-studies at the end of the course. For instance in the textbook of McMillan and Gonzalez (1973) GPSS is first introduced in Chapter 12 of an eighteen chapter book. In Naylor, Balintfy, Burdick and Chu's textbook (1968) GPSS is introduced in Chapter 7 of nine chapters. Gordon (1978) introduces GPSS in Chapter 9 of fourteen chapters although he does introduce some other languages as early as chapter four. Maisel and Gnugnoli (1972) introduce GPSS in chapter 9 of their fifteen chapter book.

The author agrees with Bronson (1978) when he states

> "the traditional approach to teaching simulation (lectures followed by a series of short assignments due in a couple of weeks) does not produce simulationists; it produces appreciative audiences for simulation"

and also

> "courses are particularly deficient when students are asked only to simulate a few small problems".

Virtually all simulation textbooks assume some background in mathematics, statistics, probability and computing, in particular Naylor et.al (1968), Shannon (1975) Fishman (1978). According to Shannon (1975)

> "one of the most significant difficulties in gaining acceptance of quantitative methods by managers is semantics".

The author feels that it is not so much the semantics but the mathematical notation which imposes an almost insuperable barrier for many.

It is also the author's opinion that many O.R. lecturers do not give simulation the attention it deserves. They prefer to teach more structured subjects such as linear programming. In most O.R. courses more attention is devoted to linear programming then to simulation although in practice simulation is used more often than linear programming. This is borne out in a survey of young O.R. workers carried out by Beasley and Whitchurch (1984) and in a survey of marketing managers by Wensley (1979) (see also Higgins 1982). Below are extracts from two of Beasley and Whitchurch's tables.

Coverage of Technique

Technique	Too Much (%)	O.K. (%)	Inadequate (%)
Linear Programming	41	56	0
Simulation	6	56	38

The relative usefulness of simulation is indicated in the following table

Use of Techniques

Technique	Frequently Used (%)	Sometimes Used (%)	Never Used (%)
Linear Programming	5%	21%	69%
Simulation	31%	47%	20%

Wensley found that 76% of marketing managers never used linear programming but 67% used marketing macro models particularly simulation.

Totals do not always add to 100% because some of those surveyed did not reply to particular questions.

1.2.2 Defects of Traditional Method

The author suggests that traditional teaching of simulation has the following defects

1. The special language (e.g GPSS) is not introduced early enough. As the choice of language influences how a model of a system is formulated the two can and should be taught simultaneously. Also leaving the special language to the end of the course constrains the size of case-study which can be undertaken.

2. Case-studies are usually relatively trivial. This is related to 1 above.

3. The dependence on mathematics and statistics is a deterrent for many potential users.

4. Many lecturers lack motivation to teach simulation preferring topics such as linear programming or queueing theory.

The isolation of the above defects is based on the author's personal experience. However the results of the questionnaire's quoted on page 6 support 4 above, the remarks ascribed to Fishman on page 10 support 1 above and Bronson's comments quoted on page 5 support 2.

1.3 METHOD ADVOCATED BY AUTHOR

1.3.1 Background

A possible method is that proposed by Bronson (1978) which involves dividing a class into teams of two to three people. Each team selects, from a list provided, a topic for a term-project. Thus all are involved in relatively large-scale simulations of non-trivial systems. The topics would have to be carefully selected to correspond to the students' backgrounds. The method basically is very good but would in practise be difficult to implement. In Hong Kong where the relevant class size might be say fifty students one would need about twenty large scale simulations going on simultaneously. Apart from the difficulties involved in obtaining such a large supply of suitable topics the supervisory load would be enormous, and would be completely impractical for evening students who are supervised by part-time lecturers and have limited access to computing facilities. Consequently the author suggests a different approach, a single large project developed into a teaching package which can be used by any lecturer rather than a number of individual projects. Presumably the method advocated by Bronson would involve teaching simulation before the students embark on the project. The author is suggesting that the two be done in parallel.

The method being proposed is based on about 20 years of teaching experience by the author plus about 10 years of industrial experience (these partly overlap.) After 5 years as a senior O.R. analyst with Aer Lingus the author was engaged full-time in a management school in Dublin for five years and became increasingly aware of the difficulties experienced by managers in trying to use quantitative techniques. Some of these difficulties are also shared by business students such as those at Hong Kong Polytechnic where the author has been employed for over eight years. The difficulties mainly relate to mathematical notation and method of presentation by the lecturer. In particular mature students who have been away from mathematics for a number of years are put off by the use of calculus, such symbols as d/dx or $\int$ or even a summation sign apparently immediately evoking a negative reaction perhaps causing them to dismiss the entire article in which they appear as irrelevant. Such symbols are avoided in the proposed teaching package except in optional appendices.

While engaged on a consultancy project in Hong Kong it occurred to the author that this particular project would provide an interesting and useful basis for teaching simulation. The project was to investigate the handling capacity of a multi-storey warehouse (known as a "godown" in Hong Kong)

1.3.2 Procedure Advocated

The procedure advocated involves starting with a very simple original system. In the godown example this simple system would consist of random arrivals at a single service centre with no traffic complications. This system can be modelled easily and computerised using a few special instructions. Then in a series of steps realistic complications are added to this simple system while in parallel the required extra elements of the chosen simulation language are introduced building up to the final simulation of the real system by the students under the supervision of the lecturer. This parallel development of modelling and programming has been done successfully by others. Fishman (1978) states: "The change from a series to parallel presentation aims to shorten the time it takes to enable a student to begin building, programming and analysing a simulation. Experience at the University of North Carolina at Chapel Hill confirms that students derive considerably more satisfaction from the parallel presentation than from the earlier series presentation. They also appear to learn more".

1.3.3 Choice of Simulation Language

Simulation languages can be divided into two categories, continuous simulation languages such as DYNAMO or CSMP and discrete simulation languages such as GPSS or SIMSCRIPT. Recently attempts have been made to devise languages such as SLAM (see Pritsker and Pegden 1979) which can do either discrete or continuous simulation. Since the godown example is a discrete simulation the author restricted his attention to discrete simulation languages. These can further be subdivided into "event scheduling" languages such as GASP or SIMSCRIPT and "process interaction" languages such as GPSS or SIMULA. For a comparison see Fishman (1978). Most people find "process interaction" languages easier to use and what the author was seeking was a language which could be easily learned. Gordon (1978) states: "GPSS has been written specifically for users with little or no programming experience while SIMSCRIPT, along with many other simulation languages requires programming skill to the level where the user is able to program in FORTRAN or ALGOL". McMillan and Gonzalez (1973) state: "The best known, most easily learned, and most frequently used language is GPSS." More recently Reitman (1982) states: "GPSS still has many advocates who particularly stress the small amount of effort required to model a complex system". Since GPSS was available in Hong Kong it became the obvious choice.

GPSS stands for General Purpose Simulation System. It was originally developed by IBM circa 1961 and has evolved through several versions. The current IBM versions are GPSS/360 and GPSS V the latter being more powerful. A recent enhanced version of GPSS V called GPSS/H is described by Reitman (1982). As the author was using a UNIVAC 1100 computer the version used was GPSS 1100.

The author is not claiming that GPSS is the best simulation language but its properties, ease of learning and wide availability make it particularly suitable. In practice which language is better depends on the application in question. Atkins (1980) compares GPSS and SIMULA for simulating sparse traffic (a similar problem to the godown one) and in fact concludes that SIMULA is better.

Of increasing importance will be the availability of a language on micro-computers. So far the languages available on micros include versions of DYNAMO for Apple II and SIMSCRIPT and GPSS for IBM-PC. The version of GPSS available for IBM-PC compatible machines reportedly has a new interactive design suitable for keyboard modification. It is not widely available but has been commercially advertised in the U.S.A. However the ability of micro-computers to handle large-scale simulations is in doubt. Even on a computer such as the UNIVAC 1100 there were problems with size and speed. The program of Unit 9 required thirty minutes to simulate fifty days. A similar simulation written in FORTRAN (which should be a more efficient user of computer time) required 24 hours of computer time on an APPLE to simulate eight hours (one working day) of system operation!

1.3.4 Prerequisite Knowledge

As previously mentioned a considerable difficulty in teaching simulation to non-specialists is the background knowledge required of such subjects as mathematics, probability, statistics, computers, a computer language, systems analysis and a technical knowledge of the system being simulated. The author has attempted to devise a procedure which assumes a minimum of previous knowledge of those subjects. They will now be considered individually.

1.3.4.1 Mathematics

Mathematics and in particular mathematical notation is one of the greatest deterrents for prospective users of simulation especially these who have done only school mathematics and that many years previously. Yet some mathematical analysis is necessary. To minimise the effect of the mathematics barrier the author suggests the following :

(a) Use a special simulation language such as GPSS. This has the advantage that many mathematical formulae and calculations are built-in in such a way that the user need not even be aware of them.

(b) Where some mathematical analysis or technique is useful in illustrating a point, whilst not being strictly necessary, it should be relegated to an optional appendix for use by more mathematically able students.

(c) By making the material so apparently relevant and interesting that a user is motivated to surmount what "small" mathematical barriers still remain (hopefully they would not be perceived as barriers).

1.3.4.2 Probability

Some discussion of probability is inevitable as it is the basis-of almost all simulations. Very often the reason why one simulates a situation is that given any proposed solution the consequences are not completely determined. They will depend on the solution selected of course, but also on the operation of chance and it is here (i.e. dealing with "chance") that probability theory is required. In a queueing/traffic type model chance affects time durations, specifically the time between successive arrivals, the time to perform a certain service, the time required to drive from point A to point B. To a lesser extent it may affect the opening or closing time of a system or the length of a meal-break. The occurrence of a breakdown is also subject to chance. The only theoretical distribution required in the models to be discussed is the negative exponential distribution (in connection with inter-arrival times) but this is built into GPSS so the formulae are not required. All of the other distributions will be empirical. Unit 1 of the 9 unit teaching package deals with the elementary probability concepts which will be required subsequently including the negative exponential distribution. The use of calculus, set-theory, limits and series is avoided. Examples relate to what is to come later.

1.3.4.3 Statistics

Many potential users of simulation will not have a knowledge of statistics. Even if they have sufficient mathematical knowledge to learn the required statistics it would hardly be feasible within the normal time constraints of a simulation course to develop from scratch sufficient statistics to be of use. One therefore has two alternatives assume a prior knowledge of statistics or try to manage without statistics. Many would argue with some justification that a considerable knowledge of statistics is required in order to understand and even more so to validate, the output produced by a simulation. Two problems can be distinguished here, as follows:

(a) To ensure that the model does indeed behave in a way which is analogous to the original system i.e. that given the same input the original system and the model would both produce the same output or at least that important outputs do not differ by more than some acceptable amount. If the original system exists then a direct comparison can be made. Ideally statistical tests could be used but people make comparisons all of the time on a common-sense basis without the benefit of statistics. If the original system does not exist statistics is of little help and one must rely on ones technical knowledge of the system and skill in modelling.

(b) To verify that samples obtained from the operating model are sufficiently large and taken in such a manner that they provide the required information with sufficient accuracy. This is not specifically a problem of simulation since it relates to experiments on the original system just as much as to experiments on the model. Undoubtedly in this area a lack of statistical knowledge is a severe disadvantage but just as a lack of statistical knowledge does not deter people from taking samples in the real world it should not deter them from taking samples from a simulation. One should however be always aware that, as stated by Law(1983) "a simulation is a computer-based statistical sampling experiment". The output should be treated accordingly.

The comments in the preceding paragraph refer mainly to industrial applications where precision of estimation may not be important. There are other applications particularly involving the testing or validation of a theory where absolute precision is required. In such cases, statistics is required and fairly complex statistics at that. Students who require such an end-product should read Appendix C. If that is not sufficient they can be referred to a textbook such as SYSTEMS SIMULATION (Shannon, 1975), or the paper by Law (1983).

The preceding discussion assumed that one wanted to estimate some characteristic (e.g. percentage of items suffering undue delays) with a certain accuracy. However in many simulations one may require only a general idea of how a system will operate. The operation of the simulation model may show that the system will not work and may show this without any statistical analysis. The procedure followed in constructing the simulation model will generally give one considerable insight into the operation of the original system, again without requiring any statistics. In any event what most end users want to see is, say, an example of one month's operation of the system and in particular what problems occurred. They usually do not want ninety-five per-cent confidence intervals for some quantity such as average queueing time. In many practical applications there seems little point in endless applications of statistics to show that the coverage of a confidence interval may only be ninety per-cent and not ninety-five percent (see Law, 1983). Because of the fact that simulation output data for queueing systems is autocorrelated (as would data obtained from the real system be) analysis is difficult. The situation is rather like that described by S. Dunn (1981) in relation to research in Mathematics education when he states "the more it (statistics) develops the more cumbersome it becomes. Each technical step forward brings with it such a plethora of unresolved or consequent side-issues or side-effects that uncertainty is increased rather than resolved". An important point to bear in mind is that even if after considerable analysis one obtains a ninety-five per-cent confidence interval for a quantity it is only for the model. Rarely is the model identical with the real system so one cannot conclude that it is a ninety-five per-cent interval for the same quantity in the original system.

1.3.4.4 Computer Experience

Obviously any computer language can be more easily learned by a person who has some previous knowledge of computers and who is familiar with the jargon. GPSS can however learned by someone with no previous computer experience. In the proposed package it is assumed that all a student has to do is to write the program i.e. someone else will look after job control instructions and the actual running of the program. This is what would normally happen. If the student is familiar with computers he may input the program himself via a VDU and run it. There is no discussion in the package of "error messages" it being assumed that anyone running a program will have a list of these available. In most cases the "error messages" are self-explanatory.

Ideally students should have done a first course in programming in say BASIC so that they are familiar with the general idea and form of computer instructions and computer output. Nowadays most students who have completed first year at third level would have this knowledge. Although it is conceded that a previous knowledge of a language is an advantage it can also cause some difficulties in understanding GPSS. Instructions in BASIC (and most languages) are obeyed sequentially whereas in GPSS they are not. Thus one should not be too disturbed about a lack of previous programming experience. It has not been assumed in the package.

1.3.4.5 Systems Analysis

Any simulation involves systems analysis since in order to design a computer model of some system it is necessary to analyse the system. In fact McMillan and Gonzalez (1973) in their book on Systems Analysis treat simulation and systems analysis as being virtually the same thing. Shannon's definition of simulation (see page 1) could also be taken as a definition of systems analysis. However in the definition of Computer Simulation proposed on page 2 by the author they are not identical, as a systems analysis does not necessarily lead to a computer model. In fact a computer system devised by the systems analyst may not be a model of the original system, it may contain new elements (improvements) and the computer system may in fact replace the original system.

It follows that a previous knowledge of systems analysis is invaluable to a student of simulation but unfortunately cannot and should not be assumed. For the "original system" chosen (the multi-storey godown) no particular knowledge of systems analysis is required. For a more complex original system it might.

1.3.4.6 Technical Knowledge of System Being Simulated

The original system chosen (the multi-storey godown) was deliberately selected as being one which only involves concepts with which most people are familiar. It would not have been possible to select say an economic system or an ecological system as these would require previous technical knowledge. Of course such systems could be used with students of those particular disciplines, thus a group of students studying biology could develop a simulation of an ecological system in parallel with their biological studies.

1.4 SUMMARY

1.4.1 Proposal

Traditional methods for teaching simulation suffer from the following defects:

1. The special language is not introduced early enough
2. Case-studies are usually relatively trivial
3. There is too much dependence on mathematics/statistics
4. Lecturers are not sufficiently motivated to devote sufficient time to simulation.

The author has proposed a teaching method which attempts to overcome these difficulties as follows

1. GPSS is introduced almost at the beginning
2. A single large-scale case-study is used
3. Mathematics and statistics are avoided
4. A complete package suitable for use by any lecturer is developed.

1.4.2 Objectives

The objectives of the method are as follows :

1. A student should find the package interesting enough to persevere to the desired end and should not be deterred by a lack of knowledge of, or a liking for, mathematics.

2. A student who completes the package should have an understanding of probability-based modelling as exemplified by discrete-time simulation.

3. He should appreciate the value of a special simulation language such as GPSS and be able to write his own GPSS programs.

4. He should have had sufficient experience of handling successively more complex models to enable him to construct his own simulation models for different situations using GPSS. He should be able to interpret the output produced.

The main difficulties which one can expect to encounter in trying to achieve these objectives are as follows :

1. Trying to teach students to model realistic systems without reliance on mathematics/statistics.

2. Making the output meaningful without the use of statistical testing.

3. Trying to avoid making the reading material dull, particularly difficult when one is trying to teach a computer language.

4. Trying to avoid leading the students too much so that they are forced to use some initiative themselves and thus gain more useful experience.

CHAPTER 2

PREFACE TO THE TEACHING PACKAGE

2.1 THE PROBLEM

A company in Hong Kong had undertaken to build a multi-storey warehouse, which in Hong Kong is known as a godown. The design of the building had been completed (see Appendix F for further details and sketches). They required information on what arrival rates the system could handle without queues becoming too long or alternatively for given arrival rates what queue sizes one could expect.

The relevant part of the building consisted of the following

1. A ground floor where vehicles (lorries and containers) entered through a check-in gate where some clerical formalities took place. No parking (other than while waiting for permission to ascend) took place on this floor.

2. Eight floors numbered one to eight above the ground level. On each of these floors there were eight bays where vehicles could be loaded or unloaded.

3. A roof area ("ninth floor") where vehicles could be parked temporarily.

4. A system of ramps connecting the floors which vehicles would travel up and down between floors.

There were certain restrictions on traffic progression imposed by the design of the building particularly the ramps. Each ramp was divided into two lanes, one for ascending and one for descending traffic. There would not be space for two containers to pass on a ramp. No overtaking could be allowed on ramps. A vehicle going from the ground to floor 3 say would first drive up the ramp to floor one, it would then cross an "apron" area (also known as a forecourt) to the ramp up to floor 2. Two containers proceeding in opposite directions could pass on this apron area but no overtaking by vehicles travelling in the same direction is allowed. There are similar aprons on all floors.

In order to obtain the required information it is necessary to know more about the system operation, in particular the following.

1. The arrival rates of containers and lorries

2. The times taken to perform each manoeuvre within the building (driving up ramps, parking, unloading etc.)

3. The operating hours of the system i.e. will it operate continuously on a 24-hour basis or will it open at a certain time and close at a certain time each day.

2.2 METHODS OF SOLUTION

Assuming that all of the required input information is available how can the required output information be obtained before the system is built? Four possible ways are now considered.

1. Direct mathematical calculations

This could only be done if everything were deterministic i.e. if one knew in advance the exact time at which each vehicle would arrive and the exact time required for each operation. However it is certain that the individual arrival times will not be known i.e. exact individual timetabling is not feasible. It would not be acceptable to users.

2. Mathematical calculations which take account of variation in arrivals

This is what is done in Queueing Theory. Unfortunately such theory can only be applied to relatively simple queueing systems which confirm to certain patterns and this is not one of them.

3. Observation of a similar system

This might be feasible in some cases if a sufficiently "similar" system could be found. However no such system existed in Hong Kong and it is unlikely in general that for any complex system another sufficiently similar one exists.

4. Construct a scale model of the system and carry out experiments on it

This is not a bad idea. However it would need to be an extremely complex model if it were to adequately mimic the original system and it would not be easy to collect data on parts of the system over a sufficiently long period of operation. Also it is relatively inflexible. To try out different configurations the model would have to be physically altered.

From a consideration of the previous discussion it can be seen that once mathematical analysis is ruled out one is left with observation. This must be either on a similar existing system or on a model. However a model as described (i.e. a physical scale-model) is awkward to manipulate. Does one need to actually construct the scale model or would it be possible to carry out on paper the operations that would be performed by the scale model or indeed by the real system i.e. simulate the system? In principle one could although it can be readily appreciated that it would involve many hours (indeed days) of tedious calculations to follow the progress of several hundred vehicles through the system and this would only be one days operation under one set of conditions. This is where the computer can help, i.e. one could carry out a computer simulation of the system. Thus one has a fifth method.

5. Construct a computer model of the system and carry out experiments on it.

To demonstrate how this can be done is one of the aims of the following units. The same method can be used to simulate many systems on a computer. To be realistic the arrival pattern in the model must be similar to that which one anticipates in the real system. However actual arrivals will be random. How can one simulate a random pattern? To do this one needs to understand a little about probability theory and this will be covered in the first two units. To actually program the model using a language such as BASIC or FORTRAN would be extremely tedious but fortunately special languages exist to reduce the drudgery. The one to be used is known as GPSS. This will be introduced in Unit 3.

To model any complex system straight off is very difficult. It is much easier to first isolate the important elements and consider a basic system which contains these but which is much simpler than the original system. This basic system can gradually be added to until eventually one has a realistic simulation of the original system. The system just described for the godown could be very briefly summarised as follows: Vehicles arrive, check in, proceed to a bay, unload and leave. This very simple system is considered in Unit 3 of the teaching package, and is then expanded in subsequent units until in Unit 9 the complete system is simulated albeit for simple traffic rules.

2.3 THE PACKAGE

The teaching package which follows consists of nine units. Units 1 and 2 introduce the necessary ideas about probability. The subsequent units (3-9) gradually develop the simple model first described in Unit 3 to the realistic model of Unit 9 while at the same time introducing more and more components of the GPSS language. Each of the following chapters contains an Introduction to a unit and the unit itself. Thus Chapter 3 contains the Introduction to Unit 1 and Unit 1 and so on, Chapter 11 contains the Introduction to Unit 9 and Unit 9 (the final Unit).

CHAPTER 3

UNIT ONE OF TEACHING PACKAGE

3.1 INTRODUCTION TO UNIT 1 OF TEACHING PACKAGE

Any user of simulation must have some notion of what is meant by a "probability distribution" since most simulations deal with stochastic systems but what is generally covered in an elementary statistics course will suffice for most simulations. Some recommended textbooks are listed at the end of this unit. For a student who has time to acquire only the minimum knowledge necessary this unit indicates what needs to be covered. This minimum knowledge includes the following:

Definition of probability

Definition of probability distribution

Definition of cumulative probability

Distinction between empirical and theoretical probability distributions

The negative exponential distribution

In the godown example to be developed in subsequent units "probability" is involved in the allocation of vehicle type and destination within the building. Thus one needs to specify for instance the probability of a vehicle going to floor n. For GPSS probabilities must be specified as cumulative distributions. In the models, arrivals are assumed to be random, independent events and a theoretical probability distribution is used viz. the negative exponential distribution. This distribution is built into GPSS so one does not need to know the formula. However the implications of the assumptions of randomness and independence (see page 41) should be discussed carefully. In what circumstances might such assumptions not be valid? What effect would dispatching procedures have? What effect would traffic conditions have etc.

On completion of this unit a student should be able to define probability (objective or subjective). He should be able by observing some process (e.g. service times at any facility or times between arrivals onto a queue) to estimate a probability distribution and hence compile a cumulative probability table and graph. He should understand and be able to apply the concepts of "randomness" and "independence" as they affect arrivals into a system. He should know in what circumstances the negative exponential distribution applies.

To test students understanding of the material, conventional test exercises on probability can be set such as examples 3.3, 3.4 in the unit or the following:

1. Plot a cumulative frequency polygon for the following data

Service Time t	Relative Frequency f
0 - 20	.05
21 - 40	.22
41 - 60	.34
61 - 80	.25
81 - 100	.14

(Time is in minutes)

2. From the graph in 1 find the cumulative relative frequencies (F) corresponding to values of t = 18, 35, 63.

   Find also the values of t corresponding to cumulative relative frequencies of .10, .40, .50, .63, .87

3. Using the data in 1 above answer the following

   (i) What is the probability of a service time exceeding 50 minutes?

   (ii) What is the probability of a service time of less than 90 minutes?

   (iii) What is the probability of a service time of more than 105 minutes?

   (iv) What is the probability of a service time between 30 and 60 minutes?

4. Are the following events (a) random (b) independent? Give reasons for your answers.

   (i) arrivals of buses at a bus-stop

   (ii) departures from a queue to commence service

   (iii) arrivals of children in a family (i.e. births)

   (iv) arrivals of passengers at an airport (departures)

   (v) arrivals of aircraft at an airport

Practical observation exercises using a stop-watch could also be set such as: Determine the inter-arrival time distribution for arrivals at the service desk in a library. Is the observed distribution similar to a negative exponential distribution? The arrival times should be tabulated as in Table 3.1.

3.2 UNIT ONE - PROBABILITY DISTRIBUTIONS

Most people have an intuitive idea of what is meant by probability and indeed to give a single precise definition is not possible. It means different things in different circumstances. Following are three possible definitions.

3.2.1 Definitions of Probability

3.2.1.1 Objective Probability (relative frequency definition)

If in n trials an event A occurs on m occasions we say that the relative frequency of A is m/n. It is assumed that on each trial it can be decided unambiguously whether A has occurred or not. This fraction $\frac{m}{n}$ is an estimate of the probability of A which we can denote by P(A). The larger n is the better is the approximation.

For example if one observes n=100 vehicles being unloaded and m=10 of them take between 30 and 40 minutes we would say that the observed relative frequency of service times in the range 30 to 40 minutes is $\frac{m}{n} = \frac{10}{100}$ or 0.1. If we assume that this pattern would persist in an even larger number of observations then for this larger number of observations $\frac{m}{n}$ would be approximately .1 i.e. the probability of a service time in the range 30 to 40 minutes is approximately 0.1.

3.2.1.2 Definition based on symmetry (a priori definition)If an experiment must lead to one of n mutually exclusive events which are all equally likely and m of these events can be considered as giving result A then the probability of result A is m/n.

For an example, suppose that a building has n=8 floors and an arriving vehicle proceeds to one only of these floors and all are equally likely. Let A = "result that the destination is above the fifth floor". m=3 of the equally likely events can result in A(i.e. destination floor 6 or 7 or 8). Therefore the probability of A is 3/8.

3.2.1.3 Subjective Probability

The probability of event A is expressed by a number between 0 and 1 (inclusive). The more certain one is of the occurrence of A the higher the number one assigns to the probability. Thus if one is completely certain then one says the probability of A is 1 or P(A) = 1. If one is certain that A cannot occur then P(A) = 0.

Consider for instance the probability of customer X renting space in the proposed godown. The first definition cannot apply since n=1 i.e. there will only be one trial. The second might apply but it might not be reasonable to assume that the two possible events (he rents or he does not rent) are equally likely. Here someone with as much knowledge as possible of all factors must assign a value to the probability i.e. it is a matter of judgement. This person might say that in his opinion there is an eighty percent chance that the customer will rent i.e. the probability is 0.8.

For the models to be described in this text the first definition is the most appropriate. Note that under this definition the probability is simply what we expect the relative frequency to be in a large number of trials. Probability in the text will refer to inter-arrival times, drive times and loading times and these are things which are repeated and hence relative frequency is relevant.

3.2.2 Probability Distributions

If an experiment leads to one of n different mutually exclusive results and the probability of each result is obtained then this set of probabilities (p_1, p_2 . . . p_n) is known as a probability distribution. Note that $p_1 + p_2 + . . . p_n = 1$. This follows from the fact that the sum of the relative frequencies must be one.

Suppose that one wishes to obtain the probability distribution of service times using the above definitions. An experiment consists of measuring a service time. In fact this can lead to an infinite number of results since time is continuous. That is to say even if service time is always between t_1 and t_2 minutes, within this range an infinite number of values for t (the service time) is possible. However if one rounds off times, say to the nearest 5 minutes, then there is only a finite number n of possible (mutually exclusive) results.

Example 3.1

Suppose that service time is never less than 7.5 minutes and never equals or exceeds 37.5 minutes then a measurement of service time must lead to one of the 6 mutually exclusive results 10, 15, 20, 25, 30, 35 minutes (if one assumes that all are rounded to the nearest 5 minutes). The probability of a service time of 10, 15 ... minutes can be estimated by observing the relative frequencies i.e. one would carry out a survey and note how many service times take 10 minutes or 15 minutes or etc. If one made say 50 observations the results might be recorded as follows:

TABLE 3.1

Actual Service-time	Service Time Rounded To Nearest 5 minutes	Frequency Observed	Relative Frequency
7.5 - 12.499	10	5	5/80 = .0625
12.5 - 17.499	15	23	23/80 = .2875
17.5 - 22.499	20	32	32/80 = .4000
22.5 - 27.499	25	12	12/80 = .1500
27.5 - 32.499	30	6	6/80 = .0750
32.5 - 37.499	35	2	2/80 = .0250
	TOTAL	80	1.0000

The observed relative frequencies can be used to estimate the probabilities hence the following table is an estimate of the probability distribution of service times.

TABLE 3.2

Service Time to nearest 5 minutes	Probability
10	.0625
15	.2875
20	.4000
25	.1500
30	.0750
35	.0250

Note that 12.499, 17.499 etc. are used to signify "up to but not including 12.5, 17.5 etc." in Table 3.1

It is often convenient to represent either relative frequency or probability in a cumulative form. Consider the statement "75% (0.75) of service times are less than 25 minutes". This figure 75% (.75) is known as the cumulative relative frequency corresponding to 25 minutes. It is an estimate of the cumulative probability.

Thus the probability distribution can be specified in the following format:

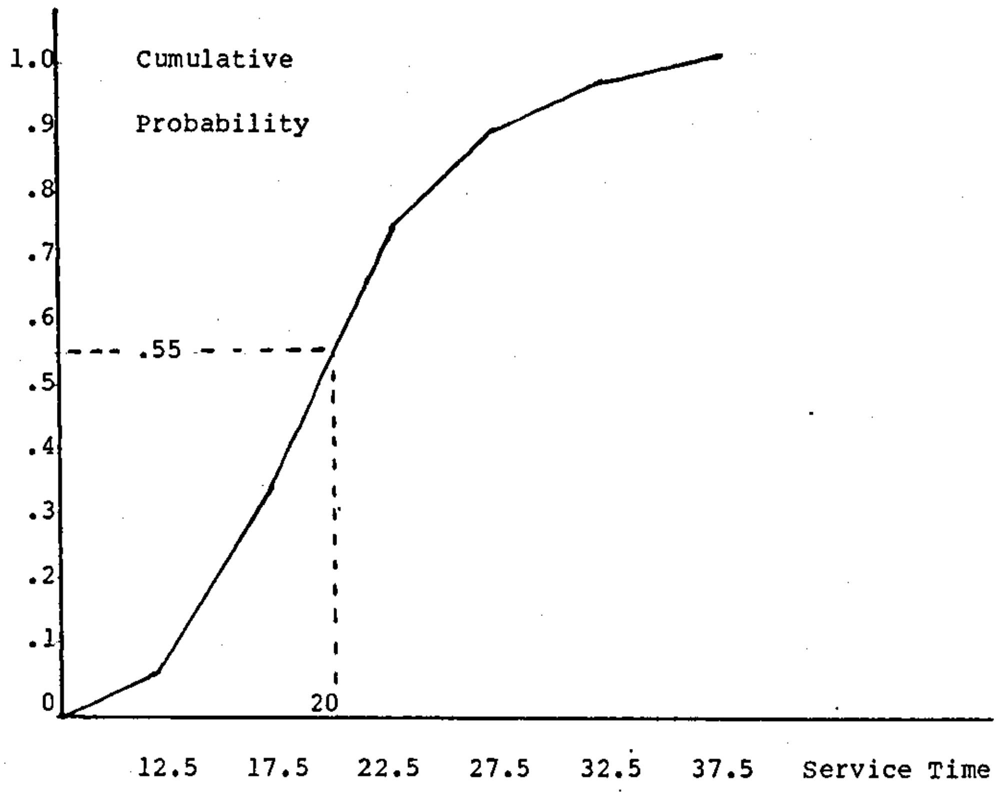
TABLE 3.3

Service Time	Cumulative Probability
12.5	.0625
17.5	.3500
22.5	.7500
27.5	.9000
32.5	.9750
37.5	1.0000

This can be represented graphically as on the next page in Graph 3.1

The graphical form is convenient if one wishes to obtain intermediate values e.g. what is the probability of a service time of less than 20 minutes? This can be read from the graph as indicated below.

GRAPH 3.1



The cumulative probability corresponding to a service-time of 20 minutes is 0.55

The preceding example (3.1) indicated how one might obtain a probability distribution empirically i.e. by observation. Occasionally it is possible by making some assumptions about a process to obtain theoretical values for "the relative frequencies in a large number of trials" i.e. the probabilities. Consider the simple experiment of tossing a coin. If one makes the assumption that the coin is perfectly balanced so that in a large number of tosses the results should not be biassed in favour of either heads or tails then one can conclude that the relative frequency of heads should be 0.5. The following probability distribution is obtained:

TABLE 3.4

Result	Probability
Head	0.5
Tail	0.5

Of course this distribution could also have been obtained empirically by tossing the coin and observing the results. Tossing a single coin is a particularly simple process but the same procedure may be applied in more complex situations to obtain theoretical probability distributions.

Example 3.2

Consider the case of arrivals onto a queue. One might wish to obtain the probability distribution of inter-arrival times i.e. the times between successive arrivals onto the queue. For example suppose that a godown opens at 9.00 a.m. and the first three vehicles arrive at 9.05, 9.06, 9.10 respectively. The first three inter-arrival times are 5, 1 and 4 minutes respectively. Note that the first "inter-arrival" time is measured from the opening time. One could obtain this distribution empirically by observing the queue over a period (say 2 days). The results of the observations might be tabulated as follows:

TABLE 3.5

Time Between Arrivals	Number of Times Observed	Relative Frequency = Probability Estimate
0 up to but not including 5 mins.	10	10/71 = .141
5 up to but not including 10 mins.	25	25/71 = .352
10 up to but not including 15 mins.	15	15/71 = .211
15 up to but not including 20 mins.	12	12/71 = .169
20 up to but not including 25 mins.	4	4/71 = .056
25 up to but not including 30 mins.	4	4/71 = .056
30 up to but not including 35 mins.	1	1/71 = .014
TOTAL	71	.999

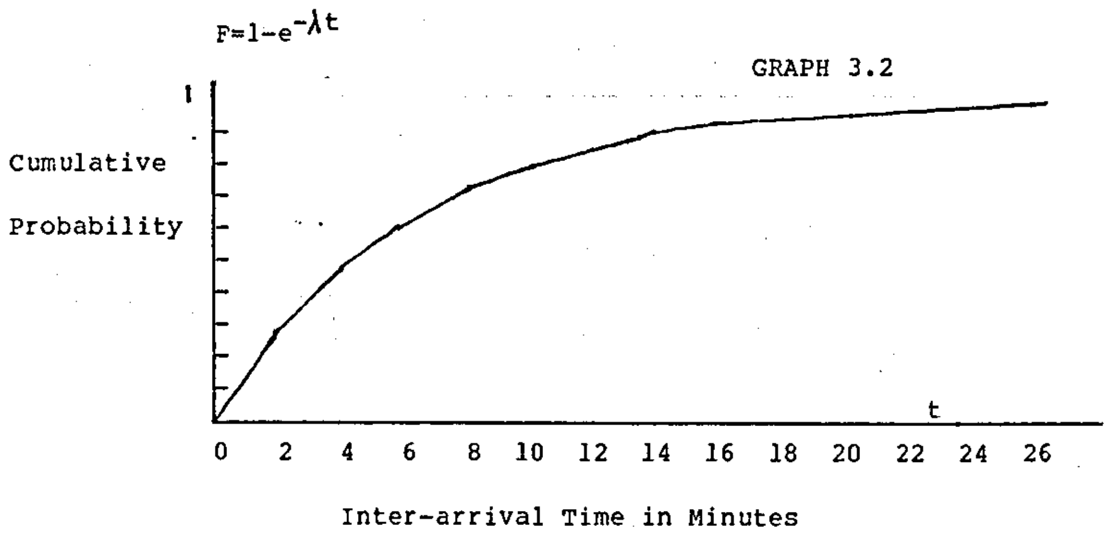
3.2.3 Negative Exponential Distribution

It is however possible to obtain theoretical values for these probabilities if one makes certain assumptions about the process. These assumptions must of course be valid just like the balanced coin in the previous example. The assumptions which one needs to make are as follows:

1. Each arrival onto the queue is independent of all previous arrivals.
2. Arrivals are random i.e. all arrival times are equally likely - there is no scheduling.
3. The average arrival rate is constant

Based on these three assumptions it can be shown that the probability of an inter-arrival time being less than t is $1 - e^{-\lambda t}$ i.e. the cumulative probability corresponding to t is $1 - e^{-\lambda t}$ where λ is the average arrival rate and e is the well-known mathematical constant with a value of approximately 2.7183. This distribution is known as the negative exponential distribution. It is plotted below for λ = 10 per hour

GRAPH 3.2



(Table 3.7 contains an example for λ = 5 per hour approximately)

Example 3.3 (in all cases assume that the three assumptions on the previous page are valid)

(a) What is the probability of less than 2 minutes elapsing before the next arrival if the average arrival rate is 10 per hour?

$$\lambda = 10 \text{ per hour or } 1/6 \text{ per minute}$$

so
$$\lambda t = 1/6 \times 2 = 1/3$$
$$e^{-\lambda t} = e^{-1/3} = .7165$$
$$1 - e^{-\lambda t} = 1 - .7165 = .2835$$

(b) What is the probability of less than 3 minutes elapsing before the next arrival if the average arrival rate is 10 per hour?

$$1 - e^{-3/6} = 1 - .6065 = .3935$$

(c) What is the probability that between 2 and 3 minutes will elapse before the next arrival if the average rate is 10 per hour?

This is obviously the difference between the answers to (a) and (b) above i.e. .3935 - .2835 = .11

Example 3.4

For the data in example 3.2 obtain the average arrival rate and based on this obtain the theoretical "relative frequencies" or probabilities and compare with the observed frequencies.

TABLE 3.6

Time Between Arrivals	Average Time in This range	Number of Times Observed	Total Time
0 up to 5 mins.	2.5	10	2.5 x 10 = 25
5 up to 10 mins.	7.5	25	7.5 x 25 = 187.5
10 up to 15 mins.	12.5	15	12.5 x 15 = 187.5
15 up to 20 mins.	17.5	12	17.5 x 12 = 210.0
20 up to 25 mins.	22.5	4	22.5 x 4 = 90.0
25 up to 30 mins.	27.5	4	27.5 x 4 = 110.0
30 up to 35 mins.	32.5	1	32.5 x 1 = 32.5
	TOTALS	71	842.5

Average time between arrivals = 842.5/71 = 11.866 minutes

Average arrival rate per hour = 60/11.866 = 5.056

Thus λ = 5.056 per hour or .0843 per minute

Probability of an interarrival time of less than 5 minutes (for instance) is $1 - e^{-\lambda t} = 1 - e^{-5(.0843)} = 1 - e^{-.4125}$

$= 1 - .6561$

$= .3439$

In a similar way we calculate cumulative probabilities corresponding to 10, 15, 20 ... and enter in Table 3.7.

TABLE 3.7

Time Between Arrivals	Cumulative Probability
0 minutes	$1 - e^{-0(.843)} = 0$
5 minutes	$1 - e^{-5(.0843)} = .3439$
10 minutes	$1 - e^{-10(.0843)} = .5696$
15 minutes	$1 - e^{-15(.0843)} = .7176$
20 minutes	$1 - e^{-20(.0843)} = .8147$
25 minutes	$1 - e^{-25(.0843)} = .8785$
30 minutes	$1 - e^{-30(.0843)} = .9203$
35 minutes	$1 - e^{-35(.0843)} = .9477$

TABLE 3.8

Time Between Arrivals	Probability	Expected frequency out of 71 observations (Probability x 71)
0 up to 5 minutes	.3439 - 0 = .3439	24.4
5 up to 10 minutes	.5696 - .3439 = .2257	16.0
10 up to 15 minutes	.7176 - .5696 = .1480	10.5
15 up to 20 minutes	.8147 - .7176 = .0971	6.9
20 up to 25 minutes	.8785 - .8147 = .0638	4.5
25 up to 30 minutes	.9203 - .8785 = .0418	3.0
30 up to 35 minutes	.9477 - .9203 = .0274	1.9
35 or more minutes	1.0000 - .9477 = .0523	3.7
		70.9

Table 3.8 has been obtained from Table 3.7 by subtractng successive cumulative probabilities, for instance the probability of an inter-arrival time between 5 and 10 minutes

= (Probability of a time of less than 10 mins.) -
Probability of a time less than 5 mins.)

= .5696 - .3439 = .2257

3.2.4 Comparison of Observed Frequencies and Theoretical Ones

The observed frequencies in Table 3.6 and the expected frequencies (obtained by assuming a negative exponential distribution) in Table 3.8 are compared below in Table 3.9.

TABLE 3.9

Time Between Arrivals	Observed Frequencies (TABLE 3.6)	Expected Frequencies (TABLE 3.8)
0 up to 5	10	24
5 up to 10	25	16
10 up to 15	15	11
15 up to 20	12	7
20 up to 25	4	4
25 up to 30	4	3
30 up to 35	1	2
35 and over	0	4

The agreement between observed and expected frequencies is not good for the first two classes. This would lead one to suspect that in the situation observed the assumptions on which the expected frequencies were based are not valid.

Close agreement between observed and expected values is only to be expected in a very large sample i.e. observation over a long period. For a small sample chance can cause big relative variations. Is it possible to determine whether or not the disagreement observed above could be due to the effects of chance or not? Yes, by using a statistical test called the chi-square test. This is done in Appendix A.

3.2.5 Statistics Bibliography

There are numerous good textbooks on elementary statistics. Those mentioned on the following page constitute only a small sample of what is available. The problem is that they are not written with simulation in mind and a reader who is in a hurry has to wade through a lot of material which, though useful in the long run, may not be immediately relevant. Below is a list of topics roughly in order of importance to users of simulation.

1. Frequency distributions
2. Probability and probability distributions
3. Negative exponential distribution
4. Chi-square test
5. Normal, binomial and Poisson distributions
6. Sampling theory
7. Confidence Intervals
8. Hypothesis testing
9. More theoretical distributions (Gamma, Beta, Erlang etc.)
10. Design of experiments
11. Correlation and regression analysis
12. Time series analysis
13. Analysis of variance

For any simulation of a stochastic system 1 and 2 will be required. 3 will be required if one wants to generate random events such as random arrivals. 4 is useful if one wants to demonstrate that an observed distribution is in fact random. 5 increases ones knowledge of theoretical distributions beyond that implied in 3 and may be of use for other types of input e.g. loading times may be normally distributed. The remaining items 6-13 relate mainly to analysis of output. It may be possible to analyse output without statistical analysis as indicated in Unit 6, in which case items 6-13 are not required. If some analysis is necessary, as detailed in Appendix C then a knowledge of 6, 7, 8 is required. Items 9-13 are not required anywhere in this package but would be useful for those who intend to pursue the subject further in circumstances where theoretical analysis is important.

All of the elementary textbooks referred to* cover 1, 2, 4, 5, 6, 8, 11 to some extent. All cover 12 to some extent except Cass and Snedechor

Unit 1 of this teaching package gave a brief exposition of 1 - 3 and Appendix A covers 4, also briefly.

* see overleaf

List of possible textbooks

Title	Authors(s)	Publisher	Number of Pages
A. Statistics in Theory and Practise	Conner and Morrell	Pitman	315
B. Statistical Methods in Management	T. Cass	Cassell	229
C. Statistics for Business	Taylor and Dunning	Polytech	547
D. Introduction to Business Statistics	R.C. Brite	Addison-Wesley	365
E. Statistics for Management and Economics	Mendenhall and Reinmuth	Wadsworth	902
F. Statistics for Management	R.I. Levin	Prentice-Hall	568
G. Modern Business Statistics	Freund and Williams	Prentice-Hall	541
H. Statistics	M.R. Spiegel	McGraw Hill	359
I. Statistical Methods	Snedecor and Cochran	Iowa State University	593

Notes on Textbook List

A and B are very elementary but A in particular covers basic material in a concise and useful way.

C, D, E, F, G, H are more ambitious and include a wide range of statistical techniques. E is particularly useful and includes Analysis of Variance and Design of Experiments.

I is a very well-known text and useful for those who expect a considerable involvement with statistical techniques, particularly design and analysis of experiments. Most of its examples are based on biology but have wide application.

CHAPTER 4

UNIT TWO OF TEACHING PACKAGE

4.1 INTRODUCTION TO UNIT 2 OF TEACHING PACKAGE

The purpose of this unit is to introduce the idea of and need for simulated samples. For this the student requires some knowledge of probability and in particular probability and cumulative probability distributions as covered in Unit 1. The text describes the use of random numbers to generate samples. As a preliminary one could use the tossing of a coin to generate samples from a distribution yielding only 2 values e.g. a service time distribution where service always takes either 5 or 10 minutes with equal probabilities. One makes the correspondances H $\rightarrow$ 5 minutes and T $\rightarrow$ 10 minutes.

Possible Sequence of Results from Tosses	H	H	T	H	T	T	H	H	H	T	H	T	T...
Corresponding Sample Values	5	5	10	5	10	10	5	5	5	10	5	10	10..

It is obvious that this works because (a) only two sample values were required viz. 5 and 10 and (b) they had equal probabilities. Suppose that the probability of a 5 should be three times that of a ten. In this case one could toss two coins simultaneously and make the correspondances HH $\rightarrow$ 10, HT $\rightarrow$ 5, TH $\rightarrow$ 5, TT $\rightarrow$ 5. It should be easy to go from tossing coins to selecting numbers from a hat - a far more flexible procedure.

In particular on completion of the unit students should understand the ideas of randomness and independence as applied to arrivals (this was also dealt with to some extent in Unit 1). They should understand the difference between random and pseudo-random numbers. They should be able to apply the Monte Carlo technique to the generation of sample values on a continuous scale.

Examples are used to illustrate arrival patterns which might not be random or independent. A service-time example is used to show the use of random numbers in obtaining samples. This is a simple extension of the coin-tossing experiment referred to above.

Some mathematics in this unit is unavoidable. However most has been left to Appendix A. What remains requires some facility in using tables and/or graphs. It is expected that all students especially management students should have this facility. All simulations require data and typically this data is in the form of frequency/probability distributions. It is the Monte Carlo technique applied to such distributions which "drives" the simulation, hence the necessity for understanding it. The concept of a mathematical function might cause some a little difficulty however it is so useful that it is worth explaining even in a simplistic way.

In the next unit (Unit 3) the ideas developed in this unit will be applied to a simple version of the godown model which will be further developed in subsequent units. However in order to test students' understanding of material in this unit some exercises such as the following could be set which partly anticipate material in Unit 3. Only nine vehicles are shown but the list should be extended to about 200 using suitable arrival and service times.

Test Exercise :

For the following observed data construct

(a) a cumulative probability distribution of inter-arrival times

(b) a cumulative probability distribution of service times

Then using the table of random numbers in the mini GPSS manual obtain a simulated sample of twenty arrival times and twenty service times.

Vehicle Number	Arrival Time of Vehicle	Check-in Commenced At	Check-in Finished At
1	9.00	9.00	9.01
2	9.00	9.01	9.03
3	9.05	9.05	9.08
4	9.07	9.08	9.09
5	9.12	9.12	9.14
6	9.13	9.14	9.16
7	9.15	9.16	9.18
8	9.15	9.18	9.21
9	9.22	9.22	9.23
.	.	.	.
.	.	.	.
.	.	.	.
200			

4.2 UNIT TWO - PROBABILITY AND RANDOM NUMBERS

The model to be developed is a traffic simulation. It deals with arrivals of vehicles and their progress through a system. It is in fact a combination of queueing systems. With any queueing system there are two basic sets of information required.

1. Arrival pattern
2. Service pattern

4.2.1 Arrival Pattern

While it is conceivable that arrivals could be timetabled and arrive at fixed pre-set times this is extremely rare in practice.

Even if an effort is made to timetable arrivals, because of many uncontrollable factors such as traffic conditions arrivals tend to be "random".

The best that one can hope for by timetabling is to control the actual numbers of arrivals. Thus a doctor can schedule patients to visit his surgery at fixed 15-minute intervals i.e. 2:00 p.m., 2:15 p.m. etc. In practice he knows that people will not arrive at these times but approximately four will arrive every hour. The arrivals will not be completely random but will tend that way. In the godown example while in theory vehicles could be given specific times at which to arrive, in fact customers would not find this acceptable. They must be able to come and go as they please. Since a simulation model seeks to imitate as far as possible the real system it must have some way of generating arriving customers in an apparently random manner, subject to an approximately fixed number per period.

4.2.1.1 Randomness

It is worth considering here exactly what is meant by "random". Arrivals are random if the probability of an arrival at one time is exactly the same as at any other time within the period in question i.e. there are no special times when arrivals are more likely. Strictly we must speak about the probability of an arrival in an interval and not at an exact point in time. Thus if we speak of the probability of an arrival at 2:00 p.m. we presumably mean in the interval 1:59:30 to 2:00:30 (hours:minutes:seconds) since the probability of an arrival at exactly 2:00 p.m. (i.e. 0.00000... seconds past 2 p.m.) is small. A more accurate definition of random arrivals would be "arrivals are random if the probability of an arrival in any interval depends only on the length of that interval and not on the time at the beginning of the interval". The complete opposite to random arrivals are timetabled arrivals. If arrivals are timetabled then the probability of an arrival at one of the scheduled times is 1 and is 0 at others. This assumes perfect adherence to the timetable. In fact many situations are in between the two extremes.

Example 4.1

Consider the situation where customers are scheduled to arrive, one at 2.00 p.m. and one at 2.07 p.m. If the timetable were strictly adhered to the probabilities at different times are estimated to be as follows:

(assume that 2.00 means 2.00 + 30 secs., 2.01 means 2.01 + 30 secs etc)

TABLE 4.1

Time	1.59	2.00	2.01	2.02	2.03	2.04	2.05	2.06	2.07
Probability of an arrival	0	1	0	0	0	0	0	0	1

Now suppose that a customer could arrive 1 minute early or up to 3 minutes late the probability of being one minute early being 0.1, one minute late 0.3, two minutes late 0.1, three minutes late 0.1 and of being on time 0.4. The probabilities of arrivals would then be as follows:

TABLE 4.2

	Probabilities for Customer 1						Probabilities for Customer 2					
Time	1.59	2.00	2.01	2.02	2.03	2.04	2.05	2.06	2.07	2.08	2.09	2.10
Prob. of an arrival	.1	.4	.3	.1	.1	0	0	.1	.4	.3	.1	.1

This is still not random as the probabilities at different times are not the same but it is tending that way. Suppose another arrival is scheduled at 2.04 so that the distributions overlap. Probabilities for any interval will then be even more similar.

4.2.1.2 Independence

Another concept applied to arrivals is that of independence. This is not the same as randomness. Independence means that the probability of an arrival at any time T is completely independent of all previous arrivals. Following are some examples of non-independence.

A. Births of babies in a family.

B. Arrival of customers at an airport check-in.

C. Arrival of trucks from customer X at a warehouse.

D. Arrival of trucks from customer X at a warehouse where the total number of arrivals per day is fixed.

A. The probability of the birth of a baby in June say depends on when the previous one (if any) was born. Thus if one was born in March, a birth in June is impossible!

B. Customers may arrive on buses. Thus they may tend to arrive in groups. Members of a group are not independent.

C. Since customer X probably loads trucks sequentially they will be spaced out by the loading time. Hence if a truck arrives at 1.00 p.m. an arrival at 1.05 p.m. is unlikely if loading takes say 1 hour.

D. Suppose 5 vehicles from customer X arrive each day at random times. Even though the times are random, arrivals are not independent since the probability of another arrival depends on how many have arrived already. For instance if 5 have arrived in the morning the probability of an arrival in the afternoon is zero.

The negative exponential distribution referred to in Unit 1 dealt with arrivals which are both random and independent.

4.2.2 Service Pattern

It is possible that a service provided to arriving customers could be of fixed duration e.g. always exactly 10 minutes. Again however this is rare except in automatic systems. For some types of service the time is almost constant e.g. the time to boil an egg. For others it may be very variable e.g. the time to repair a motor-car. Again the simulation model must have some way of generating service times which vary in a way similar to that observed in the real situation.

The tool used by a computer to provide this randomisation is called "a pseudo random-number generator". Basically this can be considered as a "black box" containing say 3-digit numbers from 000 to 999 on "pieces of paper". Each time a request is made of the "black-box" a number is drawn and made available to the computer program. It is then replaced. The actual procedure will be detailed later but for now the above picture will suffice. Of course all numbers in the box have exactly the same probability of being selected on each and every drawing.

It may be asked why 3-digit numbers, why not 2-digit or 4-digit or 5-digit? It will be seen later that there is a correspondance between random numbers and cumulative probability. The number of digits in the random number must at least equal the number of figures after the decimal point in the probability value. For most purposes three figures after the decimal point will be sufficiently accurate for the probabilities but if not then 4 or 5-digit random numbers should be used. If one only has 3-digit numbers then two can be combined i.e. if n_1 and n_2 are two 3-digit random numbers then $1000n_1 + n_2$ is a 6-digit random number.

4.2.3 Monte Carlo Technique

It will now be shown how the "black-box" can be used to generate a sample of service times.

Example 4.2

A repair-service takes approximately 10 minutes but sometimes a little more or a little less. The table below shows the results of two hundred observations made by a time-study analyst using a stop-watch. All service times have been rounded off to the nearest five minutes.

TABLE 4.3

Service Time	Number Observed	Relative Frequency
5 mins	20	10%
10 mins	120	60%
15 mins	60	30%
TOTAL	200	100%

In this example service times have been rounded off to the nearest 5 minutes for simplicity. Later this will be generalised, however it should be pointed out to students that this has been done.

Suppose that for the purpose of a simulation it is necessary to examine the progress of a large number of customers say one thousand. What service times should be used for these thousand customers?

The service time is always either 5 minutes, 10 minutes or 15 minutes and 10% of services take 5 minutes, 60% of services take 10 minutes and 30% of service take 15 minutes. It is necessary to generate service times for 1000 customers according to this pattern.

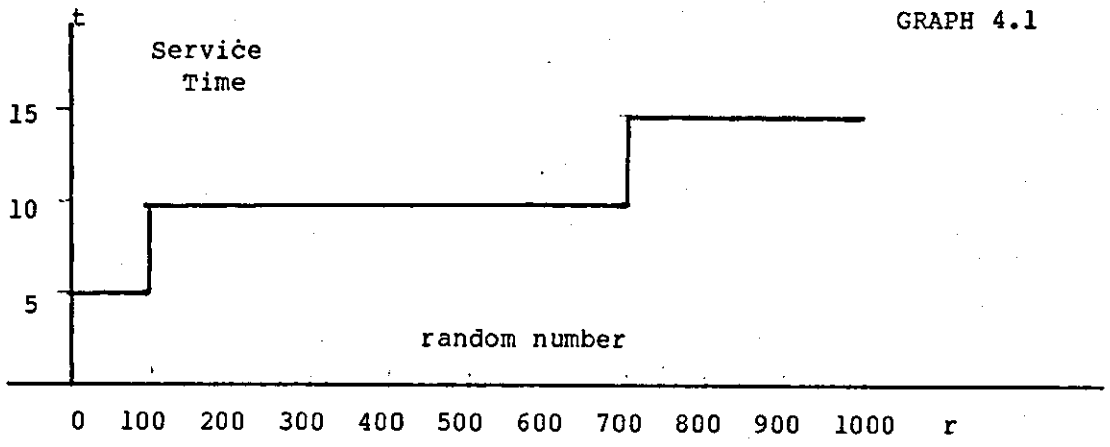
Students should be asked why one could not simply allocate service times of 5 minutes to the first 100 customers, 10 minutes each to the next 600 and 15 minutes each to the next 300. [This would produce a completely unrealistic pattern]. Of course one could go out and time 1000 customers and use the 1000 times so obtained, but that would be very time consuming and in fact is not necessary. It may also be impossible because frequently one is modelling a system which does not yet physically exist so sampling must be simulated. This is the case with the godown example to be discussed later. The sampling can be done synthetically. Consider the first customer. A time must be selected such that the probability of selecting 5 minutes is .1 (10%), the probability of selecting 10 minutes is .6 (60%) and the probability of selecting 15 minutes is .3 since those are the relative frequencies observed in practice. In fact not a time but a number from the black box will be selected. Since there are 1000 3-digit numbers between 000 and 999 but only 3 service times (5, 10, 15 minutes) then obviously more than one number must correspond to each service-time. Since the probability of selecting 5 minutes should be 0.1 then the probability of selecting from the corresponding "block" of numbers should be .1 but this is easy to arrange, it should simply consist of 10% of all possible numbers. The most obvious is the first 10% namely 000-099. Similarly 60% of random numbers (100-699) should give 10 minutes and 30% (700-999) should give 15 minutes.

TABLE 4.4

Service Time	Relative Frequency	Number of Random Numbers out of 1000	Block of Numbers
5 minutes	10%	100	000-099
10 minutes	60%	600	100-699
15 minutes	30%	300	700-999

Table 4.4 (columns 1 and 4) may be illustrated graphically as follows:

GRAPH 4.1



The procedure is to select a number between 000 and 999 from the black box. If the number is between 000 and 099 (inclusive) then take 5 minutes as the service time. If the number is between 100 and 699 take 10 minutes. If the number is between 700 and 999 take 15 minutes. Since all numbers have the same probability of being selected the probability of selection in any range equals the proportion of all possible numbers which fall in that range so the probability of selecting a number between 000 and 099 is 100/1000 = .1. Hence the probability of selecting 5 minutes is 0.1 and similarly for other values. One thousand such "random" numbers are selected from the black box and the 1000 corresponding service times are noted and used as the required sample values. Table 4.5 on the following page lists the first ten customers so obtained. This method is known as "Monte Carlo technique".

TABLE 4.5

Number Selected From Black Box (r)	This numbers falls in the range	Corresponding Service Time
484	100 - 699	10
878	700 - 999	15
082	000 - 099	5
032	000 - 099	5
653	100 - 699	10
881	700 - 999	15
659	100 - 699	10
933	700 - 999	15
657	100 - 699	10
788	700 - 999	15

The third column gives the required sample.

As an exercise students could be asked to each generate a sample of 50 values as described above either by drawing numbers from a hat or by using random number tables. Those students with a knowledge of statistics could be asked to use a chi-square test to determine whether the simulated frequencies agree (within reasonable limits) with the specified frequencies of 10%, 60%, 30%.

The instructor should emphasise what is involved in the procedure. On the one hand there is a random procedure under the control of the simulator which generates numbers 000, 001 . . . . 999 with equal probability. On the other hand there is a real life stochastic procedure which generates service times of 5, 10, 15 minutes with unequal probabilities. The important element of the simulation procedure is the rule which establishes a correspondance between the 2 sets of results, i.e. a rule which converts a random number to a service time.

General Procedure: Select a number r from the black box and then determine the corresponding service time t by means of some suitable rule or set of rules. In the example above the rule is incorporated in Table 4.4 which shows how to convert an r value to a t value. In words this rule is "find the block of random numbers which contains the selected random number r and use the corresponding service time as the sample value."

For the first customer in Table 4.5 the procedure just outlined proceeds as follows:

1. Select a random number. This is 484
2. Which block is this in? Answer, 100-699
3. Which value of t corresponds to the block 100-699? Answer, 10 minutes, using Table 4.6.

TABLE 4.6

Block of Numbers	Corresponding Service Times
000 - 099	5 minutes
100 - 699	10 minutes
700 - 999	15 minutes

This rule (or rules) must be carefully framed so that each t value has the correct probability of being chosen. This procedure is analogous to a board game where one rolls a die and then according to the result (1, 2, 3, 4, 5 or 6) there is a rule which tells one what to do. This may be for instance to advance a number of places on the board. The die is being used to simulate the operation of chance just as the black box. The rule stating what to do if one gets a six for instance is of course the essence of the game.

4.2.4 Functional Relationships

In mathematical terms such a rule for converting an r value to a t value is known as a functional relationship and is denoted t = f(r) or "t is a function of r". In general the statement "y is a function of x" simply means that the value of y depends in some way on the value of x. Suppose that y = time to unload a lorry. x = load on the lorry in kgs. It is clear that the "time to unload a lorry" depends on the "load on the lorry in kgs". Mathematically we say that y is a function of x. Such functional relationships can be expressed in several ways, in particular:

1. Table form
2. Graphical form
3. Explicit equation form
4. Implicit equation form

Only forms (1) and (2) are necessary for the models to be discussed, however, a brief description of all four is given in Appendix A.

4.2.5 Random Numbers

4.2.5.1 Random Number Generation

The model discussed so far has been over-simplified. Now it will be made a little more sophisticated, including some idea of how the black box works. This description of the generation of random numbers is mainly to satisfy ones intellectual curiosity. The generator can be treated as a black-box by students whose only concern is practical results. Others may refer to Appendix A for further details.

The numbers as provided by the imaginary black box are called uniform random numbers. They are random in the sense that each selection is completely independent of all previous selections and at each selection all numbers are equally likely to be selected. This is similar to writing numbers 000 to 999 on 1,000 pieces of paper, putting them in a box, mixing them and drawing one. After each selection the piece of paper is replaced in the box and the contents well mixed again. It is convenient to put a decimal point in front of the number so the range is .000 to .999 or in effect 0 to 1. The main reason for this will be explained later.

It can be said that the black box generates a uniform random number in the range (0, 1). While it is possible to have a black box which by some electronic procedure generates genuinely random numbers (such as for lotteries) in practice a series of pseudo-random numbers is generated by an arithmetic procedure which generates each new random number from the previous one. These numbers are not random since successive selections are not independent, indeed if one knows the arithmetic procedure one can always predict the next number from the present one.

4.2.5.2 Properties of Random Numbers

Random numbers should possess certain properties. Some examples of these properties are (a) each digit should occur with approximately the same frequency (b) the same digits should not always occur in sequence (c) there should be no unduly long runs upwards or downwards. Consider the following sequences A, B and C.

A. 853 564 452 590 405 555 759

The number 5 appears more frequently than one would expect from strictly random numbers.

B. 432 605 321 532 804 325 496

The number 3 is always followed by the number 2

C. 341 482 496 531 678 693 345

The first six numbers are in ascending order.

The above samples (7 in each case) are too small to draw any firm conclusions about their randomness or non-randomness and are given merely to illustrate some possible defects in pseudo-random number sequences. Using a suitable procedure it is possible to generate a series of pseudo-random numbers which appear to have all the properties of genuine random numbers and which will suffice for most purposes. The method has the advantage that one can repeat a sequence of random numbers knowing just the first one in the sequence. This is not possible with a genuinely random procedure unless each number in the sequence is recorded. The generation procedure is described in Appendix A.

4.2.5.3 Random Numbers in GPSS

When a request is made of GPSS for a random number one specifies which of the available generators one wishes to use. The number returned will be the first for that sequence of if that generator has been used previously in the program it will be the next in the sequence. The number can be requested in either of two formats (a) a fraction between 0 and 1, or (b) a three digit integer between 000 and 999.

In practice one can pretend that GPSS has a number of black boxes each producing a different sequence of random numbers and numbered R1, R2 ... . One may always use the same box but sometimes one wishes to use one box for one purpose and a second for another purpose or one might wish to run a program twice, once using R1 and next time using R2. Thus in our example we might use R1 to generate the service times and R2 to generate arrival times. It is easy to reproduce the numbers generated by R1 and R2 manually for checking results or tables of the numbers generated are available. One could use the same generator for both purposes but checking then becomes more complicated, it is easier to visualise one box as the "service time box" and the other as the "arrival time box". The reason one might want to use different random numbers on successive runs or replications is that one might want to know if a particular result is a consequence of the particular sample (set of random numbers) used or if the same result follows from other different samples. This will be discused in some detail later as it is an important point.

4.2.6 Generation of Sample Times on a Continuous Scale

Consider again the previous model

TABLE 4.6 (modified version of Table 4.4)

Service Time	Number Observed	Relative Frequency (as a proportion)	Corresponding Random Numbers
5 mins.	10	.10	.000 - .099
10 mins.	120	.60	.100 - .699
15 mins.	60	.30	.700 - .999
TOTAL	200	1.00	

Obviously the service times have been simplified by rounding off to the nearest 5 minutes. In practice some services must have been 6 or 7, or 12 minutes. Thus a service time recorded as 5 minutes could have been anything from say 2.5 (or whatever the minimum service time was) to 7.5 and that recorded as 10 minutes could have been anything from 7.5 to 12.5 minutes and that recorded as 15 minutes anything from 12.5 to 17.5 minutes (or whatever the maximum service time was).

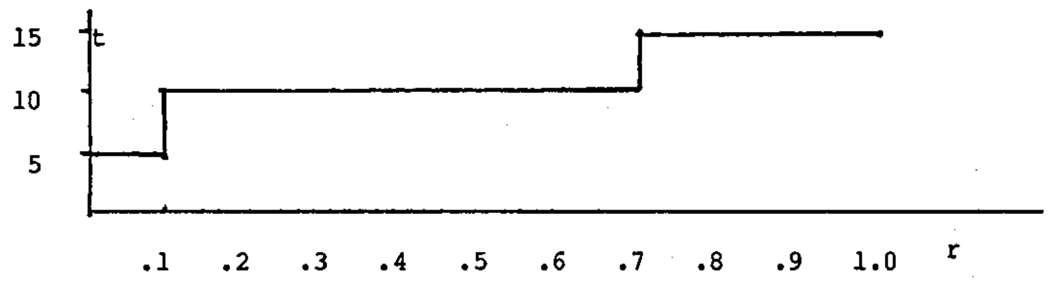
Table 4.7 below shows the range of actual service times corresponding to the rounded values used in Table 4.6 above

TABLE 4.7

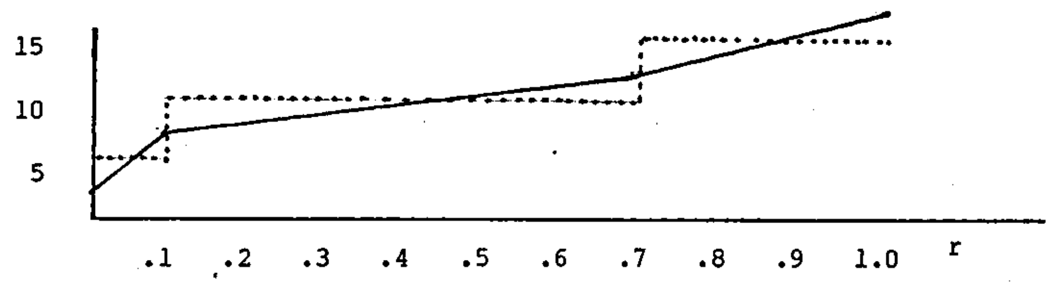
Random Number r	Approximate Service Time	Actual Service Time (t)
.000 - .099	5	2.5 - 7.5
.100 - .699	10	7.5 - 12.5
.700 - .999	15	12.5 - 17.5

Suppose that instead of plotting r against "approximate service time" as in Graph 4.1 (reproduced below) one plots r against actual service time as in Graph 4.2.

GRAPH 4.1 (using 0 to 1 scale for r instead of 0 to 1000)



GRAPH 4.2



To get this graph from Table 4.7 instead of having the block .000 to .099 for r all correspond to 5 as in Graph 4.1 .000 corresponds to 2.5 and .099 to 7.5. Similarly in the second block .100 corresponds to 7.5 and .699 to 12.5.

Graph 4.2 allows one to generate service times on a continuous scale (as really occurs) rather than to the nearest 5 minutes. For example if r = .583 then from Graph 4.2 t = 12 minutes

At this stage students could be asked to use the ten values of r in Table 4.5 to obtain ten values of t from Graph 4.2 (to the nearest minute).

In the previous illustration 3-digit random numbers have been used but in fact they can be considered as being on a continuous scale from 0 to 1. In this case they correspond to the cumulative probabilities discussed in Unit 1. This correspondance with cumulative probability is the reason we prefix a decimal point to our random numbers. The scale for cumulative probability is 0 to 1 so one also makes the random number scale 0 to 1 to make the correspondance exact, so that the horizontal scale in Graphs 4.1, 4.2 are also scales for F(t). Thus Table 4.7 would now become Table 4.8 below.

TABLE 4.8

Service Time	Probability	Random Number (r)
2.5 - 7.5	.1	.0000 - .0999...
7.5 - 12.5	.6	.1000 - .6999...
12.5 - 17.5	.3	.7000 - .9999...

The cumulative probability for any value of t simply means the probability of getting a service time less than or equal to that value of t. Thus in above the cumulative probability corresponding to t = 12.5 is 0.7 and this means that .7 or 70% of service times are less than or equal to 12.5 minutes. Table 4.8 rewritten in cumulative form is given on the next page as Table 4.9

TABLE 4.9

Service Time t	Cumulative Probability for t	Random Number corresponding to t
7.5	.1	.0999......
12.5	.7	.6999......
17.5	1.0	.9999......

4.2.7 General Monte Carlo Procedure

The procedure of selecting a random number and getting the corresponding service time can now be described as follows.

1. SELECT A RANDOM NUMBER r
2. FIND THE SERVICE TIME FOR WHICH THE CUMULATIVE PROBABILITY EQUALS r
3. THIS IS THE DESIRED SAMPLE SERVICE TIME

In common mathematical notation if t = service time then F(t) is the cumulative probability. Stage 2 of the above procedure amounts to finding the value of t which satisfies the equation F(t) = r for the selected value of r. Occasionally one may have an expression for F(t) in terms of t so that the equation F(t) = r can be solved once r is known (i.e. having been selected).

For example in Unit 1 it was shown that for the negative exponential distribution $F(t) = 1 - e^{-\lambda t}$.
Hence for this distribution $F(t) = r$
becomes $1 - e^{-\lambda t} = r$.
Once r has been selected stage 2 amounts to solving $1 - e^{-\lambda t} = r$ for t (see Appendix A). However it is more usual that the relationship between F(t) and r be expressed in tabular (as in our example) or graphical form as in Graphs 4.1 and 4.2 so solving F(t) = r is a matter of interpolating in the table or graph, as in the example, on the following page.

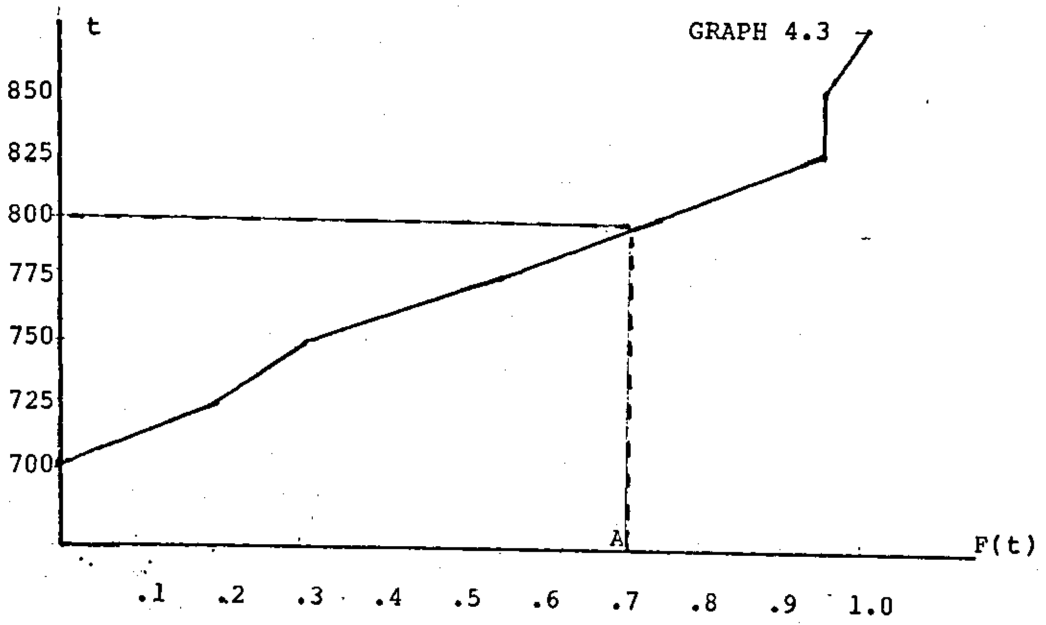
"Service Time" has been used here as a convenient example but the same procedure applies to "Inter-arrival Times" or indeed to any distribution one wishes to simulate

Example 4.3 Given the following cumulative probability distribution of service times obtain a simulated sample of three service times using the random numbers .700, .287, .906.

TABLE 4.10

Service Time t	Cumulative Probability F(t)
700	0.00
725	0.20
750	0.30
775	0.55
800	0.75
825	0.95
850	0.95
875	1.00

First draw the graph below



The first random number selected is 0.700

Select the point on the F(t) scale where F(t) = .7

This is A in above graph.

Then get the corresponding point on the vertical (t) scale.

This is 795 approximately - the first sample value.

Repeat for successive r values.

CHAPTER 5

UNIT THREE OF TEACHING PACKAGE

5.1 INTRODUCTION TO UNIT 3 OF TEACHING PACKAGE

To introduce the idea of simulation a real situation is first described. This is a highly simplified version of the case-study. First there is a description of how the system would be analysed without using simulation, using conventional observation methods. Then the simulation approach is described by showing how exactly the same steps can be followed by simulating observations.

Students with limited mathematical ability usually find difficulty in using probability distributions. Examples are related as closely as possible to the specific case being studied. Care should be taken that students understand clearly how one goes from Table 5.6 to Table 5.7 to Graph 5.1 (This refers to the inter-arrival time distribution, the others are similar) and how random numbers are used to obtain sample times from the graph as this is the essence of the Monte Carlo technique.

By the end of this unit the students should have some idea of how one might analyse a real system viz. how to obtain required information such as mean queueing time etc. They should appreciate what data would be required to enable the simulator to perform the same analysis and know how to obtain this data either by observation or synthesis.

A practical example could be set to test this knowledge. For instance a familiar canteen could be selected for analysis. Students could be asked to prepare suitable forms for collection of data and presentation of results on the operation of the system. Other examples should be readily available such as college car-parks etc. They could also be asked to show what data would need to be collected for a simulation of the system. Alternatively they could do the following test exercise.

Test Exercise on Unit 3

Customers arrive at a hairdressing salon at random times i.e. you may assume that the inter-arrival time is a negative exponential distribution. The average arrival rate is 5 per hour. The service time is uniformly distributed between 15 and 30 minutes, you may assume it always takes a whole number of minutes (i.e. service time is one of the numbers 15, 16, 17 ... 30 and all are equally likely). There are two hairdressers.

Simulate manually the arrival and processing of 25 customers, assuming the first arrives at 08.30. In order to obtain the cumulative probability distribution for arrival times see the construction of Table 3.7 in Unit 1.

(Assume customers form a single queue and that when a hairdresser becomes free she will next take the person at the head of the queue. If both hairdressers are free when a customer arrives choice must be random).

Fill in the data in tables 1 and 2, customer by customer.

Students with previous computer experience should do the following : Write a FORTRAN or BASIC program to do the same thing for 1000 customers. Assume the first customer arrives at time t = 1. Use integral times i.e. 1, 2 ... instead of clock times.

TABLE 1

CUSTOMER STATISTICS

1 Customer No.	2 Random No.	3 Inter-Arrival	4 Random No.	5 Service Time	6 Arrival Time	7 No. in System on Arrival	8 No. Waiting	9 Which Server A or B	10 Starts Service At	11 Finishes Service At	12 Waiting Time	13 Total Time
1	-	-			08.30	0	0	A	08.30		0	
2												
3												
4												
5												
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												
21												
22												
23												
24												
25												
TOTALS			▨		▨			▨	▨	▨		
MEANS												

TABLE 2

SERVER STATISTICS

SERVER A

Customer Number	Starts Service	Finishes Service	Service Time
		TOTAL	
		MEAN	

SERVER B

Customer Number	Starts Service	Finishes Service	Service Time
		TOTAL	
		MEAN	

TOTAL TIME FROM ARRIVAL OF CUSTOMER 1 TO DEPARTURE OF CUSTOMER 25 = $\boxed{\quad}$

TOTAL TIME A WAS BUSY =

OCCUPANCY RATE FOR A =

TOTAL TIME B WAS BUSY =

OCCUPANCY RATE FOR B =

5.2 UNIT THREE - SIMULATION OF A SIMPLE SYSTEM

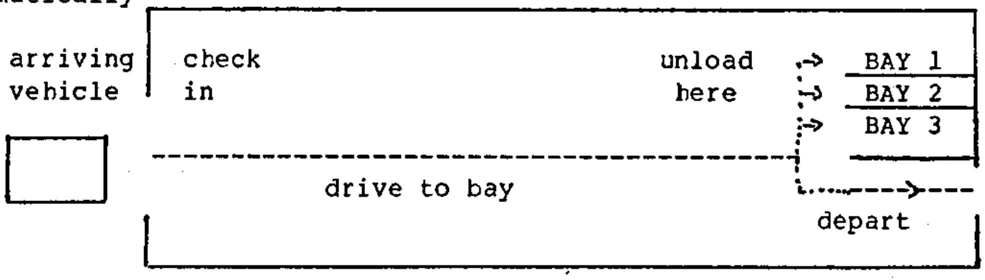
A number of simulation languages exist of which GPSS is probably the most widely used. The purpose of a simulation language is to simplify the programming which needs to be done by a person who wishes to construct a computer simulation model. Many functions which would be difficult to program are built in. For instance there is a built-in clock which can keep track of the time and arranges for things to happen at specified times. There are many built-in facilities for data collection. For example if the model is simulating a queueing system one can request GPSS to keep records of queue sizes and waiting times. There are built-in, not only random number generators, but facilities for generating random events such as arrivals.

To understand how a simulation model works consider a real situation where instead of simulation one uses actual observation and data collection.

5.2.1 A Simple System

The system to be considered first is a very simple one. Lorries arrive at a gate where there is some check-in procedure e.g. a clerk records vehicle number and/or checks the driver's identification. Once admitted the lorry drives to the unloading bay where it is unloaded. It then leaves. There are three unloading bays.

Diagrammatically



A queue may form at either the check-in or at the bays or at both.

5.2.2 Analysis of a Real System

If a time-study of the real situation, as distinct from a simulation, were to be carried out the procedure would be as follows.

One person is positioned at the gate with a stop-watch. He notes arrival times, how long a vehicle must wait before being checked in, the time required to check it in and the departure time from the check-in. He might complete a table such as Table 5.1 (which shows only the first six vehicles). The "Number in the System" is the total number in the system including any being serviced when a vehicle arrives but excluding the arriving vehicle.

TABLE 5.1

Vehicle Registration Number	Arrival Time (hr:min:sec)	Number in the System	Time at which check-in starts	Time at which check-in finishes
AG 4316	10.30.00	0	10.30.00	10.31.40
AD 3621	10.30.00	1	10.31.40	10.33.05
BF 4688	10.30.00	2	10.33.05	10.36.00
BG 1920	10.30.50	3	10.36.00	10.40.30
CB 2052	10.31.00	3	10.40.30	10.41.35
AD 4396	10.35.00	3	10.41.35	10.42.50
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Another person would be positioned at the bays also with a stop-watch and he would record the same information for the unloading function. His table would be as follows (for the first six vehicles only).

TABLE 5.2

Vehicle Registration Number	Arrival Time at Bays	Bay Number	Number in the System	Time at which unloading starts	Time at which unloading finishes
AG 4316	10.32.00	1	0	10.32.00	11.00.00
AD 3621	10.33.35	2	0	10.33.35	11.05.00
BF 4688	10.36.30	3	0	10.36.30	11.04.30
BG 1920	10.40.55	1	1	11.00.00	11.27.40
CB 2052	10.42.00	2	1	11.05.00	11.31.30
AD 4396	10.43.30	3	1	11.04.30	11.30.20
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

After 5 days of observations data would have been collected on say 200 vehicles. The results would be analysed and another set of tables compiled such as Table 5.3 (which shows only the first six vehicles).

TABLE 5.3
TIME ANALYSIS FOR EACH VEHICLE
(all times in seconds)

A Reg. No.	B Waiting Time	C Check in Time	D Drive Time	E Waiting Time	F Unloading Time	G (B+C+D+E+F) Total Time	H (B+E) Total waiting Time
AG 4316	0	100	20	0	1683	1800	0
AD 3621	100	85	30	0	1885	2100	100
BF 4688	185	175	30	0	1680	2070	185
BG 1920	310	270	25	1155	1660	3420	1465
CB 2052	570	65	25	1380	1590	3630	1950
AD 4396	395	75	40	1260	1550	3320	1655
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
TOTALS	1560	770	170	3795	10,045	16,340	5355
AVERAGE	260	128	28	632	1674	2723	892

From the above vehicle by vehicle analysis a table of average results could be compiled as below. The figures are based on the six vehicles for which figures were given. Of course in practice the number observed would need to be much bigger in order to get reasonably accurate averages and percentages.

TABLE 5.4

ANALYSIS OF RESULTS

Average time for check-in	128
Average time to drive from check-in to bays	28
Average unloading time	1674
Average waiting time at check-in	260
Average waiting time at bays	632
Average total waiting time	892
Average total time in the system	2723
Percentage which spent more than 45 minutes in the system	50%
Percentage which spent more than one hour in the system	17%

5.2.3 Analysis by Simulation

The simulation model would do the same thing as the analysis just described i.e. individual vehicles such as CB 2052, AD 4396 would be created. They would arrive at certain times, be traced through the system and records kept and subsequently analysed just as in the real study.

5.2.3.1 Variability

In the system being discussed variability is associated with four factors.

(a) the arrival time

(b) the time required to check-in

(c) the time required to drive from check-in to bays

(d) the time required for unloading

These four times will be selected for each vehicle by a procedure similar to that outlined in Unit 2. It will be recalled that this procedure requires a knowledge of F(t) the cumulative probability distribution for each factor. It will be necessary to obtain these distributions. This is usually done by observation where practical. This is of course impossible if the system is not yet in existence. What one might do in such a case is described on page 92.

5.2.3.2 Obtaining the Distributions from Observation

In the sample which follows the data are for 20 vehicles. It should be emphasised that 20 is too few. In practise about 200 would be desirable. The 20 is only for illustrating the procedure which is the same for 20 or 200. The basic data used for compiling the table below are collected as described for Tables 5.1 and 5.2

TABLE 5.5

(all durations in seconds)

VEHICLE NUMBER	ARRIVAL TIME	TIME SINCE PREVIOUS ARRIVAL	TIME TAKEN TO CHECK IN	TIME TO DRIVE TO BAY	TIME TO UNLOAD
1	8.31	60	55	50	800
2	8.32	60	50	53	750
3	8.55	1380	80	49	822
4	9.10	900	65	55	723
5	9.23	780	60	60	790
6	9.40	1020	100	52	811
7	10.03	1380	58	47	760
8	10.06	180	45	53	720
9	10.30	1440	58	68	785
10	10.45	900	55	60	763
11	11.17	1920	65	55	810
12	11.30	780	60	50	770
13	11.50	1200	63	56	743
14	12.01	660	54	49	852
15	12.10	540	83	50	775
16	12.12	120	52	53	720
17	12.15	180	59	62	803
18	12.42	1620	66	48	708
19	13.05	1380	75	50	752
20	13.31	1560	50	51	789

5.2.3.3 Inter-arrival Times

The column headed "TIME SINCE PREVIOUS ARRIVAL" gives the times between successive arrivals in seconds. In the case of the first vehicle it is the time since the opening of the system which it will be supposed was 8.30. The inter-arrival times will be analysed first.

The smallest inter-arrival time is 180 and the largest is 1920 i.e. a range of approximately 2000. This can be divided into a number of classes. Usually 5-7 classes is about right. In this case of course with only 20 vehicles there will be too few to adequately estimate the proportions in each class. In practice there should be at least 200 altogether.

TABLE 5.6

Inter-arrival Time Analysis

Inter-arrival time range	Number in this class	Proportion in this class	Cumulative Proportion
0 - 300	5	.25	.25
301 - 600	1	.05	.30
601 - 900	5	.25	.55
901 - 1200	2	.10	.65
1201 - 1500	4	.20	.85
1501 - 1800	2	.10	.95
1801 - 2100	1	.05	1.00

The cumulative proportions can be used to estimate the cumulative probability. The following table (5.7) is in the form required for GPSS.

TABLE 5.7

Inter-arrival time t	Cumulative Probability F(t)
0	0
300	.25
600	.30
900	.55
1200	.65
1500	.85
1800	.95
2100	1.00

Note that the cumulative proportion or probability refers to the top point of the range in each class e.g. 0.85 of inter-arrival times are equal to or less than 1500.

5.2.3.4 Check-in Times

The range is from 45 to 100 so intervals of 10 would give about six classes.

TABLE 5.8

Check-in time range	Number in this class	Proportion in this class	Cumulative Proportion
41 - 50	3	.15	0.15
51 - 60	9	.45	0.60
61 - 70	4	.20	0.80
71 - 80	2	.10	0.90
81 - 90	1	.05	0.95
91 - 100	1	.05	1.00

TABLE 5.9

Check-in time t	Cumulative Probability F(t)
40	0.00
50	0.15
60	0.60
70	0.80
80	0.90
90	0.95
100	1.00

5.2.3.5 Drive Times to Bays

The range is from 47 to 68 so intervals of 4 would give about six classes.

TABLE 5.10

Drive time range	Number in this class	Proportion in this class	Cumulative Proportion
45 - 48	2	.10	.10
49 - 52	8	.40	.50
53 - 56	6	.30	.80
57 - 60	2	.10	.90
61 - 64	1	.05	.95
65 - 68	1	.05	1.00

TABLE 5.11

Drive-time t	Cumulative Probability F(t)
44	0.00
48	0.10
52	0.50
56	0.80
60	0.90
64	0.95
68	1.00

5.2.3.6. Time to Unload

The range is from 708 to 852 so intervals of 25 would give seven classes. Alternatively one could use intervals of 30.

TABLE 5.12

Unload time range	Number in this class	Proportion in this class	Cumulative Proportion
701 - 725	4	.20	.20
726 - 750	2	.10	.30
751 - 775	5	.25	.55
776 - 800	4	.20	.75
801 - 825	4	.20	.95
826 - 850	0	.00	.95
851 - 875	1	.05	1.00

TABLE 5.13

Unload-time t	Cumulative Probability F(t)
700	0.00
725	0.20
750	0.30
775	0.55
800	0.75
825	0.95
850	0.95
875	1.00

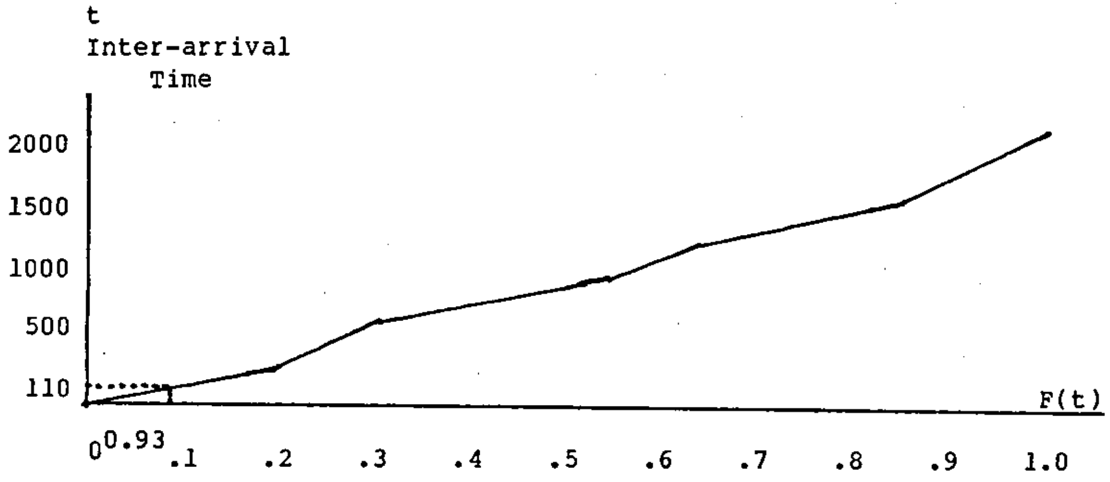
5.2.3.7 Generation of Sample Values by GPSS

Once GPSS has been supplied with the information in the four tables 5.7, 5.9, 5.11 and 5.13 it can supply for any vehicle an arrival time, check-in time, drive time and unload time.

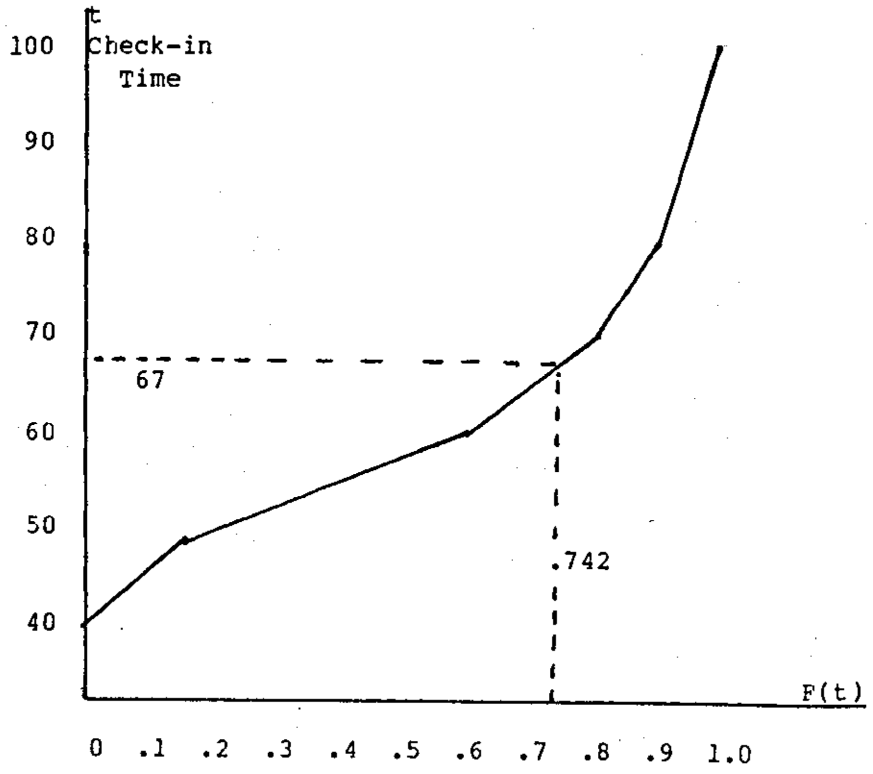
How this is done is shown below for the first two vehicles.

It can be considered that the computer holds the four tables in the form of graphs as below (Graphs 5.1 - 5.4)

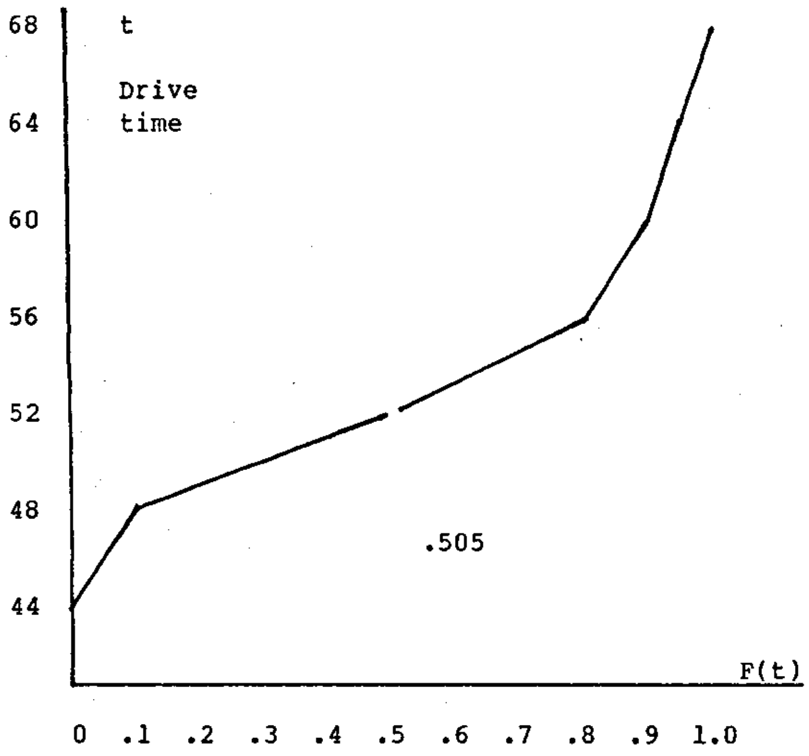
GRAPH 5.1



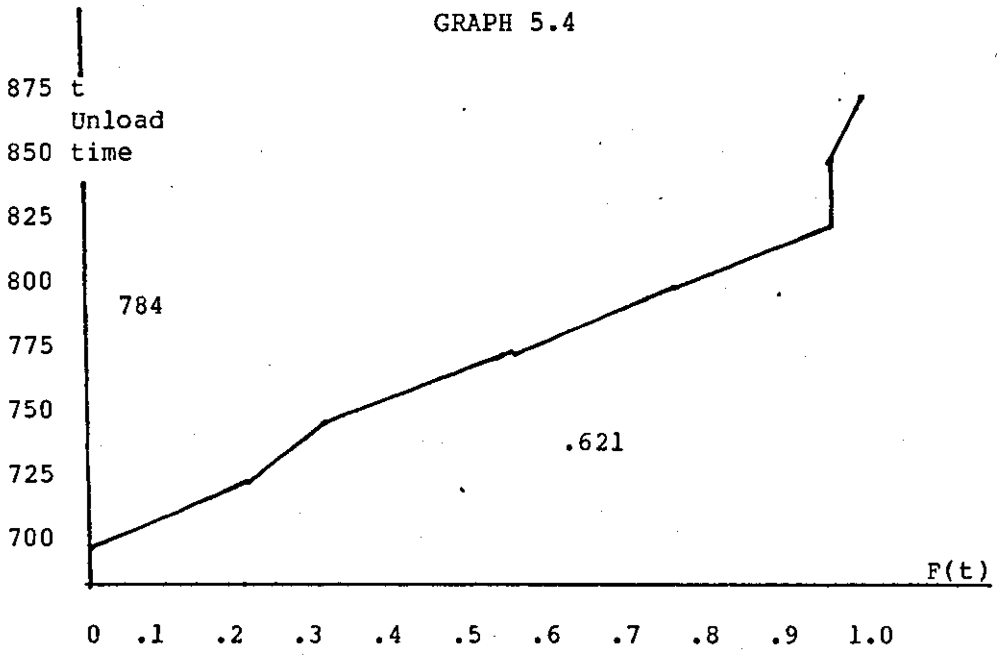
GRAPH 5.2



GRAPH 5.3



GRAPH 5.4



For each vehicle four random numbers will be selected, one for each factor, then from the graphs corresponding times will be obtained. These random numbers are generated by the pseudo-random number generators referred to in Unit 2. The procedure is as outlined on page 71.

Suppose that for vehicle 1 the four random numbers generated are .093, .742, .505, .0621. Put F(t) = the random number in each case.

From Graph 5.1 F(t) = .093 corresponds to an inter arrival time of 110.

From Graph 5.2 F(t) = .742 corresponds to a check-in time of 67.

From Graph 5.3 F(t) = .505 corresponds to a drive time of 52.

From Graph 5.4 F(t) = .621 corresponds to an unloading time of 784 (all time durations are in seconds).

Suppose that for vehicle 2 the random numbers are .700, .342, .078, .569.

The results for both vehicles are tabulated below.

TABLE 5.14

Vehicle	Inter-arrival Time		Check-in Time		Drive Time		Unload Time	
Number	F(t)	t	F(t)	t	F(t)	t	F(t)	t
1	.093	110	.742	67	.505	52	.621	784
2	.700	1275	.342	54	.078	47	.569	777

The arrival times are with respect to a starting time of 8.30. Thus vehicle 1 arrives 110 seconds after opening i.e. 8.31.50 Vehicle 2 arrives 1275 secs. (or 21 mins. and 15 seconds) after vehicle 1 i.e. at 8.53.05.

The above information can be generated for any number of vehicles. Of course the other data, waiting times etc. are not generated they are calculatable by the computer once the four times discussed above are available for each vehicle. This is done as indicated below for 10 vehicles in Table 5.15. Note that columns headed, B, C, D, E are generated as described above but columns F - L are calculated as indicated at the head of the column. Once Table 5.15 has been compiled a table of results such as Table 5.4 can be obtained.

For convenience in Table 5.15 arrival times are in seconds from the opening time. This is more convenient than using clock times. Thus vehicle 1 arrives at t = 110 (8.31.50 in clock time) and vehicle 2 arrives at t = 1385 (8.53.05 in clock time).

Also for simplicity it is assumed that there is only one bay.

TABLE 5.15

A Vehicle Number	B Arrival Time	C Check- in Time	D Drive Time	E Unload Time	F Waiting at Check-in	G Finishes Check-in	H Arrives at Bay	I Waiting Time for Bay	J Finishes Unloading	K Total Time in System	L Total Waiting Time
	These are generated as described on page 89										
1	110	67	52	784	0	177	229	0	1013	903	0
2	1385	54	47	777	0	1439	1486	0	2263	878	0
3	1401	90	54	805	38	1529	1583	680	3068	1667	718
4	2431	50	52	750	0	2481	2533	535	3818	1387	535
5	2960	82	49	820	0	3402	3091	727	4638	1678	727
6	3412	62	55	765	0	3474	3529	1109	5403	1991	1109
7	4035	57	63	728	0	4092	4155	1248	6131	2096	1248
8	5840	85	65	810	0	5990	5990	141	6941	1101	141
9	5900	63	51	785	25	6039	6093	902	7726	1826	927
10	7513	59	61	862	0	7633	7633	93	8588	1075	93

Columns F-L are obtained as in the following examples.
Vehicle 1 arrives at time 110 (B). There is no other vehicle at the check-in so it does not wait hence F is zero. The Check-in time is 67 (C) so G = 110 + 67 = 177, the time at which it finishes checking in. It requires 52 (D) minutes to drive to the bay so it arrives at H = 177 + 52 = 229. As there is no other vehicle at the bay I = 0. Its unloading time is 784 (E) so it finishes unloading at J = 229 + 784 = 1013. Since it arrived at 110 (B) its total time in the system was 1013 - 110 = 903 = K. It did not have to wait at either the check-in or bay so L = 0.

Note that when Vehicle 3 arrived at 1401, Vehicle 2 was still at the check-in since its check-in finishes at 1439 so Vehicle 3 must wait for time 1439 - 1401 = 38 = F.

The students should be asked to generate B, C, D, E for several more vehicles as described on page 89 and hence calculate F - L for each vehicle.

5.2.4 Obtaining the Data Required for the Simulation

To obtain the cumulative probability distributions one would generally need to carry out a time study. The student might well ask at this stage why one should do the simulation at all if one still has to carry out the survey of the real system. The survey outlined on pages 77 - 78 was of 200 vehicles only. This might suffice to establish the probability distributions referred to, but in most cases would not be sufficient to accurately assess the quantities in Table 5.4. One would prefer say one thousand vehicles. More will be said later about sample size. The point here is that one is substituting a small survey plus simulation for a large survey. Another reason for prefering simulation is that if there are several check-ins and/or several unloading bays then in order to follow particular vehicles' progress it would be necessary to survey all simultaneously with stop-watches but this would not be necessary to establish the probability distributions for (a,), (b), (c), (d) above (page 80), a sample of vehicles would suffice.

Still another reason is that the whole survey may be theoretical, the system may be planned but not yet exist. In this case the distributions would have to be arrived at synthetically from previous experience. This indeed was the situation in which these models were originally designed. Architectural/engineering plans had been drawn up for a building of a certain size with defined routes for vehicles. The engineers wanted to know what traffic throughput could be handled given certain constraints on traffic and to investigate different possible traffic rules.

Obviously they could not wait until the building was actually in use before doing this as it was conceivable that the results of the investigation might make design changes necessary. The distributions required for input (arrival rates, unloading rates, progression times) had to be obtained by considering similar systems elsewhere and estimating utilisation.

5.2.5 Describing Distributions

5.2.5.1 Methods of Description

There are two ways of describing any distribution.

First way - empirically i.e. a table of observed data such as Tables 5.7, 5.9, 5.11, 5.13. These tables were compiled from observed data. There are no theoretical assumptions. It might be necessary to compile different versions of Table 5.7 for different periods of the day if peaks and troughs occur.

Second way - theoretically, here one states the theoretical form or shape of the distribution and gives the values of any parameters required to describe it.

For both methods observations are desirable. For a theoretical distribution one needs observations to justify the assumed theoretical shape and to establish the values of the parameters. If observations are not possible e.g. if the system does not yet exist then the second way may be easier since from observing similar systems one might know what shape to expect and data might be available to estimate the parameters e.g. the average time to unload a lorry.

GPSS can always handle empirical distributions. It can also handle a few specific theoretical distributions.

Now consider the four distributions in turn i.e. the inter-arrival times, check-in times, drive times and unload times.

5.2.5.2 Inter-arrival Times

If for some reason arriving vehicles cannot be observed one could proceed as follows: First estimate the total number of vehicles expected to arrive per day. It might be possible to do this by comparison with similar buildings or from consultation with users (existing or proposed) of the buildings. Then estimate the pattern of arrivals. The simplest assumption is that arrivals will be random and independent at a constant rate over the day. This will not be realistic if one expects peaks and troughs in traffic flow throughout the day. If this were the case one could divide the day into periods with a different arrival rate for each period but arrivals assumed to be random and independent within each period.

Using then the second method for describing the distribution for any period:

Shape - arrivals are random and independent
(in statistics this is known as a Poisson distribution)

Parameter - average arrival rate is n per hour (where n is given a specific value for each hour).

The above information will suffice for GPSS in place of Table 5.7.

5.2.5.3 Check-in Times

If this is relatively short e.g. a matter of one or two minutes with little variability compared to other factors it might be felt reasonable to use a constant check-in time. This could be estimated by a work-study analyst from a knowledge of the work involved at check-in.

5.2.5.4 Drive Time

Knowing the exact distance involved and possible driving speeds this can be estimated. As for check-in times it might be possible to use a constant time. If however times are likely to be variable because of obstacles or varying drive speeds one might be able to argue as follows (from previous knowledge).

It should take 50 to 60 seconds to drive from gate to bay but there is a small chance that it could be 70 seconds. This could be quantified as in Table 5.16 below.

TABLE 5.16

Drive time	Probability (estimated)
50	.4
60	.4
70	.2

This would then be converted to a cumulative probability distribution as below in Table 5.17

TABLE 5.17

Drive Time t	Cumulative Probability F(t)
45	0.00
55	0.40
65	0.80
75	1.00

5.2.5.5 Unload Time

Suppose that all vehicles are of the same type (this restriction will be relaxed later) so that unloading time only depends on the load involved. Of course it will also depend on the speed of the handlers but the contribution of this to the variability may be much less than the variation in load size so it might be possible to ignore it. It should be possible by consulting customers to establish what proportion of lorries will be fully loaded, half-loaded etc. so that one could arrive at Table 5.18

TABLE 5.18

Percentage Load	Proportion of Vehicles	Unloading Time
100%	.6	20 - 24 minutes
75%	.2	18 - 20 minutes
50%	.1	14 - 16 minutes
25%	.1	7 - 12 minutes

From Table 5.18 one could derive the cumulative form as required for simulation.

TABLE 5.19

Unloading Time (minutes)	Cumulative Probability
7	0.00
12	0.10
16	0.20
20	0.40
24	1.00

5.7.5.6 Accuracy

The methods just described are a poor substitute for observation but frequently one must make decisions based on less than perfect information.

CHAPTER 6

UNIT FOUR OF TEACHING PACKAGE

6.1 INTRODUCTION TO UNIT 4 OF TEACHING PACKAGE

This unit introduces the GPSS language concepts. The version used by the author was GPSS 1100 for a UNIVAC 1100 and that is the version described here and in the GPSS manual provided. For actual use on a different system one should consult a manual for the version available on the computer one is using.

The approach suggested is to introduce the minimum number of instruction types (blocks) required to write a simple programme. In fact five suffice to write a programme for the model of Unit 3. These five blocks are described in detail.

In order to avoid introducing too many new concepts at one time a general discussion of time distributions has been deferred to Unit 5. Thus in this unit all processing times are regarded as constant or uniformly distributed over a finite range. This is of course unrealistic and these processing times will be considered again more realistically in Unit 5.

The GPSS concepts introduced in this unit are

TRANSACTIONS

FACILITIES

Data definition statements

Model statements or blocks

Model control statements

(Only Model statements are considered in detail in this unit).

The specific blocks introduced are

ADVANCE, GENERATE, HOLD, QUEUE, (RELEASE), (SEIZE), TERMINATE. The SEIZE and RELEASE blocks are not necessary for the model in this unit but it is convenient to introduce them.

By the end of the unit a student should be able to specify the blocks required to simulate a simple queueing system although as yet without detailed time specification. A simple exercise such as the following could be used to test this ability. In practice students did not appear to have any difficulty with the material in this unit which is quite straightforward.

Test Exercise :

Two different vehicle types (A and B) arrive at a dock to be unloaded. There are two bays, bay 1 for type A vehicles and bay 2 for type B vehicles. The inter-arrival time for type A is T1 (regard all times as constant, of course in practise they will be variables) and for type B is T2. The unloading time can be divided into three stages. Stage 1 takes T3 minutes and requires no special equipment, stage 2 takes T4 minutes and requires a fork-lift, stage 3 takes T5 minutes and requires no special equipment. These unloading times are the same for both vehicle types and for both bays. There is only one fork-lift. On completion of loading the vehicle leaves. Separate queues form in front of the two bays. Write a GPSS program to model this system (hint: write two separate segments for the two vehicles types each segment commencing with a GENERATE and ending with a TERMINATE statement). Generate 100 vehicles of each type.

Suggested solution for Test Exercise

```
*   Segment for Type A vehicles

     GENERATE 1,100                           TIME(T1)
     QUEUE              BAY1.QUEUE
     SEIZE              BAY1
     ADVANCE                                  TIME(T3)
     HOLD               FORK.LIFT             TIME(T4)
     ADVANCE                                  TIME(T5)
     RELEASE            BAY1
     TERMINATE


*   Segment for Type B vehicles

     GENERATE 1,100                           TIME(T2)
     QUEUE              BAY2.QUEUE
     SEIZE              BAY2
     ADVANCE                                  TIME(T3)
     HOLD               FORK.LIFT             TIME(T4)
     ADVANCE                                  TIME(T5)
     RELEASE            BAY2
     TERMINATE
```

6.2 UNIT FOUR - INTRODUCTION TO GPSS

6.2.1 Statements

The previous unit described in general terms how a simulation might be carried out but nothing about how the program actually carries out the instructions, or how one instructs GPSS was mentioned.

A GPSS program consists of three main parts:

1 Data definition statements

2 Model statements or blocks

3 Model control statements

A fourth part "Report Editing" is optional and will be discussed in later units.

The data definition statements are to provide numeric data such as probability distributions to be used, parameter values etc.

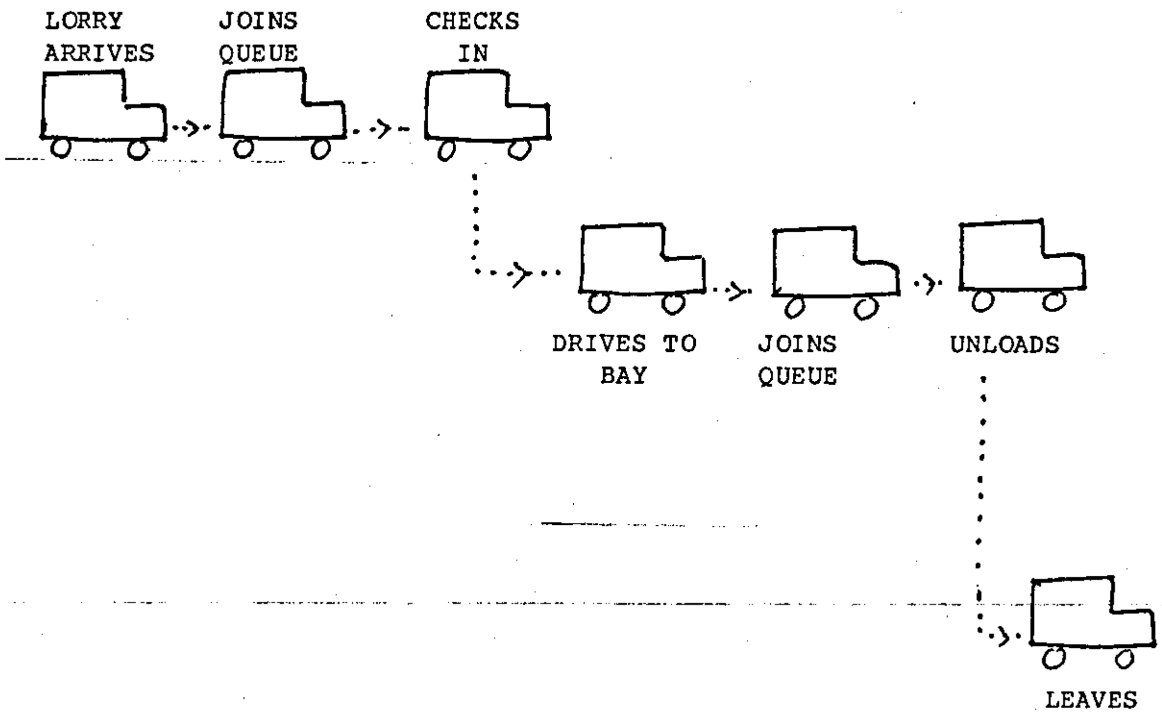
The blocks constitute the actual simulation model.

The model control statements give certain instructions on procedures to be followed by the simulator.

First the model blocks or statements will be introduced (data definition statements will be considered in Unit 5 and control statements in Unit 6). In a simple model these correspond very closely to actual physical activities and this should be stressed. This is illustrated in the table which follows (Table 6.1). In practice a block, as well as its name, contains other information which is not included in the blocks in Table 6.1 which is intended merely to introduce the idea of a block and its relationship to an actual activity. The completed blocks are described later.

TABLE 6.1

ACTIVITY	PROGRAMME BLOCK TYPE
1. Lorry arrives at time t and joins the system	GENERATE
2. Lorry joins queue for check-in	QUEUE
3. Lorry checks in	HOLD
4. Lorry drives to bay	ADVANCE
5. Lorry joins queue for bay	QUEUE
6. Lorry is unloaded	HOLD
7. Lorry leaves system	TERMINATE



6.2.2 TRANSACTIONS

The program block corresponding to the arrival of a lorry is a "GENERATE" block. As its name implies this block generates items. An item so generated is referred to in GPSS as a TRANSACTION. In the above example TRANSACTION = LORRY but in general a TRANSACTION can be any item requiring processing through the system. The opposite to a GENERATE block is a TERMINATE block. The latter terminates (i.e. removes from, the system) a TRANSACTION. There are several examples in GPSS of such pairs of complementary blocks. TRANSACTIONS can be considered as progressing from block to block. This progress may be held up temporarily or even permanently. Consider the sequence of blocks.

QUEUE

HOLD CHECKIN TIME (X)

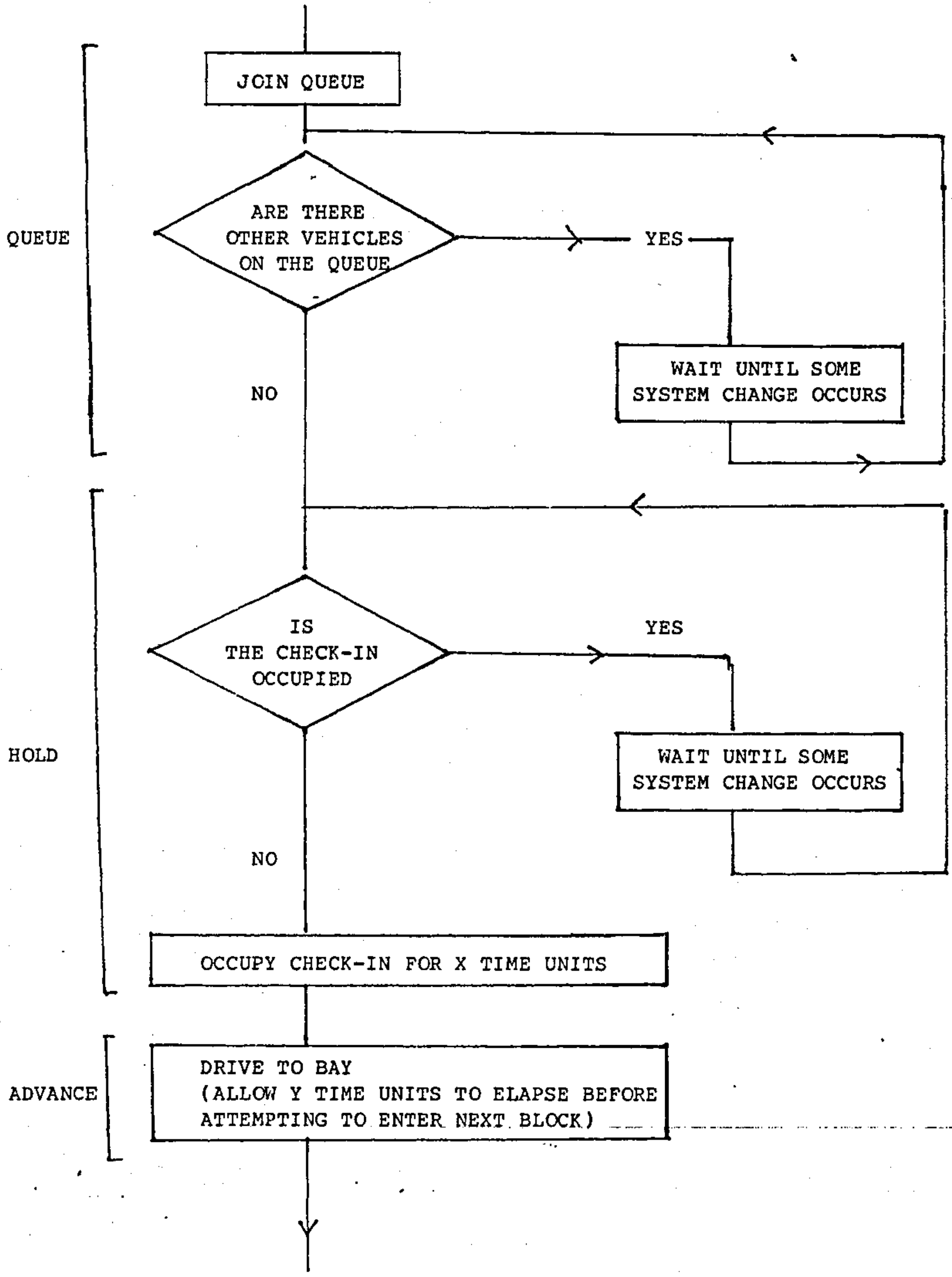
ADVANCE TIME (Y)

The blocks here have been expanded as compared to Table 6.1. The second block signifies that an entering TRANSACTION should hold the FACILITY called "CHECKIN" for a time period X. Note that a service centre such as a check-in is known in GPSS as a FACILITY. If there has been no previous reference to a FACILITY called "CHECKIN" then this HOLD statement also defines "CHECKIN" as the name of a FACILITY as the word which appears after HOLD must be the name of a FACILITY.

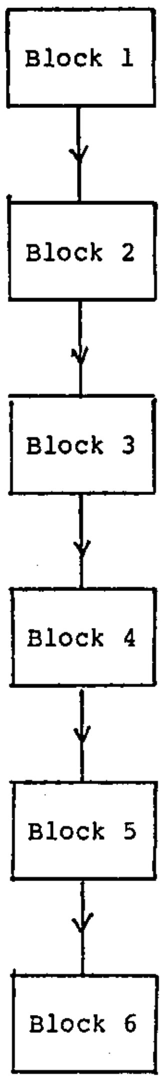
Suppose that a TRANSACTION enters the "QUEUE" block. It should next enter the "HOLD" block. If however at the time the TRANSACTION wants to enter the HOLD block another TRANSACTION is occuping the check-in then the former TRANSACTION remains in the QUEUE block until the other TRANSACTION by leaving the HOLD block and entering the ADVANCE block frees the FACILITY. Once the FACILITY (check-in) is free the TRANSACTION leaves the QUEUE block and enters the HOLD block. It remains here for time X (how this time is conveyed is described in Unit 5, for now assume x is a number i.e. a constant or fixed time) blocking any subsequent TRANSACTIONS from entering the HOLD block until the duration X has elapsed. Thus entry to and exit from blocks may be conditional. A TRANSACTION cannot enter a HOLD block if the FACILITY named in the HOLD block is occupied i.e. entry to a HOLD block is conditional. Any number of TRANSACTIONS May be in a QUEUE block or ADVANCE Block simultaneously. Entry to those blocks is unconditional. The ADVANCE block is used to indicate a passage of time which in this case corresponds to the time required (Y) to drive from the check-in to the queue for the bay. The TRANSACTION simply remains in the ADVANCE Block for a time Y and then tries to enter the next block (not shown).

One could think of the GPSS programme as a type of treasure hunt, each block as a station and each TRANSACTION as a player who goes from station to station receiving instructions at each station. Some stations have doors which may be open or closed (conditional entry). Other stations have no doors, a player can always enter (unconditional entry). When a player enters a station he receives two pieces of information (a) how long he must remain in the station (the TIME field in GPSS, see page 109) and (b) which station he should go to next (the GO TO field in GPSS, see page 109). He must wait until the specified time has expired, then he checks (by phone!) if the door to his next station is open, if it is he goes there. If it is not he remains where he is until the door to his next station opens. If there is no time information he may try to leave immediately. If there is no destination information he simply tries the neighbouring station.

The diagram below shows the procedures corresponding to each of the three blocks. Note that each block although apparently simple may correspond to quite a complex subroutine.


JOIN QUEUE
QUEUE
ARE THERE OTHER VEHICLES ON THE QUEUE
YES
WAIT UNTIL SOME SYSTEM CHANGE OCCURS
NO
IS THE CHECK-IN OCCUPIED
YES
HOLD
WAIT UNTIL SOME SYSTEM CHANGE OCCURS
NO
OCCUPY CHECK-IN FOR X TIME UNITS
ADVANCE
DRIVE TO BAY (ALLOW Y TIME UNITS TO ELAPSE BEFORE ATTEMPTING TO ENTER NEXT BLOCK)

If a student is familiar with a programming, language such as FORTRAN this idea of TRANSACTIONS being held up at blocks will appear strange and perhaps confusing.



Consider the opposite sequence of blocks. If each represented an instruction in FORTRAN and assuming no "go to" instructions (jumps) and no loops then one would expect that the instructions 1, 2, 3, 4, 5, 6 would be obeyed in sequence and when 6 is obeyed the programme is finished. A GPSS programme does not work this way. Blocks are not instructions to be obeyed in sequence. Each block is a sub-routine which is activated by an entering TRANSACTION. One TRANSACTION may be in block 1 and another TRANSACTION simultaneously in block 4. A programme may consist of hundreds of blocks and there may be hundreds of TRANSACTIONS proceeding simultaneously through the block structure. One should ensure that students understand the difference in progression through a FORTRAN type programme and a GPSS programme as failure to do so can cause much confused thinking.

For some blocks entry by a TRANSACTION is unconditional e.g. a TRANSACTION can always enter a QUEUE block. For others entry may be conditional e.g. a TRANSACTION can enter a HOLD block only if the associated FACILITY is free. A "GENERATE" block can never be entered by a TRANSACTION. TRANSACTIONS originate at a GENERATE block and leave it but cannot enter it.

6.2.3 Blocks

The general structure of blocks will be described and then each of the block types used in Table 6.1 will be described in detail. For allowable names including labels see the GPSS mini manual provided, but take note that the first character in a name or label must be alphabetic.

A statement describing a block may consist simply of a block name which indicates its function e.g. TERMINATE but more usually it contains other information some of which may be optional. A statement describing a block in general consists of five sections or fields some of which may be omitted. They are as follows LABEL, BLOCK TYPE, DATA, TIME INFORMATION, ROUTING INFORMATION

1. LABEL: This is used to identify a block if one needs to jump to that block from a block other than the preceding one. Thus one might have an instruction "GO TO A". This would indicate to a TRANSACTION that it should go to the block which is labelled A. Labels are optional.

2. BLOCK TYPE: This is compulsory and defines the function of the block. It is a name such as those in Table 6.1.

3. DATA: Depending on the block type some data may be required e.g. for a GENERATE block one must indicate the time at which the first TRANSACTION should be generated and may indicate the total number of TRANSACTIONS to be generated. The data may consist of one number or several numbers depending on the block type, or it may be a name.

4. TIME INFORMATION: This is a very important part of the block information and needs to be considered carefully. It specifies the processing time for the block. By this of course is not meant the computer processing time but the simulated time. For instance for the HOLD block corresponding to "check-in is performed" in Table 6.1 it would specify how long the check-in takes. Two formats are possible for the time information. The most common is TIME (X). If "X" is a numeric such as 10 then that is the processing time. However "X" may be a reference to a cumulative probability distribution defined among the function definition statements at the beginning of the programme as described in Unit 5. In this case a random value from that distribution will be selected and used as the processing time. The other possible format is TIME (a, b) in which case the processing time is an integer in the range a - b to a + b (inclusive). The programme will select one such integer, all having equal probability of being selected. The time is said to be uniformly distributed in the range [a - b, a + b].

5. ROUTING INFORMATION: The simplest format is GO TO (A). This instructs the TRANSACTION on leaving the block to go to the block labelled A. If this is not possible the TRANSACTION will wait in the current block until it is possible. Another possible format is GO TO (A, B, C, D....). In this case the TRANSACTION will go to A if possible. If this is not possible it will go to B. If that is not possible it will go to C and so on. If none are possible it will keep trying in rotation until it is possible or until the programme stops. These are the only formats used in the models to be described. Others are possible e.g. it is possible to direct a TRANSACTION to one of several possible destinations with specified probabilities.

The five fields within a statement describing a block are separated by one or more blanks. Items within a field are separated by commas. Blanks should not appear within a field. Thus the structure of a statement is as below

1	2	3	4	5
LABEL	BLOCK TYPE	DATA	TIME INFORMATION	ROUTING INFORMATION

2 is compulsory

1, 5 are optional and not necessary for the model in this unit

3, 4 are compulsory for certain block types

Thus for two of the statements on page 104 the structure is as follows

BLOCK TYPE	DATA	TIME INFORMATION
HOLD	CHECKIN	TIME (X)
ADVANCE		TIME (Y)

Note that the HOLD block required in the DATA field the name of the FACILITY to be held. The ADVANCE block required no data.

6.2.4 Program Blocks in Detail

The blocks required for the program (see Table 6.1) will now be described in detail.

6.2.4.1 GENERATE

This block generates TRANSACTIONS (lorries in the example). A TRANSACTION so generated remains in the block for a time period specified by the programmer. It then leaves the block and another TRANSACTION is generated i.e. each time a TRANSACTION leaves another is generated immediately unless n_2 have been generated (see below). The time for the first TRANSACTION to be generated is specified by the programmer. From the point of view of the model the significant time is the time at which the TRANSACTION leaves the block as that is the time at which it enters the system.

The format for the statement is:

BLOCK TYPE	DATA	TIME	ROUTING
GENERATE	n_1, n_2, n_3	TIME (a)	GO TO (L1)

n_1 = time at which first TRANSACTION should be generated (compulsory)
n_2 = maximum number of TRANSACTIONS to be generated (optional)
n_3 = priority to be attached to the TRANSACTION (optional)

a = some expression defining the time to be spent in the block by the TRANSACTION before it leaves and another is created, this corresponds to the "inter-arrival time" in the example.

L1 = label of the block to which this TRANSACTION is routed

Omission of data :

If n_2 is to be omitted but n_3 included two commas must follow n_1 viz. $n_1,,n_3$

If n_2 is omitted the GENERATE statement will continue to generate TRANSACTIONS indefinitely. If n_3 is omitted all TRANSACTIONS will have zero priority. If TIME (a) is omitted the TRANSACTION attempts to leave the block immediately. If GOTO (L1) is omitted it goes to the next block in sequence.

Simultaneous GENERATE Blocks

Consider the following two GENERATE blocks:

```
GENERATE 1,100 TIME (10) GO TO (L1)
GENERATE 200,100 TIME (20) GO TO (L1)
```

The two GENERATE statements operate together. The first one generates 100 TRANSACTIONS at 10 minute intervals starting at time 1. The second GENERATE statement starts generating TRANSACTIONS at time 200 and also generates 100 TRANSACTIONS the inter-arrival times being 20 minutes. Of course the TRANSACTIONS generated by the two statements overlap i.e. after time 200 both statements are generating TRANSACTIONS.

The GENERATE statements above include gaps (for ease of reading) which should not appear in statements coded for the computer e.g. the first one should be

```
GENERATE     1,100     TIME(10)     GOTO(L1)
```

6.2.4.2 HOLD

The purpose of the HOLD block is to take possession of a FACILITY such as a check-in in our example. This FACILITY must have a name which can be defined in the HOLD block. The format of the block is:

BLOCK TYPE	DATA	TIME
HOLD	name of FACILITY	TIME (X)

e.g. HOLD CHECKIN TIME (X)

If the name CHECKIN has not been encountered previously then this statement defines it as a FACILITY at the time at which the model is interpreted i.e. before generation of any TRANSACTIONS. During operation of the model when a TRANSACTION successfully enters this block it, in effect, sets a flag to denote that the FACILITY called CHECKIN is occupied. It will be occupied for the time defined in the time field i.e. X in the above example. If a second TRANSACTION arrives at the block HOLD CHECKIN TIME (X) while the FACILITY is occupied, it cannot enter the block and will remain in the block from which it is trying to depart until the interval X has expired and the first TRANSACTION leaves the HOLD block.

6.2.4.3 ADVANCE

This block is generally used to specify processing time. The format is:

BLOCK TYPE	TIME
ADVANCE	TIME (X)

An entering TRANSACTION will remain in the block for a time specified by the value of X. This block is sometimes used without a time field as a "dummy" instruction i.e. an instruction which does nothing. It may just be somewhere for a TRANSACTION to wait until it can enter the next block.

Consider the pair of statements

```
GENERATE 1 TIME (10)
HOLD CHECKIN TIME (X)
```

The purpose of the GENERATE statement is presumably to generate a TRANSACTION every ten minutes (assuming the time units are minutes). However a TRANSACTION cannot leave the block unless CHECKIN is free and this will prevent the generation of the next TRANSACTION so TRANSACTIONS will not be generated every 10 minutes. To ensure generation of TRANSACTIONS at 10 minute intervals one could use the following sequence.

```
GENERATE 1 TIME (10)
ADVANCE
HOLD CHECKIN
```

Entry to the ADVANCE block is unconditional so a TRANSACTION is generated every 10 minutes and proceeds to the ADVANCE block. Any number of TRANSACTIONS can be in the same ADVANCE block simultaneously (subject to overall program capacity).

6.2.4.4 SEIZE AND RELEASE

These two blocks are an alternative to the HOLD block.

Thus the following (denoted I and II) are equivalent:

```
I       HOLD CHECKIN TIME (X)

II      SEIZE CHECKIN

        ADVANCE TIME (X)

        RELEASE CHECKIN
```

As I is simpler it is preferable in the above example. However the SEIZE and RELEASE blocks are necessary if one wishes the TRANSACTION to do something else while still holding the FACILITY as in the following example.

```
SEIZE A

HOLD B TIME (X)

ADVANCE TIME (Y)

RELEASE A
```

The TRANSACTION first occupies the FACILITY called A and then the FACILITY called B when the latter becomes available. It occupies B for a period X while still retaining A. After a period X it release B. After a further period Y it releases A. Thus the total time for which A is occupied is X + Y + any waiting time for B. It might seem that it is never necessary to wait for B since if B is occupied then the occupying TRANSACTION would have first occupied A and the current TRANSACTION could not have entered the SEIZE block. However elsewhere in the programme it might be possible for a TRANSACTION to occupy B without first having occupied A.

As an example A could be a chair in a hair dressing salon and B a hairdresser. A customer wants his/her hair washed and dried. The TIME (X) is the time the hairdresser spends washing it. When finished the customer remains alone for a time Y while his/her hair dries but still occupying the chair. A customer having occupied the chair might still have to wait for a hairdresser who might be otherwise occupied.

The sequence would be as follows:

BLOCKS	PHYSICAL ACTION
SEIZE A	OCCUPY CHAIR WAIT FOR HAIRDRESSER
HOLD B TIME(X)	RETAIN HAIRDRESSER FOR X MINUTES RELEASE HAIRDRESSER
ADVANCE TIME(Y)	OCCUPY CHAIR FOR A FURTHER PERIOD OF Y MINUTES
RELEASE A	RELEASE THE CHAIR

This particular model is expanded on in Appendix B

6.2.4.5 QUEUE

The purpose of this block is to cause the simulator to collect information on waiting TRANSACTIONS i.e. TRANSACTIONS in this queue. Each QUEUE has a distinct name. The information gathered will be printed out on successful completion of the program.

The format is

BLOCK TYPE	DATA
QUEUE	name of queue, n

n defines the number of items to be entered onto the QUEUE at each entry. If n is omitted it is understood to be 1 and this is the usual case, however one might want n to be other than 1. Suppose a "TRANSACTION" is a bus. Each time one arrives a number of people are disgorged and simultaneously join a queue. n would be the number leaving the bus and joining the queue.

In the sequence of Table 6.1 the QUEUE block also serves the dummy purpose described for the ADVANCE block on page 114 .

Now consider the function of QUEUE in the following sequence of blocks. (Assume that times are in minutes).

```
GENERATE 1 TIME (10)
QUEUE CHECKIN.LINE
HOLD CHECKIN TIME (3)
```

TRANSACTIONS will leave the GENERATE block every 10 minutes. When a TRANSACTION leaves the GENERATE block it enters the QUEUE block unconditionally. The simulator updates the records for this QUEUE. If there are other TRANSACTIONS in the QUEUE block the TRANSACTION will wait till it reaches "the head of the queue". It then tries to enter the next block HOLD CHECKIN. If the FACILITY called "CHECKIN" is occupied it will wait until it is free. When "CHECKIN" is free the TRANSACTION moves into the HOLD CHECKIN block and the records for the queue "CHECKIN.LINE" are updated. The TRANSACTION remains in the HOLD block for three minutes. It then moves to the next block and CHECKIN is freed so that the next TRANSACTION (if any) can leave the QUEUE block and enter the HOLD block.

6.2.4.6 TERMINATE

This block causes a TRANSACTION to be removed from the model when its existence is no longer required e.g. when a lorry leaves the building the corresponding TRANSACTION can be removed. The program can hold only a limited number of "live" TRANSACTIONS at any time depending on allocation of computer space so when no longer required they should be removed. There are two formats:

```
TERMINATE
   or
TERMINATE, R
```

If the latter format is used the TRANSACTION is counted. A limit on the total number to be so counted can be set as a means of terminating the simulation. If the former format is used the TRANSACTION so terminated is not included as part of the final count.

A full set of blocks for the model on page 103 would be as follows

```
GENERATE          1                   TIME(A)

QUEUE             CHECKIN.LINE

HOLD              CHECKIN             TIME(B)

ADVANCE                               TIME(C)

QUEUE             UNLOAD.LINE

HOLD              UNLOAD.BAY          TIME(D)

TERMINATE, R
```

How the time durations A,B,C,D are specified will be discussed in Unit 5. How the model would be operated will be described in Unit 6 (as detailed above the GENERATE statement causes the model to process an unlimited number of TRANSACTIONS starting at time 1, some control information is required to specify when simulation should stop).

CHAPTER 7

UNIT FIVE OF TEACHING PACKAGE

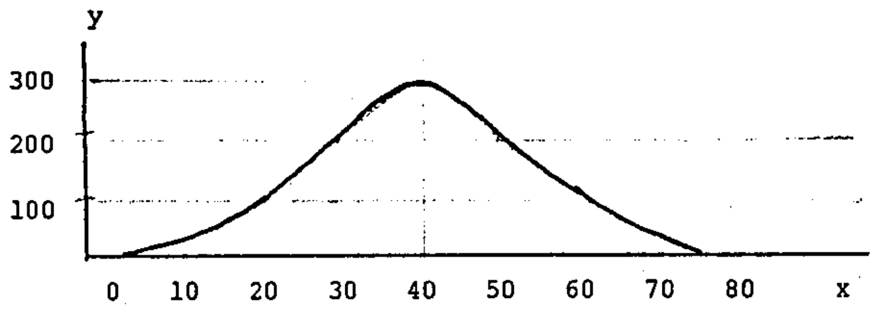
7.1 INTRODUCTION TO UNIT 5

This unit describes in detail how processing or elapsed times are specified. In particular there is a discussion of functions in general and the concept of FUNCTION as used in GPSS. Data definition statements are described in detail particularly the "FUNCTION" and "FUNCTION,C" data definition statements.

On completion of this unit students should be able to take any function specified in words, graphically or as a table and code the appropriate FUNCTION definition statement.

To test their understanding test exercises such as the following can be set.

1. Write a FUNCTION definition statement for y as defined in the graph below.



2. The cost of delivering goods depends on the number of vehicles required and on the weight of goods contained in each vehicle. Assume each vehicle can take up to a maximum of 1000 kg. The cost of each vehicle is $200 per run and the cost of loading and offloading goods is $0.10 per kg. Let "LOAD" equal total weight of goods to be transported and "COST" equal the total delivery cost.

   Assume that "LOAD" will be a whole number of kgs. in the range 1 to 4000.

   Write a FUNCTION definition statement for "COST".

3. A building contains 10 parking places, four on the first floor (numbered 1-4), four on the second floor (numbered 5-8) and two on the third floor (numbered 9-10). An arriving vehicle is twice as likely to want the first floor as the second floor and twice as likely to want the second floor as the third floor. Within a particular floor all spaces are equally desirable.

(a) Construct a cumulative probability distribution for y where y is the number of the parking place to which an arriving vehicle goes.

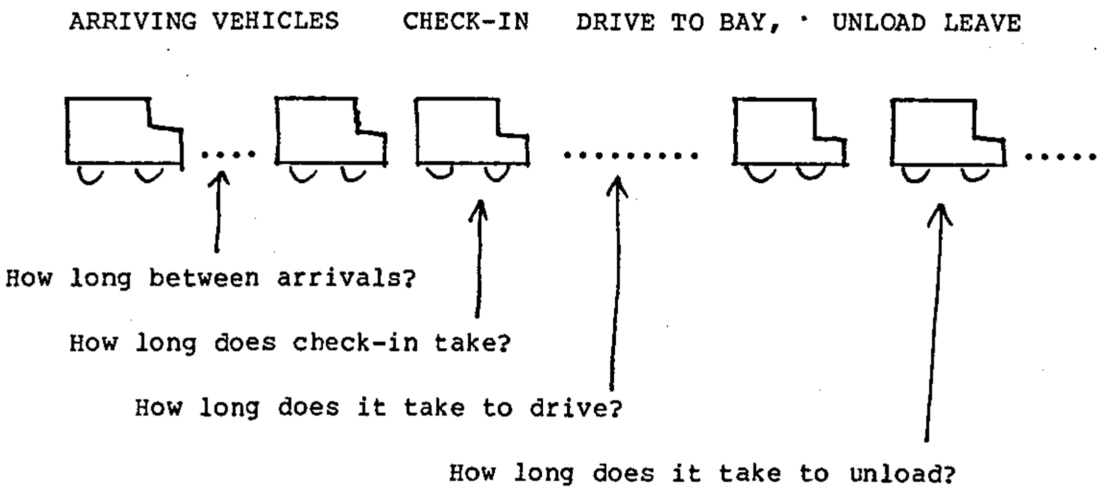
(b) Write a FUNCTION definition statement for y using the random number output from one of the random number generators as the independent variable which will produce y values with the probability distribution obtained in (a).

4. Write FUNCTION definition statements for the functions in Tables 5.7, 5.11, 5.13 of Unit 3.

7.2 UNIT FIVE - FUNCTIONS

7.2.1 <u>Time Distributions</u>

In the model used in Unit 4 four elapsed times are required viz., the time between arriving vehicles, the time required to check-in, the time required to drive to the bay and the time required to unload.


ARRIVING VEHICLES
CHECK-IN
DRIVE TO BAY,
UNLOAD LEAVE
How long between arrivals?
How long does check-in take?
How long does it take to drive?
How long does it take to unload?

Such elapsed times are specified in a TIME field as for instance in

```
        HOLD        CHECKIN        TIME(X)
```

It is important to understand the nature of X. It can be a constant which would be very restricting or it can be a variable and if so one needs to know how to specify the variation. In fact two formats are possible.

```
        Format (i)         TIME(a, b)
        Format (ii)        TIME(X)
```

If format (i) is used the computer will select as processing time an integer in the range a - b to a + b (inclusive) all values having equal probability of being selected. This selection is done anew each time a TRANSACTION enters the block.

If format (ii) is used the processing time depends on how X is specified. If X is specified as a number (which must be an integer) then that is the processing time which will be used for every TRANSACTION entering the block i.e. the processing time is constant. The most useful format however is when X is the name of a function. In this case the function is evaluated whenever a TRANSACTION enters the block and the value so obtained is used as the processing time. How the function is to be evaluated must be specified on a FUNCTION definition statement.

7.2.2 Data Definition Statements

A FUNCTION definition statement is just one type of data definition statement. Just as a statement defining a block has up to five sections or fields a data definition statement has three. These are NAME, TYPE, DATA.

NAME is a name selected by the programmer to be associated with the entity being defined.

TYPE is a word describing the type of data definition statement e.g. FUNCTION.

DATA contains the data being specified by the programmer. Note data definition statements do not have LABELS, TIME fields or GOTO fields. This is because they do not form part of the model proper. They are not "entered" by TRANSACTIONS. They are simply noted and recorded by the simulator prior to the start of the simulation. In the GPSS mini-manual provided there is a section which deals with data definition statements in detail.

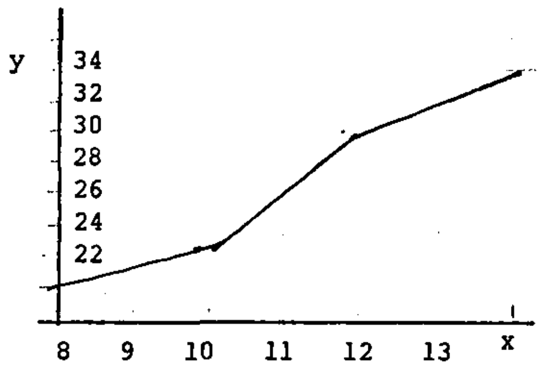
7.2.3 Functions

The use of functions has been mentioned briefly in Unit 2 and in Appendix A. However it is convenient to recall here what a function is. If one says that y is a function of x one means that a relationship exist between x (e.g. the load in a lorry) and y (e.g. the time to unload) such that if x is known y can be determined from the relationship. This relationship could be a simple mathematical one e.g. $y = 2x + 4$ or it could exist in the form of a table or graph as below.

TABLE 7.1

x	y
8	20
10	23
12	30
14	34

GRAPH 7.1



Below are some examples of x and y pairs. Note that x always stands for the variable which is specified (known as the independent variable) and y for the one which is calculated (known as the dependent variable)

TABLE 7.2

Dependent Variable	Depends On	Independent Variable
y = drive time	distance travelled	x = distance
y = time to unload	size of load	x = size of load
y = inter-arrival time	chance	x = a random number

Note that if y depends on chance x is a random number. The relationship will be as specified in Unit 2 (see example 4.3). For the model of Unit 4 four functions will be required corresponding to the four elapsed times referred to on page 79.

In practice a dependent variable can depend on more than one independent variable. Thus in the example in Table 7.2 "drive-time" should not in fact depend only on "distance travelled" but also on speed, traffic conditions etc. Such functions are called functions of several variables. GPSS functions are functions of a single variable and discussion will be limited to such functions. In fact in the model previously discussed "distance travelled" i.e. check-in to bay is constant and one would expect drive time to be constant except for the fact as explained above that it depends on other factors. Frequently when a dependent variable depends on several factors which are difficult to assess we say it depends on chance! That is to say all of the independent variables are subsumed in "chance". For instance if one tosses a coin one says that the result (head or tail) is a matter of chance. But is it? In fact it depends on a number of factors such as the side showing when it is tossed the force with which it is tossed, air resistance, air currents, etc. It is difficult to measure these independent variables so one lumps them all together and calls them "chance". The form of the dependence on chance is established by observation. Unit 3 described how one might do this. One observed a number of vehicles and obtained the relative frequency of different drive times. These relative frequencies are used to estimate probabilities. The cumulative probabilities are made synonymous with random numbers in the range zero to one. Thus for y = drive time the independent variable x is a random number (unlike the example in Table 7.2). The functional relationship is expressed in Table 5.11 where y is the drive time and x is the cumulative probability.

A function definition statement in GPSS is simply a table such as 5.11 coded in the prescribed form as described in the next section.

7.2.4 Use of FUNCTION in GPSS

What does the simulator do when one refers to a FUNCTION?

Whenever in a program the analyst refers to a FUNCTION by name the simulator will calculate a y value corresponding to the current x value according to a rule specified on the data definition statement. The data definition statement must describe:

(a) The name of the FUNCTION, this is in fact the name of the dependent variable y. For example it could be DRIVE.TIME or DURATION. This is the first field of the data definition statement.

(b) The name of the independent variable x. For example this could be WEIGHT or it could be the name of a random number generator. This forms part of the third (DATA) field of the data definition statement. If x has a name such as "WEIGHT" then this variable "WEIGHT" must have a value when reference is made to the FUNCTION. If several references are made to the FUNCTION, "WEIGHT" might have different values at the times of these references. If x is the name of a random number generator then each time a reference is made to the FUNCTION a value for x is obtained using the specified random number generator.

(c) The actual distribution i.e. the rule for converting the x value to a y value. This is in the form of a table such as Table 5.17 reproduced on the following page as Table 7.3. It forms part of the DATA field also.

TABLE 7.3

y	x
45	0.00
55	0.40
65	0.80
75	1.00

x = a random number between 0 and 1

y = Drive Time

Note that if x is a random number the procedure described above is exactly the same as that described in Unit 2 on page 70 for obtaining a sample service-time.

Example 7.1

Suppose that one wishes to be able to calculate the time required to unload a vehicle from the weight of the load given the sample observations below.

TABLE 7.4

Weight in kgs x	Unload Time y
1000	20
1700	37
3000	45
4000	58
5000	60

The independent variable is to be called "WEIGHT" and the dependent variable is to be called "DURATION".

The FUNCTION definition statement would be as follows (second row):

NAME FIELD	DESCRIPTION FIELD	DATA FIELD		
DURATION	FUNCTION,C	V$WEIGHT,1000,20	1700,37 4000,58	3000,45 5000,60
dependent variable name	type of data definition statement	independent variable name	the distribution	

The letter "C" after FUNCTION denotes that it is a continuous function. The significance of this will be explained shortly.

V$HEIGHT The V here denotes that what follows is a variable name. The dollar sign ($) is merely a separator. This notation is described in the GPSS mini-manual.

Suppose now that somewhere in the program a reference is made to the FUNCTION called "DURATION", the simulator would automatically calculate a value for this FUNCTION which would depend on the current value of the independent variable called "WEIGHT".

If "WEIGHT" had a value 2000 when reference was made to "DURATION" the simulator would look down the Table 7.4 to find 2000.

TABLE 7.4 (extract)

x	y
1700	37
3000	45

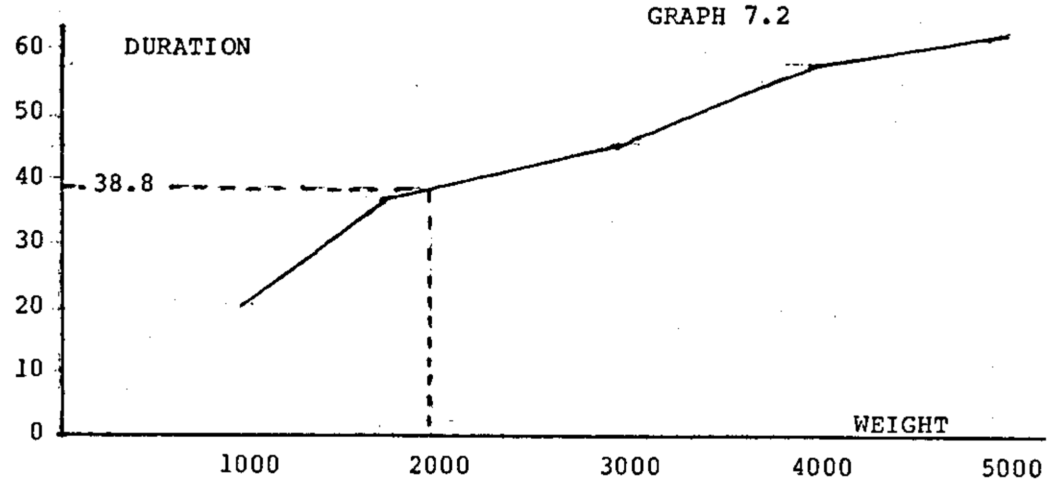
2000 is not specified but it is between 1700 and 3000 which are specified. The simulator then "interpolates" between 37 and 45 as follows:

$$\text{WEIGHT} = 37 + \frac{(2000 - 1700)}{(3000 - 1700)}(45 - 37) = 38.8$$

So if WEIGHT has a value 2000 when reference is made to DURATION, DURATION will automatically be given the value 38.8.

The above method of interpolation is called linear interpolation.It is equivalent to plotting the points on a graph and joining them by straight lines as below:

GRAPH 7.2



7.2.5 <u>Continuous and Discrete Functions</u>

Consider the following extract from Table 7.4

TABLE 7.4 (extract)

x	y
1000	20
1700	37
.	.
.	.

The above table indicates that when x is 1000, y is 20 and when x is 1700, y is 37. It does not state what value y has if x is between 1000 and 1700. At least three reasonable interpretations are possible:

(a) As x increases from 1000 to 1700 y increases from 20 to 37 proportionately i.e. if x is half-way between 1000 and 1700 then y will be half-way between 20 and 37. This is linear interpolation as just decribed.

(b) As x increases from 1000 to 1700 y remains at 20 and changes in a jump (or step) from 20 to 37 when x reaches 1700, i.e. y equals 20 for all values of x between 1000 and 1700 (excluding 1700).

(c) As x increases above 1000 y jumps to 37 and remains at 37 until x passes 1700.

y as described in (a) is a continuous function (gradual change) y as described in (b) and (c) is a discrete function (sudden change) (b) and (c) are basically the same it being simply a matter of convention whether one regards the change from 20 to 37 for y as taking place at x=1000 or at x=1700.

If the letter C after FUNCTION had been omitted i.e. if the statement was as follows:

```
WEIGHT FUNCTION V$HEIGHT,1000,20  1700,37  3000,45  4000,58
                             5000,60
```

then the interpolation would not be performed and the table would be interpreted as in Table 7.5 below. This corresponds to (c) above

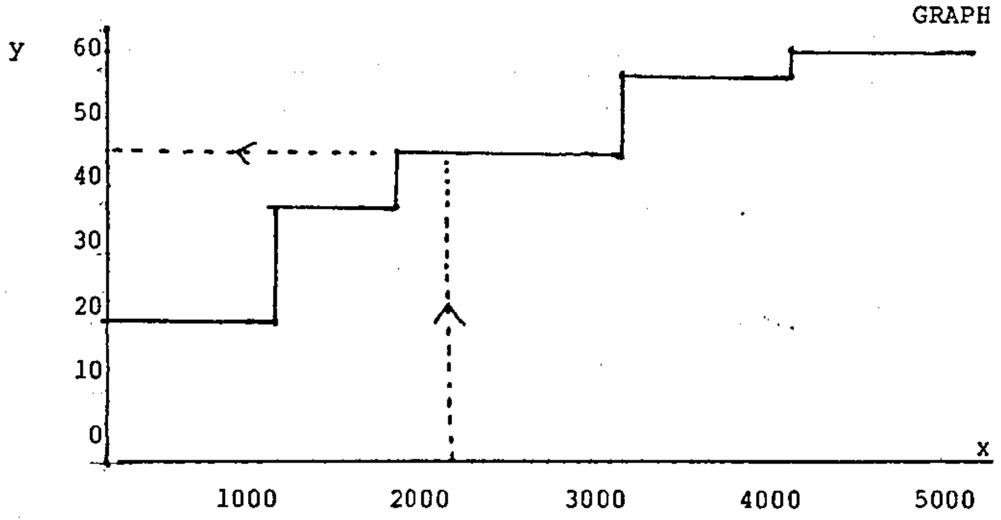
TABLE 7.5

x	y
Up to and including 1000	20
above 1000 up to and including 1700	37
above 1700 up to and including 3000	45
above 3000 up to and including 4000	58
above 4000 up to and including 5000	60

Thus if WEIGHT = 2000 corresponding DURATION = 45

This is known as a discrete function and is shown graphically below:

GRAPH 7.3



7.2.6 Using a Random Number Generator as the Independent Variable

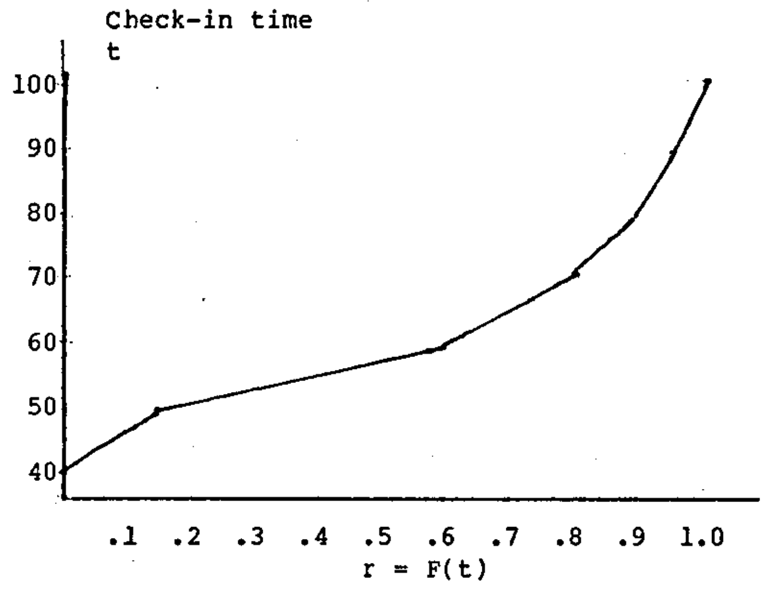
Example 7.2

Suppose that whenever one refers to a FUNCTION Called "CHECKINTIME" one wishes to obtain a random value of the check-in time according to the distribution speecified in Table 5.9 or Graph 5.2 reproduced below as Table 7.6 and Graph 7.4. In this case one would wish to use a random number as the independent variable.

TABLE 7.6

t	F(t)
40	.00
50	.15
60	.60
70	.80
80	.90
90	.95
100	1.00

GRAPH 7.4



The FUNCTION definition statement would be :

```
CHECKINTIME FUNCTION,C RF$1,0,40 .15,50 .6,60 .8,70 .9,80 .95,90
1,100
```

RF$1 refers to one of the random number generators. These are referenced as RF$1, RF$2, RF$3 etc. In principle one only needs a single random number generator however it is convenient to have several. One might wish to run a program several times using different sample data each time. This can be achieved by using a different random number generator each time one runs the program. Also if one has several functions it may be convenient (although not necessary) to use a different random number generator for each. This makes manual checking easier, for instance if one wanted the check-in time of the fifth vehicle this would have been obtained from the fifth random number in the RF$1 series which would not have been the case if the same generator had been used for other purposes. The above statement specifies that whenever a CHECKINTIME value is required the simulator will select a random number from the "black box" referenced as RF$1 and use that as the r value in Graph 7.4 to get a corresponding CHECKINTIME. Each time one refers to CHECKINTIME a different random number is selected viz. the next one in the series of random numbers, and hence a different (in general) time is obtained.

CHAPTER 8

UNIT SIX OF TEACHING PACKAGE

8.1 INTRODUCTION TO UNIT 6

At this stage the student should be able to write the program for the model of Unit 2. This could be set as an exercise before presenting the suggested program below. The control statements have not yet been described so of course the student would not yet be able to make the program run. The author discovered a software bug in connection with the JOB control statement. In spite of what the manual states J must not be in column 1 i.e. the word JOB should commence in column 2. It is not known if this bug exists in all versions of GPSS or only that for UNIVAC 1100. If J is in column 1 no error is detected but various program statements (the first FUNCTION statement, the first VARIABLE statement etc. and the first block) are not interpreted.

Following the program are some general notes on GPSS instructions both those which have been introduced and those yet to be introduced as the models are developed.

There then follows a discussion of how scheduling i.e. timing of events is accomplished by GPSS. If students are familiar with a language such as BASIC or FORTRAN it would be a good idea to get them to write the program for the model of Unit 2 in that language so as to appreciate the amount of "housekeeping" that is avoided by using GPSS. They could be asked to do this using either or both of the following methods of time updating

(1) Update the model at equal intervals (i.e. at t=1, t=2, t=3 etc.)
 or

(2) Update the model only when events are scheduled to occur

In the simple model already discussed the events which can occur are the following, (a) a vehicle arrives and joins the check-in queue (b) a vehicle commences checking in (c) a vehicle finishes checking in and starts to drive to the bay (d) a vehicle arrives at the queue for the bay (e) a vehicle starts unloading (f) a vehicle finishes unloading and leaves.

Using the first method of time updating, assuming the day starts at 9.00 a.m. and that the time interval to be used is 1 minute one would proceed as follows :

Start clock at 9.00 a.m.	Are any of the events (a-f above) due to happen now? If so, update model
Advance clock to 9.01 a.m.	Repeat above
Advance clock to 9.02 a.m.	Repeat above

•
•
•

Using the second method of time updating one would proceed as follows

At what time is the first event scheduled? Advance the clock to this time and update the model.

At what time is the next event scheduled? Advance the clock to this time and update the model.

etc.

Method 2 is more appropriate for discrete simulation where one is only interested in discrete events and not in what happens in between e.g. we want to know when a vehicle leaves the check-in and when it arrives at the bay but not points in between. Method 1 would be appropriate for a continuous simulation where relevant changes occur continuously and not at unequal intervals. Nevertheless Method 1 can be used for discrete simulation especially if events occur frequently relative to the basic time interval.

Students could be divided into groups of about 5 to do the exercise in section 8.2.5. This should definitely be done if students cannot write their own programs in BASIC or FORTRAN in order to demonstrate that they appreciate the details of scheduling.

Finally there is a description of the standard output provided by GPSS, and a discussion of the operation of a simulation model. It was felt appropriate to include that here for students who only wanted an introduction to simulation and who would not be proceeding beyond Unit 6.

This section provides opportunity for discussion and student participation e.g. they could be asked to provide examples of terminating and continuous systems and asked what information they think would be useful to obtain from simulations of various systems.

The principal new G.P.S.S. concept introduced in this chapter is that of "event chains" which are important in scheduling. Three control statements are introduced viz.

END, JOB, START

To test understanding of this unit students could be asked to do the following test exercises.

(1) Write a program for the hairdressing salon model discussed on page 116. They should be encouraged to add as many details as they can to the model to make it realistic. With the limited knowledge so far gained of GPSS not much can be done. However it is of value for them to discover what they would like to do but cannot and hence infer what extra facilities the language needs. Appendix B gives suggestions but mostly these cannot yet be implemented.

(2) Each student could be asked to run the program on page 145 for 500, 1000, 2000, 5000, 10,000 vehicles i.e. 5 separate runs using different random numbers. Each student should also use different random numbers so that their results can be collated. If there are say 20 students then one would end up with 20 samples of 500, 20 samples of 1000 etc. and one could demonstrate the increase in accuracy with sample size (see page 173).

(3) Expand the model slightly by having two vehicle types with different arrival rates (requires two separate GENERATE statements and two "ARRIVAL.TIME" functions. These would share the same check-in but have separate loading bays.

Basically all that is required is to duplicate the seven blocks i.e. have two segments, 7 blocks in each. The only changes in the second segment would be the names of the inter-arrival time function, the bay queue and the bay. Of course the students should be allowed to discover this for themselves.

8.2 UNIT SIX - OPERATION OF THE PROGRAM

8.2.1 The GPSS Program for the Model of Unit 3

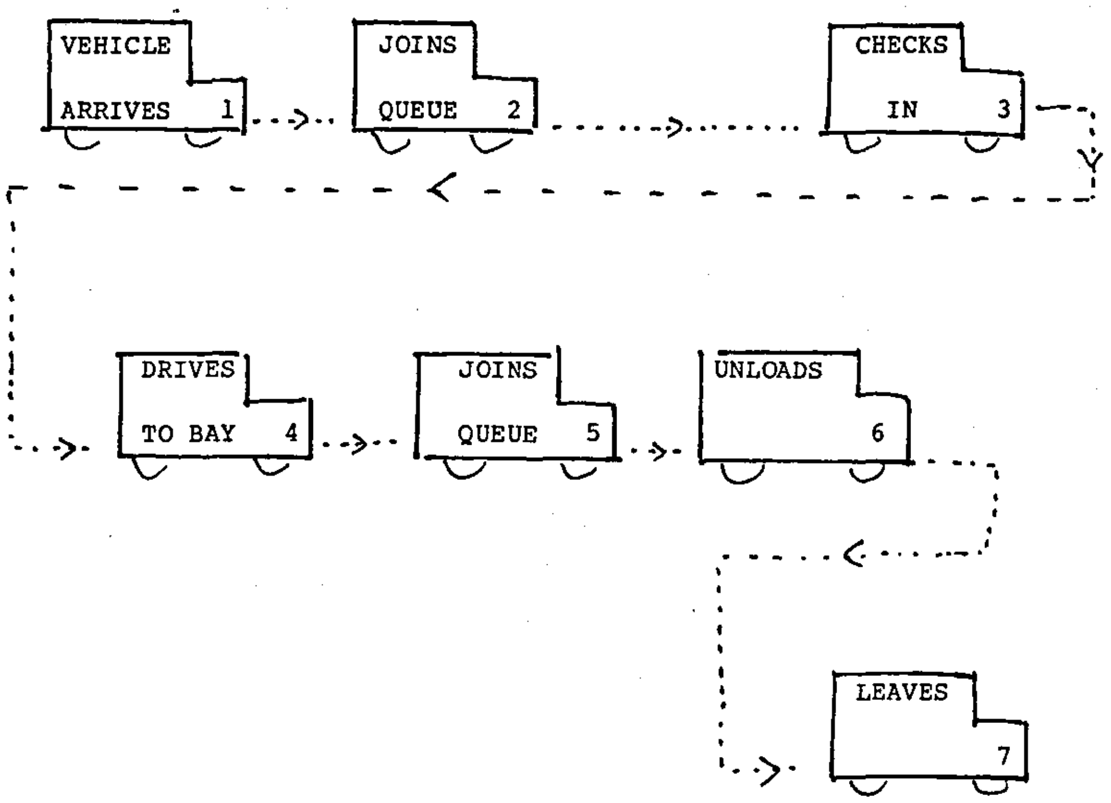
A GPSS program can now be written for the system described in Unit 3. Of course nothing has been said yet about what output will be produced, that will be discussed later.

At this stage it is convenient to recall the system described in Unit 3 (see next page for schematic representation). Vehicles arrive at a warehouse entrance at a specified rate (inter-arrival time distribution). There they join a queue for a "check-in" procedure. The check-in time distribution is known. Having checked in they drive to an unloading bay. The drive-time distribution is also known. Having arrived at the unload bay they join another queue for unloading. They are subsequently unloaded. The unload-time distribution is also known. Having been unloaded they leave.

The first thing to do is to define the four probability distributions required viz., the inter-arrival time, the check-in time, the time to drive from check-in to the bay and the time to unload. The four function definition statements follow. The data are obtained from Tables 5.7, 5.9, 5.11, 5.13

```
ARRIVAL.TIME FUNCTION,C RF$1,0,0   .25,300   .30,600   .35,900
.65,1200   .95,1800   1,2100
CHECKIN.TIME FUNCTION,C RF$2,0,40   .15,50   .6,60   .8,70   .9,80
.95,90   1,100
DRIVE.TIME FUNCTION,C RF$3,0,44  .1,48   .5,52   .8,56   .9,60
.95,64   1,68
UNLOAD.TIME FUNCTION,C RF$4,0,700   .2,725   .3,750   .55,755
.75,800   .95,825   .95,850   1,875
```

Schematic Representation



Blocks representing the above physical system

Block Number	Block Type	Data	Time Field
1	GENERATE	1	TIME(FN$ARRIVAL.TIME)
2	QUEUE	CHECKIN.LINE	
3	HOLD	CHECKIN	TIME(FN$CHECKIN.TIME)
4	ADVANCE		
5	QUEUE	UNLOAD.LINE	
6	HOLD	UNLOAD.BAY	TIME(FN$UNLOAD.TIME)
7	TERMINATE,R		

The above seven blocks together with the function definition statements constitute the model. The blocks are the same as in Table 6.1 but in more detail.

In addition to the function definition statements and the model statements (blocks) at least three model execution control statements are required. They are the JOB, START and END statements which will now be described.

JOB

This signifies the beginning of a simulation model and zeroes all previous statistics, TRANSACTIONS and blocks. It also zeroes clock times and initialises random number generators. It must be the first card of the model program.

START

This causes the simulator to begin the simulation. It may also convey information about the length of the simulation and what print out is required. For the format see the GPSS manual.

END

This causes the termination of the GPSS program and must be the last statement of the model.

There are other control statements apart from these three some of which will be encountered later as their need arises, but for many simulations these three suffice.

8.2.2 Order of Statements

In addition to the model itself certain instructions are required in order to summon the GPSS program from tape or whereever it is stored but these instructions do not form part of the model proper and will not be discussed. Their format can be obtained from the manual supplied with the software. These instructions have the symbol @ in column 1.

The entire batch of instructions appears as follows:

```
        @ . . . . . . . . . .

        @ . . . . . . . . . .

        ┌  JOB                                      Control statement
        │
        │  ARRIVAL.TIME FUNCTION,C . . . . .   ┐
        │                                      │    Data definition
        │  .                                   │    Statements
        │  .                                   │
MODEL   │  UNLOAD.TIME FUNCTION,C              ┘
        │  GENERATE . . . . .                  ┐
        │  .                                   │    Model
        │  .                                   │    Statements
        │  .                                   │
        │  TERMINATE, R                        ┘
        │
        │  START 1000                          ┐    Control
        │                                      │    Statements
        │                                      │
        └  END                                 ┘

        @ . . . . . . . . .
```

Note the suggested order of the statements within the model

First	JOB card
Next	Data definition statements
Next	Model statements or blocks
Next	Model control statements

Below is a copy of the program as printed out by the computer. The numbers 1-32 in the first column and the block numbers 1-7 are not coded they are inserted by the computer.

```
@HKU*USER.GPSS

GPSS 4.1    -06/04-12:41-(000)

 1        *
 2        *
 3         JOB
 4        *
 5        *
 6        * FUNCTION DEFINITION STATEMENTS
 7        *
 8        ARRIVAL.TIME FUNCTION,C RF$1,0,0 .25,300 .30,600 .55,900
 9        + .65,1200 .85,1500 .95,1800 1,2100
10        *
11        CHECKIN.TIME FUNCTION,C RF$2,0,40 .15,50 .6,60 .8.70 .9,80
12        + .95,90 1,100
13        *
14        DRIVE.TIME FUNCTION,C RF$3,0,44 .1,48 .5,52 .8,56 .9,60
15        + .95,64 1,68
16        *
17        UNLOAD.TIME FUNCTION,C RF$4,0,700 .2,725 .3,750 .55,755
18        + .75,800 .95,825 .95,850 1,875
19        *
20        *
21        *MODEL STATEMENTS
22        *
23      1      GENERATE 1 TIME(FN$ARRIVAL.TIME)
24      2      QUEUE CHECKIN.LINE
25      3      HOLD CHECKIN TIME(FN$CHECKIN.TIME)
26      4      ADVANCE TIME(FN$DRIVE.TIME)
27      5      QUEUE UNLOAD.LINE
28      6      HOLD UNLOAD.BAY TIME(FN$UNLOAD.TIME)
29      7      TERMINATE,R
30        *
31        * SIMULATE 2000 VEHICLES WITH PRINOUT AFTER EVERY 1000
32             START 2000,,1000
```

Note: The END statement which in the input was immediately following the START card, in the output appears after the final results are printed.

8.2.3 Note on GPSS Statements in General

The number of statements described so far is 11.

These are as follows:

JOB, START, END, FUNCTION, GENERATE, SEIZE, RELEASE, ADVANCE, QUEUE, TERMINATE, HOLD.

In GPSS 1100 there are about seventy-four different statements viz.

nine data definition statements.

fifty-four model statements.

eleven control statements.

However in the models to be described in subsequent units at most 40 different instructions will be required, viz.

seven data definition statements

twenty-six model statements or blocks

seven control statements

Following is a list of instructions which may be referred to. They will be described as their need arises. All are defined in the GPSS mini-manual provided.

Data Definition Statements (7)

CAPACITY, FUNCTION, INITIAL, MATRIX, QTABLE, TABLE, VARIABLE

Model statements (26)

ADVANCE, ASSIGN, COMPARE, ENTER, GATE, GENERATE, HOLD, INQUEUE, INTERRUPT, LEAVE, MARK, MSAVEX, OUTQUEUE, PRINT, PRIORITY, QUEUE, RELEASE, SAVEX, SEIZE, STOP, STORE, TABULATE, TERMINATE, TRANSREAD*, TRANSWRITE*, UNGUARD*

Control Statements (7)

CLEAR, END, JOB, ORDER, RESET, SEED, START.

* These statement are not required in units 1-9 but may be required in further, optional developments of the model referred to in Appendix F.

8.2.4 Scheduling by GPSS

GPSS has a built-in simulator clock which measures time in units. It is up to the programmer to decide if these units are to be regarded as seconds, minutes, hours, days or whatever. Initially the clock is set to the time at which the first TRANSACTION is due to enter the model. That TRANSACTION is moved from block to block until it cannot move any further at that time either because entry to the next block is for some reason prohibited (perhaps temporarily) or because the activity in the present block requires a certain processing time. The simulator then checks whether any other activity is scheduled at this time (there will not be, if this is the first TRANSACTION unless another TRANSACTION is scheduled to enter the model at the same time). The clock is then updated to the next time at which activity is scheduled. This might be the time at which the next TRANSACTION is due to enter the model or the time at which the blocked TRANSACTION is due to move. When the simulator has been in operation for some time there will be many TRANSACTIONS at different points in the model waiting to move. In this model this corresponds to vehicles being at various points in a building waiting for conditions to change so that they can move. All TRANSACTIONS which exist in a model i.e. those which have been created and have not yet reached a TERMINATE block are held in one of four so-called TRANSACTION chains.

These are as follows:

Current Events Chain

Future Events Chain

Interrupt Chain

User Chain

Current Events Chain

All TRANSACTIONS scheduled to move at the current or an earlier clock time are held in this chain in order of priority.

The TRANSACTION representing a lorry waiting on a queue would be held on the current events chain since it is free to move now if conditions allow it.

Future Events Chain

All TRANSACTIONS scheduled to move at a specific clock time greater than the present clock time are held in the future events chain. They are held in time order.

The TRANSACTION representing a lorry being unloaded would be held on the future events chain since it is scheduled to move at the future time when unloading finishes.

Interrupt Chain

TRANSACTIONS which cannot be scheduled until certain model conditions are met are held in this chain. When the conditions are met the TRANSACTION will be put in the current or future events chain.

Suppose a lorry is being unloaded and the person unloading it has to leave it to go and fetch some equipment for which he may have to queue then the future time at which unloading finishes cannot be determined until he returns. In the meantime the TRANSACTION is held on the interrupt chain.

User Chain

This is an alternative future events chain which is at the disposal of the programmer who wants to directly control event scheduling. It is not used anywhere in the models described in this text.

As far as programming is concerned the analyst need not concern himself with chains. However to understand how the scheduling works and to appreciate all of the output it is necessary to understand the current events chain and the future events chain.

Order of Events in the Current Events Chain

Those with highest priority number are first. Within a priority class, those with earliest scheduled time go first. If for two TRANSACTIONS both priority and scheduled time are the same then they will be in the order in which they were encountered in the programme. The following list is correctly ordered.

TRANSACTION	Priority number of TRANSACTION	Scheduled Time
A	5	1000
B	5	1200
C	5	2200
D	2	1800
E	1	1000
F	1	1200

Assume that the current clock time is 2200.

Note that the higher the priority number the higher the priority.

TRANSACTIONS A,B,C have priority number 5 and so are scheduled ahead of D,E,F which have lower priority (2,1,1 respectively). A,B,C have the same priority however A was scheduled to move first i.e. at t=1000 and has been waiting longer than B (scheduled to move at t=1200) or C (scheduled to move at t=2200) it is therefore ahead of B and C on the list.

Timing of Events

Whenever the clock is updated the simulator commences a scan of the current events chain. It selects the first TRANSACTION on the chain and moves that if possible from block to block until further movement is impossible. It then selects the next TRANSACTION (if any) on the current events chain and moves that as far as possible. If when moving the second or subsequent TRANSACTION conditions change which allow the moviement of a previously blocked TRANSACTION then the simulator immediately returns to the earlier blocked one and moves that. Thus a TRANSACTION can stop moving for one of three reasons:

1. It is in a block which requires processing time. In this case, it is taken from the current events chain and put on the future events chain.

2. It is seeking entry to a block but conditions are such that entry to that block is not possible. In this case, it is still held on the current events chain since conditions may change as subsequent TRANSACTIONS are moved (still at the current time) which allow it to move.

3. It has just caused a change which allows a TRANSACTION higher on the chain, which had previously been blocked, to move. The simulator then reverts to the earlier one.

When the simulator has scanned all events on the current events chain and finds that none can be moved it switches its attention to the future events chain. All events on the future events chain at the lowest scheduled time (call this t) are transferred to the current events chain and put in the correct position according to priority and time. The clock is updated to t and the scan of the current events chain begins again. If the future events chain is empty the program will terminate in error because things have, as it were, ground to a halt, nothing can move. This is a frequently encountered error and generally occurs because of an error in the model logic.

Following are some examples of the type of errors in model logic which can cause simulation to stop in error.

Example 8.1 The GENERATE statement has been programmed to generate up to 100 vehicles. The START statement has been programmed to stop simulation at time t=2000.
Suppose that the 100th (final) vehicle leaves the system at t=1000 and that at that time the system becomes empty i.e. there are no TRANSACTIONS currently in the system and none scheduled. The simulation will terminate in error because both CURRENT and FUTURE EVENTS chains are empty, but simulation is not due to terminate until t = 2000.

Example 8.2 Suppose that there are two blocked TRANSACTIONS on the CURRENT EVENTS chain i.e. 2 vehicles waiting until conditions change which will allow them to move. If the FUTURE EVENTS chain is empty then obviously no change can occur so they are permanently blocked and the simulation terminates in error. This could happen for instance because vehicle A is waiting for B to move and vehicle B is waiting for A to move.

Example 8.3 Suppose that the GENERATE statement has been programmed to generate up to 200 vehicles, starting at time t=0. At some point in the system they pass through a gate. An instruction specifies that this gate should not open until the clock time is 500 i.e. C$1=500. The START statement has been programmed to stop the simulation when 100 vehicles have left i.e. when the TERMINATION count = 100. One might be surprised to find that the simulation stopped at say t=2000 with the message "no new event in the sytem". An examination of the output reveals that although 200 TRANSACTIONS have been generated the TERMINATION count is zero because the gate did not open. How could this be? Since simulation started at t=0 and finished at t=2000 then at some time in between the time must have been 500. Right? No, wrong! The simulator clock does not advance in units. It jumps from one event to the next scheduled event. If no event was scheduled for t=500 then the simulator clock would never read 500 i.e. the statement C$1=500 would never be true. One would have to generate a TRANSACTION at t=500 to open the gate.

FLOW-CHART OF THE SCHEDULING ALGORITHM

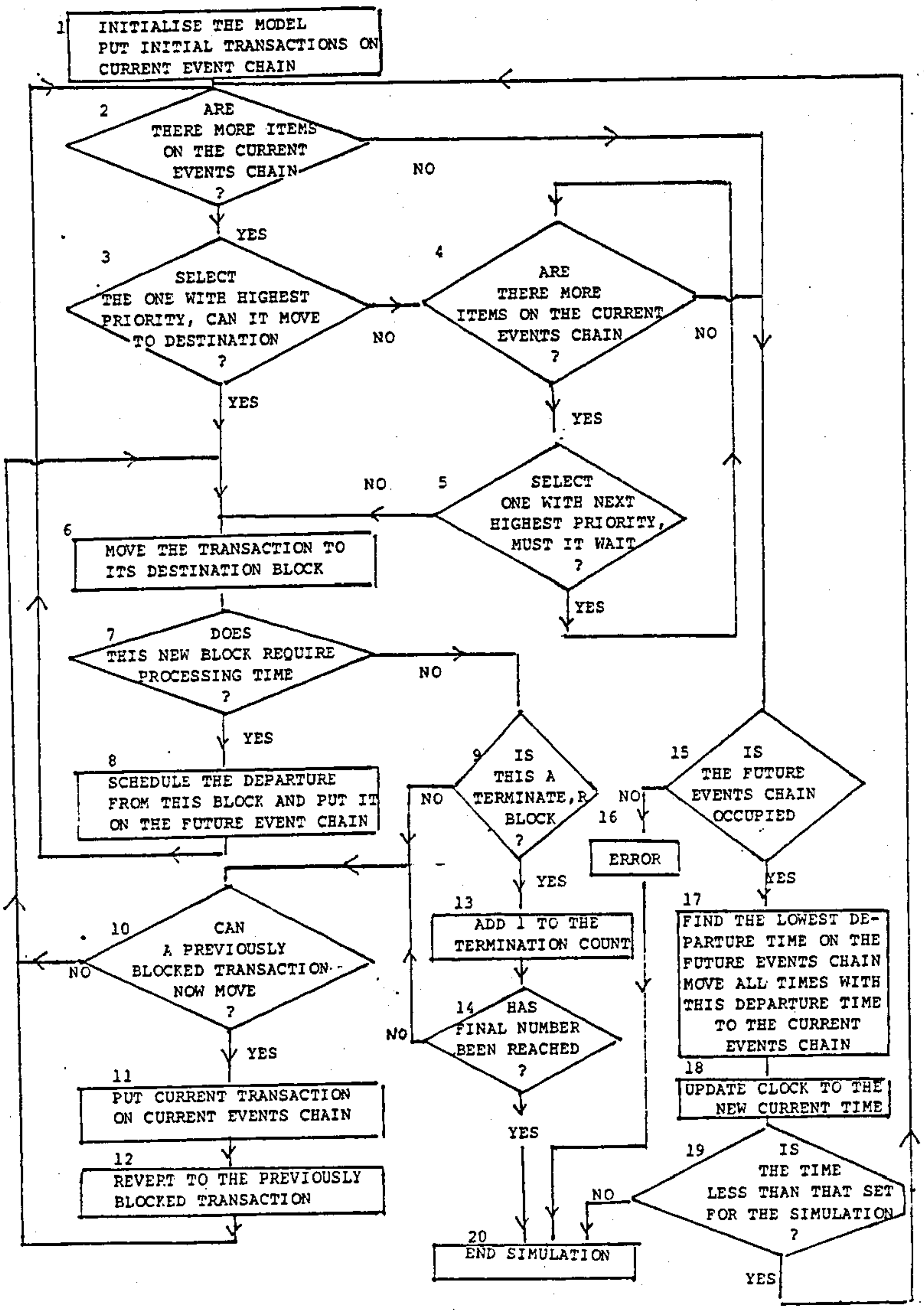

1 INITIALISE THE MODEL PUT INITIAL TRANSACTIONS ON CURRENT EVENT CHAIN
2 ARE THERE MORE ITEMS ON THE CURRENT EVENTS CHAIN ?
NO
YES
3 SELECT THE ONE WITH HIGHEST PRIORITY, CAN IT MOVE TO DESTINATION ?
NO
YES
4 ARE THERE MORE ITEMS ON THE CURRENT EVENTS CHAIN ?
NO
YES
5 SELECT ONE WITH NEXT HIGHEST PRIORITY, MUST IT WAIT ?
NO
YES
6 MOVE THE TRANSACTION TO ITS DESTINATION BLOCK
7 DOES THIS NEW BLOCK REQUIRE PROCESSING TIME ?
NO
YES
8 SCHEDULE THE DEPARTURE FROM THIS BLOCK AND PUT IT ON THE FUTURE EVENT CHAIN
9 IS THIS A TERMINATE,R BLOCK ?
NO
YES
10 CAN A PREVIOUSLY BLOCKED TRANSACTION NOW MOVE ?
NO
YES
11 PUT CURRENT TRANSACTION ON CURRENT EVENTS CHAIN
12 REVERT TO THE PREVIOUSLY BLOCKED TRANSACTION
13 ADD 1 TO THE TERMINATION COUNT
14 HAS FINAL NUMBER BEEN REACHED ?
NO
YES
15 IS THE FUTURE EVENTS CHAIN OCCUPIED
NO
YES
16 ERROR
17 FIND THE LOWEST DEPARTURE TIME ON THE FUTURE EVENTS CHAIN MOVE ALL TIMES WITH THIS DEPARTURE TIME TO THE CURRENT EVENTS CHAIN
18 UPDATE CLOCK TO THE NEW CURRENT TIME
19 IS THE TIME LESS THAN THAT SET FOR THE SIMULATION ?
NO
YES
20 END SIMULATION

Figure 8.1

To illustrate the operation of the scheduling algorithm the operation of the program on page 145 will now be described. The numbers in brackets at the end of the lines refer to items on the flow-chart. The steps below (1, 2 . . . 24) should be followed on the flow-chart. (see Figure 8.1)

FLOW-CHART BLOCK NUMBER

1. Check syntax (1)

2. Initialise FUNCTIONS with data on DATA DEFINITION statements (1)

3. Determine the time at which the first TRANSACTION is due to move

   This is TRANSACTION 1 from Block 1 to Block 2 at t=1 (1)

4. Initialise clock to time obtained in 3 i.e. t = 1

   This is now the current time.

   Put all TRANSACTIONS due to move at this time on the current Events Chain at t = 1 (1)

CURRENT EVENTS CHAIN

TRANSACTION NUMBER	CURRENT BLOCK	DESTINATION BLOCK	PRIORITY
1	1	2	0

5. The simulation can now commence (2)

6. What is the first move on the current events chain?

   This is TRANSACTION 1 from Block 1 to Block 2 (2)

7. Can this move be made?

   YES (3)

8. Move TRANSACTION 1 from Block 1 to Block 2 (6)

9. Does Block 2 require processing time? (7)

   NO

10. Is Block 2 a TERMINATE,R block?

    NO (8)

11. As a consequence of moving TRANSACTION 1 from Block 1 to Block 2 can any previously blocked TRANSACTION now move?

    YES, a new TRANSACTION can now be generated in Block 1.(10)

12. Restore TRANSACTION 1 to the Current Events Chain with Current Block equal to 2 and Destination Block equal to 3. (11)

13. Generate a new TRANSACTION (No. 2) in Block 1. (12, 6)

14. Does Block 1 require processing time?

    YES (7)

15. Generate an inter-arrival time. Suppose it is 1600 (8) schedule TRANSACTION,2 to leave Block 1 at t = 1601 Put this information on the future events chain.

FUTURE EVENTS CHAIN at t = 1

TRANSACTION NUMBER	AT BLOCK	DESTINATION BLOCK	SCHEDULED TIME	PRIORITY
2	1	2	1601	0

16. Are there more items on the current events chain?

    YES (2)

17. Select TRANSACTION,1 which is due to move from Block 2 to Block 3 at the current time. Can it move now?

    YES, as CHECKIN is free. (3)

18. Move TRANSACTION 1 from Block 2 to Block 3. (6)

19. Does Block 3 require processing time?

    YES (7)

20. Generate a random service-time. Suppose it is 48 TRANSACTION 1 is scheduled to try to move from Block 3 to its next destination, Block 4 at t = 49. This information is put on the future events chain which now is as follows (8)

FUTURE EVENTS CHAIN at t = 1 (updated)

TRANSACTION NUMBER	AT BLOCK	DESTINATION BLOCK	SCHEDULED TIME	PRIORITY
1	3	4	49	0
2	1	2	1601	0

21. Are there more items on the current events chain? (2)

    NO

22. Are there items on the future events chain? (15)

    YES

23. What is the lowest scheduled time on the future events chain? (17)

    It is 49.

    Select all TRANSACTIONS due to move at t = 49 (there is only one) and transfer to a new current events chain.

NEW CURRENT EVENTS CHAIN

TRANSACTION NUMBER	AT BLOCK	DESTINATION BLOCK	PRIORITY
1	3	4	0

24. Update the clock to a new current time t = 49 (18)

25. Is simulation due to finish before t = 49? (19)

    NO (Assume it is not)

26. Are there more items on the current events chain? (2)

    YES

    etc.

8.2.5 <u>Exercise on the Operation of the Simulation</u>

If time were available it would be worthwhile having a group of students simulate the operation of GPSS.
If one had a group of 5 students called A, B, C, D, E then

- A could be in charge of movements from block to block at the request of B.
- B would be in charge of maintaining the CURRENT EVENTS CHAIN
- C would be in charge of maintaining the FUTURE EVENTS CHAIN
- D would be in charge of FUNCTIONS
- E would be in charge of record keeping (QUEUE statistics, FACILITY utilisation)

A, B, C and E should each have a white-board or something similar for recording. D would need a table of random numbers and a set of graphs.

Simulation programs are often viewed as having a hierarchical structure (see M. Pidd, 1984) of three levels :

Level 1 : executive (control program);

Level 2 : operations

Level 3 : detailed routines.

Viewed this way B and C are acting as level 1, the executive (or control) level, A is acting as level 2 the operations level and D and E are acting as level 3 - the detailed routines used by level 2.

A's board would have a copy of the program blocks. He would move TRANSACTIONS (vehicles) from block to block on the board at B's instructions. At a particlar stage his board might appear as below

1.	GENERATE	1	TIME(FN$INTERARRIVAL.TIME)	
2.	QUEUE	CHECKIN.LINE		16 15 14
3.	HOLD	CHECKIN	TIME(FN$CHECKIN.TIME)	13
4.	ADVANCE		TIME(FN$DRIVE.TIME)	12 11
5.	QUEUE	UNLOAD.LINE		10 9 8 7
6.	HOLD	UNLOAD.BAY	TIME(FN$UNLOAD.TIME)	6
7.	TERMINATE			

Each symbol represents a TRANSACTION and the number on it is the TRANSACTION NUMBER. These numbers are assigned in sequence by the GENERATE block, i.e. as each vehicle enters it gets a number. Note that the symbols give a pictorial representation of the system at any time. Thus in the diagram above the situation is as follows: There is one vehicle being unloaded (number 6), there are four waiting to unload (7-10), there are two en route from the check-in to the bay (11-12), there is one being checked in (13), there are three waiting to check in (14-16) and there is one waiting to enter the system at its appropriate arrival time (17).

B's board would have a table such as that below which he updates at each new event time at the request of C. The values shown reflect a possible picture at T = 4745 in agreement with the details on A's board.

CURRENT EVENTS CHAIN AT T = 4745

TRANSACTION NUMBER	POSITION ON CHAIN	CURRENT BLOCK	DESTINATION BLOCK	PRIORITY
7	1	5	6	0
8	2	5	6	0
9	3	5	6	0
10	4	5	6	0
14	5	2	3	0
15	6	2	3	0
16	7	2	3	0

C's board would have a table such as that below. The scheduled times are obtained from D. For example when A moves a TRANSACTION into block 3 (i.e. a vehicle commences check-in), A will ask D to supply C with a duration for a check-in and hence a scheduled completion time (enters the details in his table (as for TRANSACTION number 13).

FUTURE EVENTS CHAIN AT T = 4745

TRANSACTION NUMBER	POSITION ON CHAIN	CURRENT BLOCK	DESTINATION BLOCK	SCHEDULED TIME	PRIORITY
6	1	6	7	4750	0
11	2	4	5	4750	0
12	4	4	5	4810	0
13	3	3	4	4800	0
17	5	1	2	5700	0

Note that the scheduled time at which 13 will finish checking in is 4800 which is earlier than the time at which 12 will reach the bay (4810). When writing these on a board it is simpler just to change the number in the "POSITION" column rather than interchange rows.

Example of Updating Procedure at T = 4745 and T = 4750

Suppose that at T = 4745 number 16 has just arrived and joined the check-in queue, details being entered on the CURRENT EVENTS CHAIN. None of the TRANSACTIONS on the CURRENT EVENTS CHAIN can move so time is updated to the earliest time on the FUTURE EVENTS CHAIN viz. 4750 and TRANSACTIONS 6 and 11 are moved to the CURRENT EVENTS CHAIN which is updated by B who then instructs A on what changes to make in the diagram. In fact 6 moves to block 7 and is removed, 7 moves to block 6 and D is asked to provide a load-time for 6, the details being entered by C on the FUTURE EVENTS CHAIN. 11 is moved to block 5 remaining on the CURRENT EVENTS CHAIN. E meanwhile records how long 7 has spent on the Unload Queue, how long it took 6 to be unloaded, and how long it took 11 to drive from the check-in to the bay. It is suggested that E maintains a table such as the following one from which he can compile statistics when simulation finishes. (This is shown overleaf.)

RECORD OF TRANSACTION DETAILS

TRANSACTION NUMBER	ARRIVAL TIME	START CHECK-IN	FINISH CHECK-IN	FINISH DRIVE	START TO UNLOAD	FINISH UNLOAD
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
5	1650	1650	1710	1770	3160	3900
6	3650	3650	3720	3775	3900	4750*
7	3700	3720	3775	3825		
8	3800	3800	3840	3900		
9	3840	3840	3919	3980		
10	3900	3919	4001	4051		
11	4610	4610	4710	4750*		
12	4620	4710	4740	4810*		
13	4650	4740	4800*			
14	4660					
15	4685					
16	4745					
17	5700*					
.						
.						
.						

* At T = 4745 these are still in the future but they have all been scheduled and are on the FUTURE EVENTS CHAIN.

8.2.6 Standard Output

It is possible by means of a PRINT block to print information requested by the user while the simulation is progressing. However, whether or not a PRINT block is used a standard report is produced which gives information on the queues in the program. On pages 168 and 169 there is a copy of the output produced by the program to which reference should be made while going through the points below. The details provided (in this instance) can be discussed under five headings (1) termination details (2) block details (3) FACILITY details (4) QUEUE details (5) random number generators.

8.2.6.1 Termination Details

Three pieces of information are provided. These are the values at the time of the printout of (a) relative clock time (b) absolute clock time (c) termination count. In this instance (a) and (b) are the same. However (as will be described later) it is possible to run a simulation in phases. The relative clock starts from zero in each phase but the absolute clock continues. The "termination count" is the number of TRANSACTIONS which had entered TERMINATE, R blocks i.e. the number of recorded terminations. In this case it is 2000 as the program requested the simulator to stop after 2000 terminations. This was requested on the START card. The clock time units are whatever the programmer intended. All time details were provided in seconds. Thus the first printout time of 895162 equals 248 hours, 39 minutes, 22 seconds which is the time simulated at that stage. Two printouts are provided since the START card was START 2000,,1000 i.e. simulate 2000 TRANSACTIONS with printouts after each 1000.

8.2.6.2 Block Details

For each of the seven blocks in the program two details are provided, the current TRANSACTIONS and the total TRANSACTIONS. The "total TRANSACTIONS" means the total number of TRANSACTIONS which have gone through the block during the simulation. The current TRANSACTIONS means the number of TRANSACTIONS which were in the block when the printout was requested i.e. when the 1000th TRANSACTION terminated.

Thus in this case when the final printout was produced the state of the system was as follows: (see page 169)

Current contents of all blocks except block 5 were zero

Current contents of block 5 are five.

This implies that when the 2000th vehicle left the system (causing the simulation to terminate and statistics to be printed) there were 5 vehicles waiting to be unloaded (block 5 is the "queue for unloading").

8.2.6.3 FACILITY Details

For each FACILITY in the model three statistics are provided, the average utilisation, the total number of entries and the average time per TRANSACTION. In this model there are two FACILITIES, the check-in and the unload-bay. It can be seen from the figures (page 168) that 1000 vehicles used the check-in, the average check-in time was 59.87 seconds and the utilitisation of the check-in was .0669 i.e. the FACILITY was in use for .0669 of the total simulated time. This can also be got by dividing the total time utilised (1000x59.87) by the elapsed time 895162.

$$\frac{1000 \times 59.87}{895162} = 0.06688 \approx .0669$$

8.2.6.4 Queue Details

For each queue eight statistics are provided (see page 168). Their meaning in most cases is obvious. "Zero Entries" means the number of TRANSACTIONS which spent zero time in the queue i.e. did not have to wait. "Average time/Ent (Non Zero)" means the average waiting time in the queue for all vehicles which did have to wait i.e. excluding the ones which did not wait. The example below should make this clear.

Waiting times 0, 5, 3, 0, 0, 2, 0, 6, 4, 0 total = 20

Average waiting time (All) $= \frac{20}{10} = 2$

Average waiting time (Non-zero) $= \frac{20}{5} = 4$ (only 5 waited)

The final column headed "Table Name" is not used here. It is possible to assign a table name to a QUEUE. In this case a table is produced giving extra details of the QUEUE entries including the standard deviation. This will be discussed later. It is not part of the standard output.

8.2.6.5 Random Number Generators

This is simply a list of the multiplier, increment and seed for each of the ten generators.

```
  RELATIVE      ABSOLUTE  TERMINATION
 CLOCK TIME    CLOCK TIME    COUNT
     895162        895162    1000

BLOCK  CURR  TOTAL      BLOCK  CURR TOTAL      BLOCK  CURR TOTAL      BLOCK  CURR TOTAL      BLOCK  CURR TOTAL
  #    TRAN  TRAN         #    TRAN  TRAN        #    TRAN  TRAN        #    TRAN  TRAN        #    TRAN  TRAN
   1     0   1000         2         1000          3     0  1000          4     0  1000          5     0  1000
   6     0   1000         7         1000

  FACILITY      AVERAGE   NUMBER      AVERAGE
    NAME     UTILIZATION ENTRIES    TIME/TRANS
CHECKIN         .0669      1000         59.87
UNLOAD.BAY      .8542      1000        764.62

 QUEUE       MAXIMUM   AVERAGE    TOTAL   ZERO    ZEROS  AV. TIME/ENT AV. TIME/ENT  CURRENT   TABLE
 NAME       CONTENTS  CONTENTS  ENTRIES ENTRIES PERCENT     (ALL)      (NON ZERO)  CONVENTS   NAME
CHECKIN.LINE    2         .00     1000    967     96.7       1.02         30.82        0
UNLOAD.LINE    15        1.78     1000    265     26.5    1593.33       2167.80        0

 RANDOM      RANDOM       RANDOM      RANDOM
GENERATOR  MULTIPLIER   INCREMENT      SEED
    1      1220703125           0 26615203829
    2      3141592653  2718281829 32828076357
    3      2718281829  3141592653  5570918525
    4     10604499373  7261067085 17801426661
    5     17249876309  7261067085 17249876309
    6     30517578125  7261067085 30517578125
    7      2565727293    35981228  2565727293
    8       107936437     4292354   107936437
    9     22438762221        6891 22438762221
   10       621444377    92111326   621444377
 RELATIVE     ABSOLUTE  TERMINATION
CLOCK TIME   CLOCK TIME    COUNT
  1777244      1777244     2000
BLOCK  CURR TOTAL        BLOCK  CURR TOTAL        BLOCK  CURR TOTAL        BLOCK  CURR TOTAL        BLOCK  CURR TOTAL
  #    TRAN  TRAN          #    TRAN  TRAN          #    TRAN  TRAN          #    TRAN  TRAN          #    TRAN  TRAN
  1      0   2005          2      0   2005          3      0   2005          4      0   2005          5      5   2005
  6      0   2000          7      0   2000

 FACILITY     AVERAGE    NUMBER      AVERAGE
   NAME     UTILIZATION ENTRIES    TIME/TRANS
CHECKIN        .0679      2005         60.15
UNLOAD.BAY     .8594      2000        763.65

 QUEUE       MAXIMUM   AVERAGE     TOTAL    ZERO      ZERO   AV. TIME/ENT AV. TIME/ENT  CURRENT      TABLE
 NAME        CONTENTS  CONTENTS   ENTRIES  ENTRIES  PERCENT     (ALL)      (NON ZERO)   CONVENTS     NAME
CHECKIN.LINE     2         .00     2005     1923     95.91        1.46        35.72         0
UNLOAD.LINE     15        1.49     2005      525     26.18     1317.63      1785.03         0

 RANDOM      RANDOM      RANDOM      RANDOM
GENERATOR  MULTIPLIER   INCREMENT     SEED
    1      1220703125           0 15594875049
    2      3141592653  2718281829 15543230746
    3      2718281829  3141592653 28377170658
    4     10604499373  7261067085 28330082077
    5     17249876309  7261067085 17249876309
    6     30517578125  7261067085 30517578125
    7      2565727293    35981228  2565727293
    8       107936437     4292354   107936437
    9     22438762221        6891 22438762221
   10       621444377    92111326   621444377
```

8.2.7 Analysis of Output

Usually one of the first things one would look at is the column "MAXIMUM CONTENTS" for the queues to see if any were too long. It can be seen that for the queue UNLOAD.LINE, the MAXIMUM CONTENTS = 15 which seems excessive. This relates to the fact that for Unload bay the AVERAGE UTILISATION = .8594 which is quite high. The average waiting time for the unloading bay was 1317.63 seconds or 22 minutes approximately. If this is undesirable then a second bay for unloading is required.

Note that for the purpose of calculating QUEUE statistics items still in the system when simulation stops will be treated as if they left the QUEUE at that time. This will underestimate queueing time if there are many such items, since they would presumably have continued queueing had simulation continued.

8.2.8 Interpretation of Output

What was simulated was approximately 250 hours of continuous operation. Is this realistic? Would a real system operate non-stop for 250 hours with the same arrival-rate?

One can distinguish between two types of simulation, terminating simulations and steady-state simulations.

An example of a terminating simulation would be the simulation of a godown which opened each morning at 9 a.m. and closed at 6 p.m. where one wished to simulate a complete days operation. Of course any number of days could be simulated but each is separate. Closing could also depend on the vehicles inside i.e. close at 6 p.m. or when the last vehicle leaves, whichever is later (no vehicles being admitted after 5.30 p.m. say). A battlefield simulation could terminate when the last soldier on one side is killed. The point is that the occurrence of some event terminates these simulations.

A steady-state simulation continues indefinitely i.e. there is no natural end-point. The programmer decides on an arbitrary end point e.g. after 2000 vehicles have been dealt with, after 1000 hours etc. The simulation described in this unit is a steady-state simulation. It is possible that one might use a steady-state simulation even though the real system is of the terminating type. Suppose that in a bank the peak period is noon till 3 p.m. when the average arrival rate is 100 per hour. One might want to simulate the peak only i.e. one is not concerned with the off-peak periods. In that case one could perform a steady-state simulation of the peak i.e. a continuous peak with arrival-rate 100 per hour. In some cases particularly if the peak is of very short duration this might not be realistic. Suppose that in a real system there was a sharp peak for 30 minutes only. During 30 minutes there is a maximum size any queue can reach. Provided the arrival rate drops off sharply after the peak the queue can be dealt with then. In a steady state simulation peak conditions would persist and the queue might simply grow indefinitely.

A problem with steady state simulations is the starting condition. The simulation described in this unit starts with an empty system as do most simulations. If however one were simulating say the peak bank operation just described it would be unrealistic to start with an empty system i.e. the bank is never empty at noon. On way of dealing with this problem is to start with an empty system but not to start collecting statistics until normal (peak) operation conditions have been reached e.g. if one wanted to collect data on 20 hours of peak operation one could simulate 40 hours but only collect data for the second 20 hours. The length of this initial warm-up period is an important consideration. One could do a preliminary simulation with intermediate results printed after 10, 20, 30, 40 hours etc. until one believes that steady state conditions prevail. One can then repeat only collecting data after this initial period. How this can be done using a RESET Block is described in Unit 8.

Before one carries out any simulation one should have a clear idea of what information one requires. The standard output discussed on pages 165-167 might not suffice. Generally speaking the average queueing time is not of much use especially in a terminating simulation where it has been averaged over peaks and troughs. What a user wants to know is "will the system work?" What does this mean in the case of the godown? Basically it is the same as asking "will many customers complain about delays in the system?" In other words one wants to know how many (or what percentage of) vehicles will be unduly delayed. To answer this one needs to know the distribution of queueing time i.e. what percentage of queueing times exceed 30 minutes? 40 minutes? 60 minutes? This is not answered by the standard output. It requires a special table to be produced. How this is obtained is described in Unit 8.

The program discussed in this chapter simulated 2000 vehicles but why 2000? Why not 1000 or 10,000? As in all sampling experiments large samples give more accurate estimates than small samples. However long runs cost more than short runs. Statistics can be of use here (see Law, 1983) however an intuitive approach such as the following might suffice.

Suppose one wishes to estimate the average queueing time to within $\pm$ 5 minutes. Carry out say four simulations of 500 vehicles (using different random numbers each time). Suppose the four results are as follows. Average queueing time - 43 minutes, 44 minutes, 40 minutes, 41 minutes. It is fairly safe to assume that if one takes the average of these four independent estimates viz. 42 minutes it will not be in error by more than 5 minutes.
If however the answers had been 35 minutes, 50 minutes, 40 minutes, 45 minutes then one would have litte confidence of the mean being correct to within 5 minutes. One would need to try again with about 4 simulations of about 3000 vehicles in each. This could continue until one was satisfied that the four results were reasonably consistent.

CHAPTER 9

UNIT SEVEN OF TEACHING PACKAGE

9.1 INTRODUCTION TO UNIT 7

In this unit more GPSS concepts will be introduced, enabling students to model slightly more complicated systems. The system modelled in Unit 4 will be expanded to allow for

1. A period at the beginning of the day when vehicles can arrive but must wait until the check-in opens.
2. A lunch-break period during which check-in is closed.
3. A closing time for the check-in.
4. Two vehicle types with different arrival rates.
5. Multiple unloading bays.
6. Maximum building capacity.
7. Change in arrival rate during lunch period.
8. Printout of details during the simulation.

To model the above some new GPSS concepts and blocks will be required.

The concepts required relate to the following:

Parallel segments (several GENERATE blocks)

TRANSACTIONS as "triggers" rather than physical entities

STORAGES (multiple service facility)

PARAMETERS of TRANSACTIONS

PRINT facility

VARIABLE entity

In connection with the above the following additional data definition statements and blocks will be required:

Data definition Statements: CAPACITY, VARIABLE

Blocks: ASSIGN, COMPARE, ENTER, GATE, INTERRUPT, LEAVE, PRINT, STOP, STORE

Having completed the unit students should be asked to produce their own version of the model but for a modified system. These modifications could include the following:

1. Bays open at 8.45 i.e. fifteen minutes later than the check-in

2. No Fast Delivery Service (F.D.S.) vehicles will be checked in after 16.30 if all F.D.S. bays are full. No Express Transport (E.T.) vehicles will be checked in after 16.45 if all E.T. bays are full.

3. Although the maximum capacity of the system is 9 the number of E.T. vehicles waiting inside for a bay cannot exceed 3 and the number of F.D.S. vehicles waiting inside for a bay cannot exceed 2.

4. During the lunch period when no vehicles were allowed in did the system ever become empty i.e. no vehicles in the building and if so for how long?

5. At what time did the check-in actually close for lunch?

6. The drive time from check-in to bay should be variable not constant.

Students should perform the above or similar modifications using only the blocks encountered so far.

9.2 UNIT SEVEN - FURTHER GPSS CONCEPTS

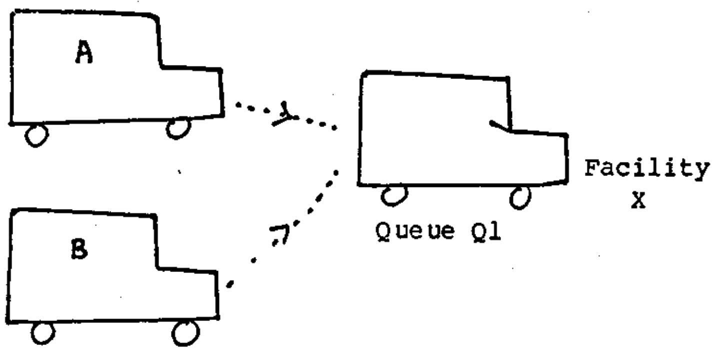
9.2.1 Program Segments

In the model used in the previous units there was only one programme segment commencing with a GENERATE statement and ending with a TERMINATE. However one can have any number of GENERATE blocks.

Consider the system illustrated below which shows arrivals from two different sources

Vehicles from Client A
Arriving at rate 1

Vehicles from Client B
Arriving at rate 2



The two sources can be represented by two GENERATE statements. One could write separate program segments for each client

```
*SEGMENT REPRESENTING CLIENT A

     GENERATE 1 TIME(FN$ARRIVAL.1)

     QUEUE Q1

     HOLD FACILITY.X TIME(X)

     TERMINATE,R

*SEGMENT REPRESENTING CLIENT B

     GENERATE 1 TIME(FN$ARRIVAL.2)

     QUEUE Q1

     HOLD FACILITY.X TIME(Y)

     TERMINATE,R
```

An asterisk in column one denotes a comment statement i.e. it is not part of the model.

The two segments are not obeyed in sequence but in fact are superimposed since they both start generating TRANSACTIONS at the same time although at different rates and both sets of TRANSACTIONS occupy the same QUEUE and FACILITY. A program may consist of many such segments which may be operating simultaneously or in sequence, depending on the time spans involved.

In fact the two segments on page 176 could be written as a single segment using the GOTO field as below.

```
          GENERATE 1 TIME(FN$ARRIVAL.1)   GOTO(L1)
          GENERATE 1 TIME(FN$ARRIVAL.2)
     L1   QUEUE Q1
          HOLD FACILITY.X TIME(X)
          TERMINATE,R
```

<u>Exercise 9.1</u>

At this point students should be given a table of random numbers (or use that in the GPSS mini-manual) and asked to manually go through several cycles of the two segments on the previous page using Tables 9.1, 9.2 and 9.4 (page 187) for the relevant distributions.

9.2.2 <u>Interventions (Opening, Closing, Lunch Breaks etc.)</u>

TRANSACTIONS as used previously represented vehicles but a TRANSACTION does not always represent a physical entity. It may be used to trigger some activity. Suppose for instance that one wants to close the check-in for one hour at lunch time. A single TRANSACTION can be generated for this purpose in a special segment. One must consider however how one wants to deal with an existing queue when the check-in is to be closed. There are at least three possibilities.

(1) At 12.30 a notice is placed at the end of the queue to inform arriving vehicles that no more vehicles will be dealt with until the FACILITY re-opens but the existing queue will be cleared.

(2) If a check-in is in progress at 12.30 that will be completed before the FACILITY closes but any queueing vehicles must wait.

(3) At 12.30 p.m. the check-in closes even if in the midst of a check-in. The interrupted vehicle must wait and complete its check-in when the FACILITY re-opens at 1.30.

(1) is the easiest to deal with and will be illustrated first

```
GENERATE 240,1
HOLD CHECK-IN TIME (60)
TERMINATE
```

The above segment generates a single TRANSACTION at time 12.30 (assuming t = 0 is 08.30 and time is measured in minutes). It tries to occupy the check-in but must wait till all TRANSACTIONS ahead of it are dealt with. The fact that there is no QUEUE block preceding the HOLD block does not mean that it does not wait. It merely means that it is not recorded in the queue data and one does not want it to be as it is not a vehicle. Once it occupies CHECK-IN it holds it for 60 minutes and then releases it. The next block is TERMINATE and not TERMINATE,R as this TRANSACTION should not be counted as if it were a vehicle.

(2) is accomplished by using a higher priority TRANSACTION

```
GENERATE 240,1,1
HOLD CHECK-IN TIME(60)
TERMINATE
```

This is similar to the segment in (1) except that the TRANSACTION is generated with PRIORITY 1. Assuming that all other TRANSACTIONS (vehicles) have been generated with PRIORITY zero this TRANSACTION immediately goes to the head of the queue and need only wait for the current check-in (if any) to finish.

(3) is accomplished by using a new block called INTERRUPT

```
GENERATE 240,1
INTERRUPT CHECK-IN TIME(60)
TERMINATE
```

The INTERRUPT block is similar to HOLD but has a higher priority and actually interrupts an ongoing service. Thus if there is a TRANSACTION in CHECK-IN when a TRANSACTION enters the INTERRUPT, the service being provided to the former TRANSACTION is halted (the TRANSACTION is removed to the interrupt chain to await rescheduling) while the interrupting TRANSACTION waits its required time (60 minutes in above example). The interrupted TRANSACTION is then resumed and spends the remaining portion of its processing time. If the TRANSACTION being interrupted is itself an interrupting TRANSACTION then the second interrupt must wait unless it has a higher priority. This complication will not arise in the models to be discussed.

9.2.3 Multi-Server Facilities

The FACILITY and QUEUE described in the model discussed in Unit 6 refer to what is known in queueing theory as a single server queue i.e. only one TRANSACTION at a time can occupy the FACILITY. Frequently one might wish to use a multi-server system. For instance in the system previously described there might be several unloading bays to which a vehicle could proceed. The vehicle would only queue if all bays were occupied. In GPSS this is represented by a STORAGE. A STORAGE is similar to a FACILITY but whereas only one TRANSACTION can occupy a FACILITY, several TRANSACTIONS can simultaneously occupy a STORAGE, the maximum number being specified on a data definition statement called appropriately "CAPACITY". The equivalent of the FACILITY blocks SEIZE and RELEASE are called ENTER and LEAVE. The equivalent of the HOLD block is a block called STORE. Consider the following sequence.

```
UNLOAD.BAY CAPACITY 6
GENERATE 1 TIME(X)
QUEUE Q1
STORE UNLOAD.BAY TIME(Y)
TERMINATE,R
```

The first statement defines UNLOAD.BAY as a storage with capacity 6. As TRANSACTIONS are generated they proceed onto Q1 and then to UNLOAD.BAY where they remain for a time Y (Y is calculated anew for each TRANSACTION). Up to six TRANSACTIONS can occupy the STORE block simultaneously. If a seventh one seeks entry it is prevented and remains in Q1.

The pair ENTER and LEAVE may be used in place of STORE if one wishes the TRANSACTION to engage in activity after entering and before leaving the STORAGE. It is possible for a TRANSACTION to enter more than one unit into a STORAGE. The full formats are as follows

```
STORE NAME,n
ENTER NAME,n
LEAVE NAME,n
```

n is the number of units entered into the STORAGE. If not specified it is understood to be one. For other applications the STORAGE might represent an actual store of inventory and one might want to enter n units.

9.2.4 <u>Attributes of Entities Represented by TRANSACTIONS (Parameters)</u>

TRANSACTIONS as previously described cannot be distinguished one from another. Frequently however one might want the TRANSACTION to carry details of some attributes of the entity which it represents. For instance if it represented a vehicle one might want the record to include the vehicle type, size of load, name of owner etc. as these things might affect how the TRANSACTION (vehicle) is to be dealt with as it progresses through the model. This is done by assigning a set of PARAMETER values to each TRANSACTION. In addition to these user defined PARAMETERS each TRANSACTION has a unique identification number assigned to it by the simulator and this number is accessible to the programmer by refering to TN$1. PARAMETER values are assigned by means of an ASSIGN block. The format for this is as follows:

```
ASSIGN X,Y
```

X is the name of the PARAMETER; Y is a value or a name to be stored in the PARAMETER called X.

Examples of PARAMETER Assignments

```
        ASSIGN FLOOR,FN$FLOOR.ASSIGN

        ASSIGN ROUTE,L25

        ASSIGN SERVICE,CHECKIN.4
```

A TRANSACTION which goes through all of the above three ASSIGN statements has three PARAMETERS called respectively, FLOOR, ROUTE and SERVICE.

The PARAMETER called "FLOOR" is assigned a numerical value which is the current value of the FUNCTION called FLOOR.ASSIGN

The PARAMETER called "ROUTE" is assigned the name L25. This could be a block label so that if in any block there is an instruction,

```
    GO TO(*ROUTE)
```

The asterisk indicates that what follows is an indirect specification of a label i.e. that ROUTE is not the label but the name of a PARAMETER which contains the label.
The simulator will substitute the contents of the PARAMETER called ROUTE which will depend on the TRANSACTION. For the TRANSACTION here refered to it would be GO TO(L25) i.e. go to the block labelled L25.

The PARAMETER called SERVICE contains the name CHECKIN.4. This might be the name of a FACILITY. If a subsequent block said HOLD *SERVICE the entering TRANSACTION would attempt to occupy a FACILITY. Which FACILITY would depend on the contents of the PARAMETER called SERVICE of that TRANSACTION. For the TRANSACTION in this example the statement would be interpreted as HOLD CHECKIN.4. Thus different TRANSACTIONS could be assigned to different FACILITIES according to the contents of their PARAMETER called SERVICE.

9.2.5 Printing Information During a Simulation

The PRINT block allows one to print numerical values only, during the progress of the simulation. Up to five numbers per line are printed. The format is as follows:

PRINT n_1, n_2, n_3 . . . .

when a TRANSACTION enters this block the values of n_1, n_2 . . . . are printed, five per line. n_1, n_2 ... are NUMERIC ATTRIBUTES (see GPSS mini-manual)

Exercises 9.2 and 9.3

To test assimilation of the material so far encountered in this unit students should do the following exercises.

9.2 Write a program segment which will print the following details at t=200, and thence at intervals of 100.

1. The value of t
2. The total number of TRANSACTIONS which have terminated

(see mini-manual for NUMERIC ATTRIBUTES C$1 and N$label)

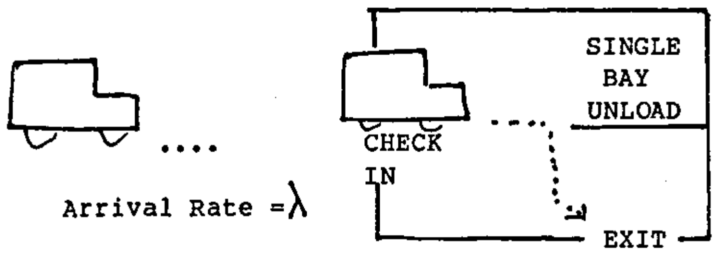
9.3 Write a program segment which will start generating TRANSACTIONS at t=0 and will assign PARAMETERS A,B,C,D with equal frequency i.e. 25% of TRANSACTIONS will be assigned A, 25% B etc. TRANSACTIONS will then be routed each to its appropriate FACILITY called respectively FAC.A, FAC.B, FAC.C and FAC.D where they are serviced.
Service at FAC.A takes 5 minutes, at FAC.B 10 minutes, at FAC.C 12 minutes and at FAC.D 15 minutes. After service they go to a TERMINATE block.

9.2.6 More Complex System

It is now possible to model a more complex system than that of Unit 4. Below are diagrams of the system modelled in Unit 4 and a more complex one.

SYSTEM MODELLED IN UNIT 4

Single Vehicle Type No Maximum Capacity Operating Times: Continuous

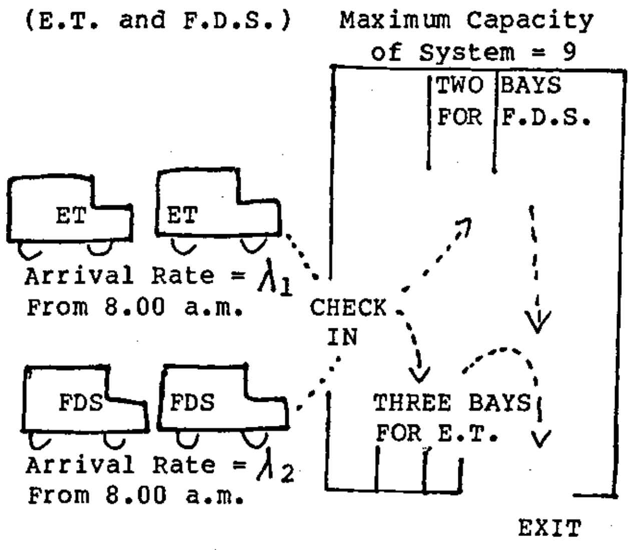


Arrival Rate: Constant

MORE COMPLEX SYSTEM (Details here are slightly simplified and will be explained more fully in the text)

Two Vehicle Types

(E.T. and F.D.S.)



Operating Times For Check-in:
8.30 - 12.30
13.30 - 17.30

Arrival Rates:
Constant except during lunch-hour when they are halved

Description of the More Complex System to be Simulated:

The check-in opens at 08.30 but vehicles will have been arriving since 08.00. The same arrival rate persists from 08.00 to 17.30 except between 12.30 and 1.30 when the check-in is closed. During this time the arrival rate is halved. At 12.30 any check-in in progress is completed before the facility closes for one hour. After 17.30 no more vehicles are accepted but any vehicles in the system (i.e. which have already started check-in) are cleared.

Inside the sytem there are five unloading bays, three belonging to a company called Express Transport (E.T.) and two to a company called Fast Delivery Service (F.D.S.). Vehicles belonging to E.T. must go to one of the E.T. bays and those belonging to F.D.S. to one of the F.D.S. bays. All have the same service time distribution. To drive from the check-in to the unloading bay takes a constant 1 minute. No more than nine vehicles can be admitted to the building at any one time. The bays do not close for lunch.

Simulate one days operation. In addition to the standard reports, provide a list of all vehicles which used the system specifying for each the company identity, arrival time and departure time. Then simulate a second day using different random numbers and provide the same information. The required distributions are given in Tables 9.1 - 9.4.

TABLE 9.1

Inter-arrival Time for E.T.	Relative Frequency
15	.1
20	.3
25	.3
30	.2
35	.1

TABLE 9.2

Inter-arrival Time for F.D.S.	Relative Frequency
15	.2
30	.3
45	.3
60	.2

TABLE 9.3

Check-in Service time (All)	Relative Frequency
3	.10
4	.15
5	.40
6	.25
7	.10

TABLE 9.4

Unload Time (All)	Relative Frequency
15	.10
30	.15
45	.20
60	.30
75	.15
90	.10

All times are in minutes, to the nearest minute.

At this stage students could be asked to write the program for the revised system before this is done by the lecturer. In general it is found that people with little experience of GPSS, such as students, at this stage tend to make the program more complex than is necessary so one might ask which group (assuming that the class has been divided into groups) can write the program using the least number of blocks.

If time is scarce and this must be done during a single class then each group could be given one complication from the nine listed on the following pages and then asked to write their segment on the board for criticism.

Complications which distinguish this situation from the previous one in Unit 4 will be considered one at a time and each one dealt with in the following nine steps numbered 1-9.

1. Vehicles begin to arrive at 08.00 but the check-in does not open until 08.30.

This can be dealt with by having a TRANSACTION arrive at 07.59 (i.e. before any vehicles) which occupies the check-in for thirty-one minutes. It shall be assumed that the basic unit of time is one minute and that the time of day is measured in minutes from 00.00. Thus 07.59 is 7 x 60 + 59 = 479. The required segment is:

```
*OPENING TIME SEGMENT
     GENERATE 479,1
     HOLD CHECK.IN TIME(31)
     TERMINATE
```

2. The check-in is closed for 60 minutes at 12.30. This can be dealt with by having a TRANSACTION of high priority arrive at 12.30 and occupy the check-in for 60 minutes. The required segment is:

```
*LUNCH BREAK SEGMENT
     GENERATE 750,1,1
     HOLD CHECK.IN TIME(60)
     TERMINATE
```

3. The check-in must be closed at 17.30 but simulation must be continued until all vehicles are cleared. This can be dealt with by ensuring that no TRANSACTIONS leave the generate block after 17.30 i.e. time = 1050. To do this it is necessary to use a COMPARE block. Entry to this block is dependent on the relationship between two NUMERICAL ATTRIBUTES (See GPSS mini-manual).

The format is as follows:

COMPARE n_1 X n_2

where X can be LT (less than), LE (less than or equal to) EQ (equal to) NT (not equal to) GE (greater than or equal to) or GT (greater than). Consider the following:

```
GENERATE 480 TIME(...)

COMPARE C$1 LT 1050
```

C$1 returns the current clock time, therefore a TRANSACTION cannot leave the GENERATE block to enter the COMPARE block if the time is greater than or equal to 1050.

4. There are two types of vehicle (E.T. and F.D.S.). They can be generated by two separate GENERATE statements with different arrival rate distributions.

5. There are 5 unloading bays 3 for E.T. and 2 for F.D.S. This is modelled by defining two STORAGES as follows:

```
E.T.BAY         CAPACITY 3

F.D.S.BAY       CAPACITY 2
```

6. No more than nine vehicles can be admitted to the building at any one time. It may be assumed that if nine vehicles are inside subsequent arrivals will be held at the check-in. This can be accomplished by defining the building as a STORAGE with CAPACITY 9 as follows

```
BUILDING CAPACITY 9
```

When a TRANSACTION finishes check-in it will try to ENTER BUILDING. If nine are already occuping BUILDING then it must remain at the checkin. The instructions would be as follows:

```
HOLD CHECK.IN TIME(FN$CHECKIN.TIME)
ENTER BUILDING
•
•
LEAVE BUILDING
TERMINATE,R
```

Note that the last instruction before TERMINATE,R is LEAVE BUILDING.

7. The arrival rate must be halved between 12.30 and 13.30. To do this it is necessary to check the simulator clock time. The clock time may be got by referring to C$1. It will be necessary first to understand the GPSS "VARIABLE" entity for which one should refer to the GPSS mini-manual.

Suppose that one defines an INTEGER VARIABLE called ARR.T as follows:

```
ARR.T  VARIABLE,I C$1/750-C$1/810+1
```

then if $C\$1 < 750$ V\$ARR.T = 1

if $750 \leqslant C\$1 < 810$ V\$ARR.T = 2

if $810 \leqslant C\$1 < 1500$ V\$ARR.T = 1

Then if the inter-arrival time is multiplied by the value of this VARIABLE it will be doubled if $750 \leqslant C\$1 < 810$ i.e. if the time is between 12.30 and 13.30.

The GENERATE statement will read

```
    GENERATE 480 TIME(FN$ARRIVAL *V$ARR.T)
```

8. A print-out showing for each vehicle the company identity, arrival time and departure time is required. This must be printed after completion of unloading of each vehicle as the departure time is required. The TRANSACTION must carry a record of its arrival time. This is done by assigning the clock-time to a PARAMETER called ARRIVAL.TIME for each TRANSACTION as it is generated.

Following are two program segments for E.T. and F.D.S. respectively containing all the model details discussed so far in 1-8 above.

If the symbol @ appears in a statement what follows is ignored i.e. it is a comment for ease of reading.

```
*EXPRESS TRANSPORT SEGMENT
 GENERATE 480 TIME(FN$E.T.ARRIVAL*V$ARR.T)
 COMPARE C$1 LT 1050
 ASSIGN ARRIVAL.TIME,C$1    @  (note C$1 = current clock time)
 QUEUE CHECKIN.LINE
 HOLD CHECK.IN TIME(FN$CHECKIN.TIME)
 ENTER BUILDING
 ADVANCE TIME (1)
 QUEUE E.T.LINE
 STORE E.T.BAY TIME(FN$UNLOAD.TIME)
 LEAVE BUILDING
 PRINT 1,P$ARRIVAL.TIME,C$1 @  (1 refers to E.T.)
 TERMINATE,R

*FAST DELIVERY SERVICE SEGMENT
 GENERATE 480   TIME(FN$FDS.ARRIVAL*V$ARR.T)
 COMPARE C$1 LT 1050
 ASSIGN ARRIVAL.TIME, C$1
 QUEUE CHECKIN.LINE
 HOLD CHECK.IN TIME(FN$CHECKIN.TIME)
 ENTER BUILDING
 ADVANCE TIME(1)
 QUEUE F.D.S.LINE
 STORE F.D.S.BAY  TIME(FN$UNLOAD.TIME)
 LEAVE BUILDING
 PRINT 2,P$ARRIVAL.TIME,C$1
 TERMINATE,R
```

9. The eight steps just dealt with describe the model. What remains is how to stop the simulation when the conditions specified on page 186 are met. It will be recalled from Unit 6 that one can distinguish two types of simulation, a steady state simulation (as in Unit 4) and a terminating simulation (as is the model under discussion).

A steady state simulation is usually stopped by means of data in the START block. It is necessary here to recall the format of the START block (see GPSS manual). If the format used is START n_1,n_2

then n_1 = the number of TRANSACTIONS to be recorded at TERMINATE,R blocks before stopping simulation

n_2 = the time at which simulation should be stopped.

Simulation will cease whenever the earlier of the two conditions is satisfied.

In the present case one wants the end of the simulation to depend on model conditions, specifically on when the last vehicle left and not by specifying n_1 and/or n_2 i.e. the simulation is of the terminating type and not steady state. To do this one uses a STOP block. This allows one to stop the simulation as a consequence of model conditions. Here one wants to stop simulation when the last vehicle has left. This will be the time when the following two conditions are first true.

(a) The time is 17.30 or later

(b) The building is empty.

Thus starting at 17.30 and every minute thereafter one wants to ask the question "is the building empty?" and if so end the simulation. "Is the building empty" corresponds in GPSS to asking "is the STORAGE called "BUILDING" empty?" This is achieved by using a GATE block which is somewhat similar to a COMPARE block. Entry to a COMPARE block depends on the relationship between two NUMERIC ATTRIBUTES whereas entry to a GATE block depends on conditions associated with a STORAGE or FACILITY (or some other possibilities not considered here). The format is as follows:

```
     GATE X, name
```

"name" is the name of a STORAGE or FACILITY. X denotes the state of the STORAGE or FACILITY which will allow entry to the GATE block, specifically whether or not the STORAGE or FACILITY is occupied.

X can have a number of designations such as

SE = <u>S</u>TORAGE <u>E</u>MPTY or

SNE = <u>S</u>TORAGE <u>N</u>OT <u>E</u>MPTY

A list is given in the GPSS mini-manual.

Consider the block

```
        GATE SE,BUILDING
```

This allows entry only if the STORAGE called BUILDING is empty. To recap what one wants to do is as follows with corresponding GPSS action

1. Starting at 17.30	Generate a TRANSACTION at 17.30 (i.e. t = 1050)
2. ask if the building is empty	Use GATE block
3. if so, stop the simulation	Use STOP block

The blocks in detail are as follows:

```
        GENERATE 1050,1
        GATE SE,BUILDING
        STOP
        TERMINATE
```

The above segment generates a TRANSACTION at 1050 (i.e. 17.30). A TRANSACTION will wait in the GENERATE block until it is allowed entry to the GATE block i.e. until the STORAGE called "BUILDING" is empty. It will then go to the STOP block, and simulation stops and the report is printed.

The STOP block may be useful also for debugging. One can have several STOP blocks with different numbers e.g. STOP 1, STOP 2 etc. Whenever a TRANSACTION enters a STOP block with a number simulation stops, the number of the STOP block is printed and the standard output is printed.

An alternative format is STOP,GO. In this case simulation stops, a standard output report is printed and then simulation resumes.

9.2.7 The Complete Program

The complete program will now consist of the following

DATA DEFINITION STATEMENTS	FUNCTION DEFINITIONS
	VARIABLE DEFINITION
	CAPACITIES
SERVICE OPERATION SEGMENTS	OPENING TIME SEGMENT
	LUNCH CLOSING SEGMENT
	END OF SIMULATION SEGMENT
VEHICLE TRAFFIC SEGMENTS	E.T. SEGMENT
	F.D.S. SEGMENT

The data for the function definition statements are obtained by converting Tables 9.1 - 9.4 to cumulative distributions. Then the whole program and output will be as on the following pages.

Having seen (or produced) the output students could be asked to comment on it answering questions such as the following (which might involve extra runs)

1. Are there any undue delays at any point?
2. What do you think of the utilisations?
3. Could any of the CAPACITIES be reduced? What effect would this have?
4. What would happen if the arrival rate was doubled?
5. What would happen if arrival rates were not reduced during the lunch break?
6. If average queue lengths of 3 are tolerable by what factor can the arrival rate be increased?
7. What would happen if the E.T. (or F.D.S.) bays had to close for 1 hour unexpectedly?

```
* FUNCTION DEFINITION STATEMENTS
 E.T.ARRIVAL  FUNCTION,C RF$1,0,0 .1,17.5 .4,22.5 .7,27.5 .9,32.5
 +                          1,37.5
 F.D.S.ARRIVAL  FUNCTION, RF$2,0,0 .2,22.5 .5,37.5 .8,52.5 1,67.5
 CHECKIN.TIME  FUNCTION,C RF$3,0,3 .1,3.5 .25,4.5 .65,5.5 .9,6.5
 +                          1,7.5
 UNLOAD.TIME  FUNCTION,C RF$4,0,0 .1,22.5 .25,37.5 .45,52.5 .75,67.5
 +                          .8,82.5 1,97.5
 ARR.T  VARIABLE,I  C$1/750-C$1/810+1
*   CAPACITY STATMENTS
    E.T.BAY CAPACITY 3
    F.D.S.BAY CAPACITY 2
    BUILDING CAPACITY 9
*
*   SEGMENT TO CLOSE CHECK-IN UNTIL OPENING TIME AT 08.30
    GENERATE 479,1
    HOLD CHECK.IN TIME(31)
    TERMINATE
*
*   SEGMENT TO CLOSE CHECK-IN FOR LUNCH BETWEEN 12.30 AND 13.30
    GENERATE 750,1,1
    HOLD CHECK.IN TIME(60)
    TERMINATE
*
*   SEGMENT TO TERMINATE SIMULATION AFTER LAST VEHICLE HAS LEFT
    GENERATE 1050,1
    GATE SE,BUILDING
    STOP 1
    TERMINATE
```

```
*
*EXPRESS TRANSPORT SEGMENT
 GENERATE 480 TIME(FN$E.T.ARRIVAL*V$ARR.T)
 COMPARE C$1 LT 1050
 ASSIGN ARRIVAL.TIME, C$1   @  (note C$1 = current clock time)
 QUEUE CHECKIN.LINE
 HOLD CHECK.IN TIME(FN$CHECKIN.TIME)
 ENTER BUILDING
 ADVANCE TIME (1)
 QUEUE E.T.LINE
 STORE E.T.BAY TIME(FN$UNLOAD.TIME)
 LEAVE BUILDING
 PRINT 1,P$ARRIVAL.TIME,C$1 @  (1 refers to E.T.)
 TERMINATE

*FAST DELIVERY SERVICE SEGMENT
 GENERATE 480   TIME(FN$FDS.ARRIVAL*V$ARR.T)
 COMPARE C$1 LT 1050
 ASSIGN ARRIVAL.TIME, C$1
 QUEUE CHECKIN.LINE
 HOLD CHECK.IN TIME(FN$CHECKIN.TIME)
 ENTER BUILDING
 ADVANCE TIME(1)
 QUEUE F.D.S.LINE
 STORE F.D.S.BAY  TIME(FN$UNLOAD.TIME)
 LEAVE BUILDING
 PRINT 2,P$ARRIVAL.TIME,C$1
 TERMINATE
*
*SIMULATE A MAXIMUM OF 1000 VEHICLES
 START 1
```

```
*CHANGE THE RANDOM NUMBER GENERATORS FOR DAY 2

*

*FUNCTION DEFINITION STATEMENTS

E.T.ARRIVAL FUNCTION,C RF$5,0,0 .7,17.5 .4,22.5 .7,27.5 .9,32.5

+                           1,37.5

F.D.S.ARRIVAL FUNCTION,C RF$6,0,0 .2,22.5 .5,37.5 .8,52.5 1,67.5

CHECKIN.TIME FUNCTION, RF$7,0,0 .1,3.5 .25,4.5 .65,5.5 .9,6.5 1,7.5

UNLOAD.TIME FUNCTION,C RF$8,0,0 .1,22.5 .25,37.5 .45,52.5 .75,67.5

+                           .8,82.5 1.97.5


*SIMULATE A MAXIMUM OF 1000 VEHICLES

 START 1

 END
```

Note: In "START 1" the 1 is artificial as simulation will be stopped by the STOP block. In fact since simulation is to be terminated by a STOP block and not by a specific number of TRANSACTIONS it is not necessary to record the number of TRANSACTIONS and accordingly the previously used TERMINATE,R blocks have been replaced by simple TERMINATE blocks, thus no TRANSACTIONS will enter TERMINATE,R blocks so the START 1 block will never cause the simulation to terminate.

On the following pages are extracts from the output for the simulation just described. The standard output in addition to the output described in the previous unit includes details of the STORAGES used. The details provided are as follows

Maximum Contents

Average Contents

Maximum Capacity

Average Capacity

Average Utilisationl

Total Entries

Total Transaction

Average Ent/Trans

Average Time/Ent

Current Contents

In the present case since CAPACITY is never changed in the program Maximum Capacity = Average Capacity = Capacity as defined in the CAPACITY block, so these have been omitted from the output. Similarly "Current Contents" has been omitted since it is always zero when the system closes.

The LINE number on the following page of output refers to the line in the program which contained the corresponding PRINT statement.

		TYPE 1 = E.T. 2 = F.D.S.	ARRIVAL TIME	DEPARTURE TIME
LINE	49:	1	480	532
LINE	63:	2	480	556
LINE	49:	1	531	582
LINE	63:	2	495	586
LINE	49:	1	542	587
LINE	49:	1	512	593
LINE	63:	2	542	603
LINE	49:	1	577	615
LINE	63:	2	590	625
LINE	49:	1	599	676
LINE	49:	1	637	683
LINE	49:	1	617	695
LINE	63:	2	645	711
LINE	49:	1	697	727
LINE	63:	2	699	734
LINE	49:	1	685	755
LINE	49:	1	658	761
LINE	49:	1	715	788
LINE	63:	2	709	808
LINE	49:	1	746	838
LINE	63:	2	753	839
LINE	49:	1	767	847
LINE	49:	1	844	879
LINE	49:	1	866	897
LINE	63:	2	883	902
LINE	49:	1	832	925
LINE	49:	1	890	950
LINE	49:	1	903	959
LINE	49:	1	933	996
LINE	63:	2	965	1020
LINE	63:	2	941	1025
LINE	49:	1	951	1044
LINE	49:	1	979	1044
LINE	49:	1	998	1059
LINE	63:	2	1008	1103
LINE	49:	1	1027	1105
LINE	63:	2	1030	1127

```
SIMULATION ENDED BY STOP              1
AT BLOCK SEQUENCE NUMBER              9


      RELATIVE                 ABSOLUTE               TERMINATION
     CLOCK TIME               CLOCK TIME                 COUNT
       1127                      1127                      37


BLOCK    CURR    TOTAL    BLOCK    CURR    TOTAL    BLOCK    CURR    TOTAL
  #      TRAN    TRAN       #      TRAN    TRAN       #      TRAN    TRAN
  1        0       1        2        0       1        3        0       1
  4        0       1        5        0       1        6        0       1
  7        0       1        8        0       1        9        0       1
 10        0       0       11        1      25       12        0      24
 13        0      24       14        0      24       15        0      24
 16        0      24       17        0      24       18        0      24
 19        0      24       20        0      24       21        0      24
 22        0      24       23        1      14       24        0      13
 25        0      13       26        0      13       27        0      13
 28        0      13       29        0      13       30        0      13
 31        0      13       32        0      13       33        0      13
 34        0      13


      FACILITY          AVERAGE            NUMBER          AVERAGE
        NAME          UTILIZATION          ENTRIES        TIME/TRANS

      CHECK.IN           .2405               39              6.95


STORAGE    MAXIMUM     AVERAGE       AVERAGE       TOTAL      AVERAGE
 NAME      CONTENTS    CONTENTS    UTILIZATION    ENTRIES    TIME/ENT

E.T.BAY        3         1.14         .3789          24        53.37
F.D.S.BAY      2          .61         .3066          13        53.15
BUILDING       6         1.83         .2028          37        55.59


QUEUE          MAXIMUM   AVERAGE   TOTAL    ZERO     AV.TIME/ENT  AV.TIME/ENT
NAME           CONTENTS  CONTENTS  ENTRIES  ENTRIES     (ALL)      (NON ZERO)

CHECKIN.LINE      3        .19       37       28        5.76         23.67
E.T.LINE          1        .03       24       20        1.50          9.00
F.D.S.LINE        1        .01       13       11         .92          6.00
```

CHAPTER 10

UNIT EIGHT OF TEACHING PACKAGE

10.1 INTRODUCTION TO UNIT 8 OF TEACHING PACKAGE

The program of Unit 7 produced standard output only. This Unit introduces more complex data collection and the production of non-standard output using TABLE and QTABLES.

The system modelled in Unit 7 is used as an example but instead of simulating one day only, 30 days will be simulated. This will be repeated using different random numbers to ensure that results do not depend on the particular set of random numbers used. In this connection students should be asked to reread the section at the end of Unit 6, in particular that part concerning accuracy of estimation. If students are divided into groups each group could be asked to write the program using different random number generators (group one could use RN$1 to RN$4, group two RN$2 to RN$5 etc.). The results of the groups could then be compared. An example of such a comparison is given.

The function of the control statements CLEAR and RESET is explained. In this connection some time should be spent considering the effect of starting conditions in a simulation, in particular whether one should collect data in the early stages of simulation given that the simulation has started with an empty system. Again one should revise the discussion concerning continuous and terminating simulations in Unit 6.

The additional statements required for this unit are:

TABLE, QTABLE, MATRIX

MARK TABULATE SAVEX MSAVEX

CLEAR RESET

The purpose of this unit is as follows:

(a) To give more experience of material covered in earlier units

(b) To introduce more detailed output which is available on request (i.e. TABLES)

(c) To show how the analyst can produce his own output tables of information (SAVEX MATRIX)

It is suggested that students work in parallel with progress through this unit rather than at the end. Page 221 sets out the extra information which is required of the model. Items (1) and (2) can be done using previous knowledge and so could be set as a student exercise immediately while the lecturer covers the material on pages 207 - 220. When the lecturer has covered the material on 207 - 220 students could also be asked to do item (3) page 221 before receiving the material on pages 223 - 228. Before commencing with the material on page 229 students should have written programs to do items (1), (2), (3) on page 221.

The remainder of the unit deals with collection and recording of data by the programmer as distinct from data which is automatically collected by GPSS. To do this the concepts of SAVEX and MATRIX SAVEXES need to be understood. These are explained briefly on page 216 and more fully in the mini-manual. This is rather more like conventional programming e.g. FORTRAN but a little more complex as GPSS being a specialised language is not really geared to scientific calculations done by the programmer. Students with no previous programming experience may find some difficulty here.

It is useful to get students to include a SEED statement (for details see the mini-manual) for each generator used. The effect of this is that since the choice of seed will depend on the time at which the program is run each student's sequence of random numbers will be different even if all use the same generators. A possible disadvantage is that if the program is rerun the output will not be the same as new seeds will be chosen. The output however states the seeds that have been used so that it is possible to duplicate a run if required by specifying the seeds.

The complete program is given at the end of the unit. Before seeing it students should have at least attempted to write their own version.

10.2 UNIT EIGHT - ANALYSIS OF SIMULATION RESULTS

10.2.1 Non-standard Output

In addition to the standard output described in previous units it is possible to request additional information in the form of special tables. It is also possible to do some editing e.g. print special headings or draw histograms.

The standard output provides information on FACILITIES, STORAGES and QUEUES. However one might want details of durations from point to point within a system or indeed an analysis of the total time spent in a system. There are a number of ways of doing this, requiring special blocks which will be described. First however, it is necessary to consider a special block for measuring transit times, called a MARK block.

There are two formats for this block which are as follows:

```
        MARK        or        MARK,name
```

In the latter case "name" is the name of a TRANSACTION PARAMETER. In this latter case the effect of the block is to store the value of the current simulation clock time in the PARAMETER called "name". If one refers subsequently to MP$name the value returned is the time which has elapsed since the mark was set i.e. C$1-P$name. This allows one to measure the time between points. If one uses the former format i.e. without specification of "name" then a reference to M$1 returns the elapsed time but the starting time is not available.

The block in the previous model ASSIGN ARRIVAL.TIME,C$1 could be replaced by MARK,ARRIVAL.TIME.

10.2.2 Tables

It is possible to have the simulator collect data on any specified quantity and to compile a table which is printed after the standard output. How this is done is best described by an example.

Example 10.1

In the model of the previous unit compile data on the total time spent in the system from arrival time on the queue for check-in to departure from the building.

The required table must be given a name, say SYSTEM.TIME. This is done on a data definition statement TABLE with the following format:

```
SYSTEM.TIME TABLE M$1,a,b,c
```

M$1 (the time elapsed since the last MARK block is the variable to be tabulated in a table called SYSTEM.TIME. Its value will be calculated whenever a TRANSACTION enters a TABULATE block. The table will contain c+1 classes. The last class is for any overflow. The top of the first class is a and the interval is b. If a = 80 b = 30 c = 4 the table would be as follows:

TABLE 10.1

CLASS	Frequency
0 - 80	
81 - 110	
111 - 140	
141 - 170	
$\geqslant$ 171	

The instructions required in the program would be a MARK instruction just after GENERATE and a TABULATE instruction just before termination. These two blocks inserted as follows will provide the data requested in the example (10.1).

```
GENERATE .......
MARK                    @ set timer
.
.
.

TABULATE SYSTEM.TIME    @ calculate and tabulate elapsed time
TERMINATE,R
```

When a TRANSACTION enters the TABULATE block its transit time is recorded in the table SYSTEM.TIME

The format of the table produced at the end is as shown below.

TABLE 10.2

Upper Limit	Observed Frequency	Relative Frequency	Cumulative Percentage	Cumulative Remainder	Multiple of Mean	Deviation from Mean
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

In addition a summary table is produced giving the following

Number of entries

Sum of arguments

Mean argument

Standard deviation

Weighted and non-weighted versions of the above 4 items are provided.

In the models to be described weighted and non-weighted values will be identical since only weights of 1 are used. It is possible to use a block such as

```
TABULATE SYSTEM.TIME,P$WEIGHT
```

When a TRANSACTION enters the above block an entry would be made in the table SYSTEM.TIME according to the value of M$1 and with a weight specified by the PARAMETER called WEIGHT. The general format of the TABULATE block is TABULATE name, number.
If "number" is omitted it is understood to be 1.

10.2.3 Use of an Artificial Queue to Record Data on Elapsed Times

An artificial queue could be used in place of the conventional TABLE just described to monitor any time interval between system points. Thus for the system time, an arriving vechicle i.e. one just generated could join a specially defined (artificial) system queue which it leaves when it leaves the building. If activity must take place while a TRANSACTION is on a QUEUE a pair of instructions is required. This is analogous to using the SEIZE and RELEASE pair in lieu of HOLD. Thus after the GENERATE block and before any time lapse the TRANSACTION enters a block INQUEUE SYS.QUEUE,ARRIVAL.TIME and before the TERMINATE block it enters a block OUTQUEUE SYS.QUEUE,ARRIVAL.TIME. Thus for the entire time it was in the system it was a member of the QUEUE called SYS.QUEUE for which records will be automatically compiled. "ARRIVAL.TIME" is a PARAMETER name which is automatically assigned the value of the time at which the TRANSACTION joined the queue. A PARAMETER name must be specified.

Thus the following statements will also suffice for Example 10.1

```
GENERATE
INQUEUE SYS.QUEUE, ARRIVAL.TIME
.
.
.
OUTQUEUE SYS.QUEUE, ARRIVAL.TIME
TERMINATE,R
```

10.2.4 Use of QTABLE to Obtain Extra Statistical Information on Queues

It is possible to tabulate queue data in the format just described for TABLE. It is only necessary to allocate a name in a way similar to that for TABLE. The format for the data definition statement is:

```
    name of table       QTABLE    a, b, c, name of queue
```

For example in the model of Unit seven there was a QUEUE called CHECKIN.LINE. If one wanted the associated data tabulated (in addition to the standard output) one would define a table called, say, CHECK.TIME as follows:

```
    CHECK.TIME          QTABLE    15,15,20,CHECKIN.LINE
```

The first class of this table will be $t \leqslant 15$.
The second class will be $15 < t \leqslant 30$ and so on in 15-unit intervals. The maximum number of classes will equal 20 plus an overflow class if required. Note that if 21 classes are not required they will not be printed. If the maximum queue time is such that only, say, 10 classes are required then only 10 classes will be printed.

To recapitulate it is possible by means of either a MARK plus TABULATE block or an INQUEUE Plus OUTQUEUE block to compile statistical data on the time which it takes a TRANSACTION to go from the former to the latter block of the pair. The TABULATE block automatically provides a full table. The OUTQUEUE block only provides standard QUEUE details unless one defines an associated QTABLE.

Note that items are only inserted in QTABLE when they enter the OUTQUEUE block thus items still queueing when simulation stopped would not be included in QTABLE statistics. This differs from standard output on QUEUES (see page 170).

10.2.5 Repetition of a Series of Instructions

The pair of segments for E.T. and F.D.S. vehicles in the model of Unit 7 are almost identical the only differences being the GENERATE statement (different arrival rates) and a different unloading procedure (2 blocks). With as few as ten instructions in each segment it is just as easy to have two completely separate segments. As however the number of instructions in each segment increases it becomes increasingly onerous in case of changes to have to modify both segments hence doubling the probability of programming/input errors.

There are a number of ways around this. One way is to use a subroutine which in GPSS is called a MACRO. The instruction calling the subroutine specifies certain parameters which could differ for the two vehicle types. The other (and in this case, simpler) method is to use two GENERATE statements and then a common segment for processing. TRANSACTIONS can carry in a PARAMETER an identification which will distinguish the two vehicle types and allow different processing where required. The only distinction required is for unloading. The unloading blocks for E.T. and F.D.S. can be written separately with labels, allowing GO TO instuctions to distinguish between them. Suppose the blocks for E.T. have a label E.T. those for F.D.S. have a label F.D.S.: the TRANSACTIONS can carry in a parameter the appropriate label.

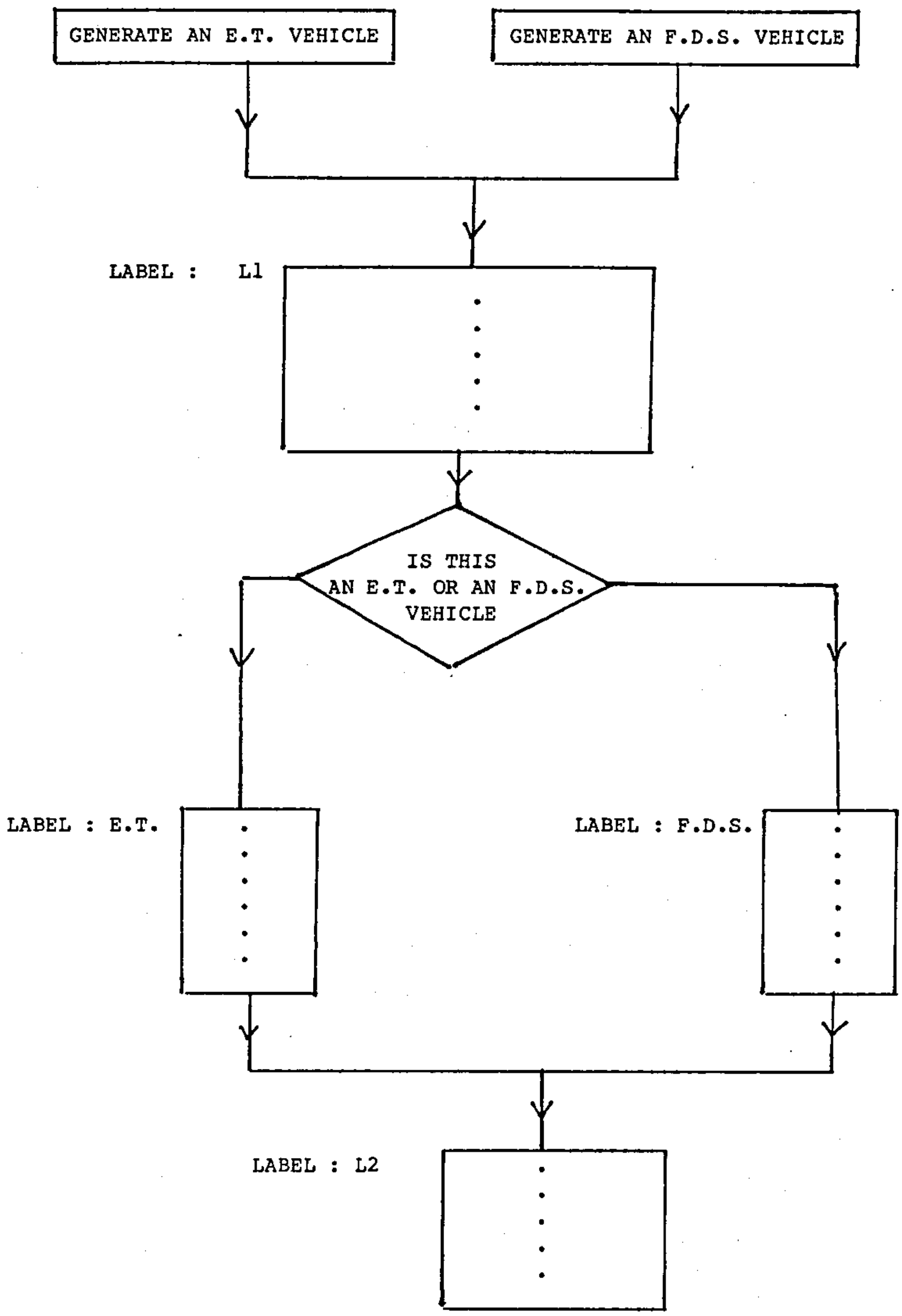
Example 10.2 Two types of vehicles are to be generated E.T. and F.D.S. After generation they go through several processes in common (i.e. it is not necessary to distinguish the vehicle types, these processes could be checking in and driving to a destination). They then go through some processes which are different for the two types (e.g. unloading). Finally they go through several more processes in common. Indicate schematically how this can be programmed.

Solution to Example 10.2

```
        GENERATE .............              (generate E.T. vehicles)
        ASSIGN BLOCK, E.T.  GO TO (L1)      (this puts the "name" E.T. in
                                             the PARAMETER called BLOCK)

        GENERATE .............              (generate F.D.S. vehicles)
        ASSIGN BLOCK, F.D.S.                (this puts "name" F.D.S. in
                                             the PARAMETER called BLOCK)

L1      ....................
        ....................                    common blocks
        ........GO TO (*BLOCK)

E.T.    ....................                    specific to E.T.
        ...........GO TO (L2)

F.D.S.  ..................                      specific to F.D.S
        ..................

L2      ....................                    common blocks
        ....................
```

Note the instruction GO TO (*BLOCK). The asterisk indicates that what follows is an indirect specification of a label i.e. BLOCK is not the label (as it would be if the instruction were GO TO(BLOCK) but is the name of a PARAMETER which contains the label. If the TRANSACTION had been generated by the E.T. generator BLOCK would contain E.T., if it had been generated by the F.D.S. generator it would contain F.D.S.

The above is illustrated in flow-chart form on the next page.


GENERATE AN E.T. VEHICLE
GENERATE AN F.D.S. VEHICLE
LABEL : L1
IS THIS AN E.T. OR AN F.D.S. VEHICLE
LABEL : E.T.
LABEL : F.D.S.
LABEL : L2

10.2.6 SAVEX and MSAVEX

A SAVEX is simply a storage location which has a name assigned by the programmer. It holds a numerical value assigned by the programmer and can be modified at any point in the program. This is done by means of a SAVEX block. The format is SAVEX X,Y

where X is the name of the SAVEX and Y is the value assigned

e.g.   SAVEX NUMBER, 3

This statement defines NUMBER as the name of a SAVEX and assigns it the value 3. If in a block the value of a SAVEX such as NUMBER is required one must use the representation X$NUMBER (see NUMERIC ATTRIBUTES in mini-manual). X here is the mnemonic for a SAVEX, $ is a conventional separator.

Thus   X$NUMBER = The value of the SAVEX called NUMBER.

Consider the block   SAVEX NUMBER, X$NUMBER +1

This assigns to the SAVEX called NUMBER the value of the SAVEX called NUMBER plus 1 i.e. whenever a TRANSACTION goes through the block the value of "NUMBER" is increased by 1, in other words it counts the number of TRANSACTIONS. Note that a block such as SAVEX NUMBER, 3 is only implemented when a TRANSACTION enters the block. If no TRANSACTION enters the block NUMBER will not have the value 3.

It is possible to have a singly subscripted SAVEX viz.,

X(1), X(2), X(3) ... etc.

Example   SAVEX X(3), X$X(1) + X$X(2)

This assigns to the SAVEX called X(3) the sum of the values of X(1) and X(2).

A doubly indexed SAVEX is called MSAVEX (matrix SAVEX).
The dimension of a subscripted SAVEX must be declared on an ORDER statement.
For a detailed description read the description in the mini-manual carefully.

Pay special attention to the distinction between a SAVEX and a PARAMETER. Note that the values of all SAVEX AND MSAVEX locations will be automatically output on termination.

Rounding

It should be noted that SAVEX and MSAVEX locations hold integers only. An arithmetic expression will be evaluated and then the result truncated to an integer i.e. rounded down.

Consider the following :

```
SAVEX    X1, 10/6
SAVEX    X2, 23/12
SAVEX    X3, X$X1 + X$X2
SAVEX    X4, 10/6 + 23/12
```

```
X1  will contain the value  1   (i.e.  1.666... truncated)
X2  will contain the value  1
X3  will contain the value  2
X4  will contain the value  3   (i.e. 1.66... + 1.9166... = 3.5833..)
```

Note particularly that X4 is not the same as X3

If one wants to ensure rounding to the nearest number then add 0.5.

Thus

```
SAVEX  X1,  10/6 + 0.5              will retain 2
SAVEX  X2,  23/12 + 0.5             will retain 2
SAVEX  X3,  X$X1 + X$X2             will retain 4
SAVEX  X4,  10/6 + 23/12 + 0.5      will retain 4
```

Exercise on the use of SAVEX and MSAVEX

To the program in Unit 7 students should add the necessary blocks to do the following

1. Count the number of TRANSACTIONS (vehicles) which TERMINATE between 1.30p.m. and 2.30p.m.

2. Produce a 2 x 3 table (MSAVEX) to provide the following information.

	Total Number	Average System Time	Number with System Time in Excess of 60 minutes
E.T. Vehicles			
F.D.S. Vehicles			

(The system time for each vehicle can be got by using a MARK block)

10.2.7 CLEAR AND RESET

One may wish to run a simulation for a period before starting to collect data. Initial conditions may be unusual e.g. the system starts empty and gradually fills up. One may want to collect data for the system when it is "in full swing" only. This is achieved by using a RESET control statement after a START statement. Consider the following sequence

```
START 50
RESET
START 200
```

The simulator will simulate 50 recorded terminations according to the START 50 statement. The RESET statement causes all data so far collected to be discarded. Then, starting with the sytem exactly as it was when the fiftieth transaction terminated simulation is resumed for a further 200 terminations collecting data for this new phase. If one assumes that after the first 50 terminations the system would be in normal operating mode then all data recorded in the second phase would refer only to this normal operating mode.

One may wish to run a model two or more times with slight changes e.g. using different random number generators or different arrival time distributions. This is done by using a CLEAR statement followed by the revised statements and then a new START statement.

Revised Statements

If the revised statement is a data definition statement then it is simply written in the same format as the original statement but with the required alterations and is put after the CLEAR statement. If the revised statement is a model block then both it and the statement which it replaces must carry labels - the same label on both.

Example 10.3

```
          START 200   (simulate 200 TRANSACTIONS using original model)

          CLEAR       (clear, to repeat)
     ARRIVAL.TIME  FUNCTION,C RF$5 .....          ] replacement
     L1   HOLD  CHECK.IN  TIME(80)                ] statements
          START 200

          CLEAR       (clear to repeat again)
     ARRIVAL.TIME  FUNCTION,C RF$6 .....          ] replacement
     L1   HOLD  CHECK.IN  TIME(90)                ] statements
          START 200
```

The simulator would simulate 200 TRANSACTIONS using the original model. Then CLEAR indicates that the model is to be cleared i.e. all data is discarded, all values reset as they were prior to commencing the simulation. The only thing not reset is the absolute clock, but the relative clock is reset to zero. The function called ARRIVAL.TIME is redefined perhaps using a different random number generator and the block labelled L1 in the original model is replaced by L1 HOLD CHECK.IN TIME(80), perhaps the time was not originally 80. Then 200 more TRANSACTIONS are simulated. This can be repeated as many times as required.

10.2.8 More Detailed Data Collection for the System Described in Unit 7

The information required of the model in Unit 7 will now be increased to include the following:

(1) Simulate 30 days of operation and produce summary data for the whole period i.e. not just for each day separately

(2) for each vehicle print the following (all times in clock times)

- (a) Company
- (b) Arrival Time on Check-in Queue
- (c) Time of Entry to Building
- (d) Departure Time
- (e) Total Time in System (in minutes)

(3) Produce statistical tables on

- (a) Queue for check-in
- (b) Total time in system

(4) Produce a table of results for each day and a summary for the whole period showing

- (a) Time of departure of the last vehicle (i.e. closing time)
- (b) Total number of vehicles for each company
- (c) Utilisation of the check-in
- (d) Utilisation of each company's bays
- (e) Average time spent in the system
- (f) Maximum time spent in the system
- (g) The number of vehicles which spent more than 2 hours and 30 minutes in the system

(5) Repeat using a different set of random numbers and compare the results. That is, produce a few runs (say n runs) each of 30 days. From each run select a measure such as the average queueing time for all vehicles, then compare the n values so obtained. Are they similar? Why do they differ? Can you give an opinion concerning the range in which the value would lie if you simulated a very large number of days (what in statistical terms would be called a confidence interval for the mean)?

(6) Repeat with a 50% increase in the arrival rate

1 <u>Effect of Time Continuing Beyond 24 Hours</u>

The 30 days will be simulated continuously. If t is a counter representing time measured in minutes from 00.00 on day 1 to 23.59 on day 30 then $0 \leqslant t \leqslant 43{,}199$. Since each day contains exactly 1440 minutes if one wishes to obtain the time within the day from t one just divides by 1440 and takes the remainder.

Example 10.4: What time and day is represented by C$1=20420?

C$1/1440 = 14 with remainder 260

So C$1=20420 is the 260th minute on day 14

i.e. 04.20 on day 14

Modulo division does this directly

i.e. 20420//1440 = 260 (the remainder when 20420 is divided by 1440)

Thus if C$1 = simulator clock time

Time within the day T = C$1//1440

A slight problem is caused by time periods which include midnight. Consider the following.

A vehicle arrives at C$1 = 1020 and leaves at C$1 = 1500. When these are converted to T values one gets

arrival time = 1020//1440 = 1020

departure time = 1500//1440 = 60

Thus if one calculates, duration = departure time - arrival time, one would get a negative number to which 24 hours would have to be added i.e. correct duration = 60 - 1020 + 1440 = 480.

To get over this problem one can count times after midnight and before 8 a.m. (i.e. before the next day starts) as if they were on the previous day! Thus 1.00 a.m. on Tuesday would be 25.00 on Monday etc. If one puts T = C$1//1440 then T needs to be modified if it is less than 480 (i.e. before 8 a.m.) by adding 1440 (i.e. 24 hours). One could do this within the program by means of a COMPARE statement as follows

```
         SAVEX  T, C$1//1440   GO TO (L1, L2)
    L1   COMPARE   X$T  LT  480
         SAVEX  T, X$T + 1440
    L2
```

Alternatively and rather more cleverly it can be done by means of an integer VARIABLE as follows.

```
         T1  VARIABLE       C$1//1440
         T   VARIABLE, I    (1-V$T1/480 + V$T1/960)*1440 + V$T1
```

Given that V$T1 is between 0 and 1439 show that the expression in brackets i.e. 1 - V$T1/480 + V$T1/960 is 1 if and only if V$T1 is between 0 and 479 inclusive, and is zero otherwise.

1(a) <u>Treatment of Arrival Rate During Lunch Break</u>

In Unit 7 in order to vary the arrival time during the lunch break use was made of an INTEGER VARIABLE called ARR.T which was defined as C$1/750-C$1/810+1 where C$1 was the current clock time within the day. In the present model C$1 is not the time within the day but the VARIABLE called T is (see previous page). Thus the single definition

```
        ARR.T     VARIABLE, I    C$1/750-C$1/810+1
```

is replaced by the pair of definitions

```
    T         VARIABLE       C$1//1440
    ARR.T     VARIABLE,I     V$T/750-V$T/810+1
```

1(b) <u>Treatment of Arrivals Before 08.00 and 17.30</u>

In the model of Unit 7 one simply started generating arrivals at 08.00 so that it was not necessary to check if the time was earlier than 08.00. In the present case since simulation continues from day 1 to day 2 and so on, the block of Unit 7 viz.,

```
COMPARE C$1 LT 1050                    (i.e. time before 17.30)
```

is not sufficient, one must also check if the time is after 08.00. To do this use a SAVEX called OPEN which will be zero initially and then set to 1 at 8.00 each day and to 0 at 17.30 each day.

Thus there will be an opening segment as follows

```
    GENERATE   480,,1  TIME (1440)    @ I.E. AT 08.00 EVERY DAY
    SAVEX    OPEN,1                   @ START ARRIVALS
    HOLD CHECK.IN TIME(30)            @ OPEN CHECK-IN AT 8.30
    TERMINATE
```

There will be an end of day segment at the end of each day which will be as follows

```
    GENERATE  1050   TIME (1440)      @ I.E. AT 17.30 EVERY DAY
    SAVEX    OPEN,0                   @ STOP ARRIVALS
    TERMINATE
```

1(c) <u>To Close Check-in During the Interval 13.30-14.30</u>

In the model of Unit 7 this was done by a special TRANSACTION which occupied the check-in for the required period. The only difference now is that instead of a single occurrence on day 1 this must be repeated each day

i.e. at intervals of 1440 minutes so the statement

```
GENERATE  750,1,1
```

will be replaced by

```
GENERATE  750,,1 TIME(1440).
```

1(d) <u>Terminating the Simulation</u>

In the model of Unit 7 this was done by means of a special TRANSACTION which was generated at t=1050 i.e. at 17.30 on day 1.

Now it must be done on day 30 so when the day counter reaches 30 simulation will be stopped by means of a STOP block. The most convenient place to put this will be in the segment compiling the day records immediately after compiling that days records (to be described on pages 229-230). The blocks required will be

```
COMPARE X$DAY EQ 30   @ DAY IS A SAVEX COUNTING DAYS
STOP
```

Since simulation is to be stopped by a STOP statement the START statement should not stop it. It should read START 1
In effect this means stop the simulation when one TRANSACTION enters a recorded TERMINATION (i.e. TERMINATE,R) block but there will be no such block so the START statement will never stop the simulation.

(2) <u>To Convert Times to Clock Times</u>

In order to print arrival times for individual vehicles it will be necessary to change the time in minutes given by V$T to clock time for convenient reading. Whole number division by 60 will give the hour.

i.e. if V1 is defined as V1 VARIABLE,I V$T/60 then V1 will contain the hour part of the time. The minutes can be got by subtracting hour part x 60 from the total minutes i.e. if V2 is defined as

```
V2 VARIABLE V$T - V$V1*60
```

then V2 contains the minutes.

To combine hours and minutes multiply the hour part (V1) by 100 and add to the minutes part i.e. define V2 as

```
V2 VARIABLE V$V1*100+V$T-V$V1*60.
```

or more simply

```
V2 VARIABLE V$V1*40 + V$T
```

In fact both definitions can be included in one viz

```
V2 VARIABLE,I (V$T/60)*40 + V$T
```

Having defined T, V1 and V2 as above, within the program a reference to V$V2 will return the clock time in standard 24 hour clock format.

(3) PRODUCTION OF STATISTICAL TABLES

To produce the required two tables two data definitions are required. For the queue-time TABLE assume that class intervals of 15 minutes will suffice and that it is unlikely that queue time will exceed 5 hours (i.e. 20 classes will suffice) and that a suitable name would be CHECK.TABLE the definition would be

```
CHECK.TABLE    QTABLE    15,15,20,CHECKIN.LINE
```

For the total system time a TABLE must be defined. Assuming that the first class can be 0-60 and thereafter 15 minute intervals will suffice, that ten classes will be sufficient and that a suitable name would be SYSTEM.TIME then the definition is:

```
SYSTEM.TIME    TABLE     M$1,60,15,10
```

It is also necessary to have a MARK block after the GENERATE block to start timing from arrival and a TABULATE block just before TERMINATE to record the elapsed time, as described on page 209. The table produced will be the same as that on page 209.

(4) Analysis of Results for Each Day

As each day requires a set of seven values this information needs to be compiled as a SAVEX MATRIX. A SAVEX (see GPSS manual) is a single valued numerical record maintained by the programmer. A SAVEX MATRIX (see GPSS Manual) is simply a table of numbers compiled by the programmer as distinct from tables automatically provided. It requires a matrix definition statement with format:

```
MATRIX     DAILY.RECORD(31,7)
```

This defines "DAILY.RECORD" as a matrix or table with dimension (31,7) i.e. thirty-one rows (30 days plus summary row) and seven columns. The items to be recorded will be dealt with one at a time.

4(a) Time of departure of last vehicle each day

This is the time at which in actual practice the system would close. The closure is achieved in the model by simply terminating immediately any TRANSACTIONS generated between 17.30 and 08.00. To determine the departure time of the last vehicle (or 17.30 whichever is later) a single TRANSACTION is generated at 17.30 (t = 1050) everyday which will wait at a GATE until the condition "building empty" is met. As soon as this condition is met the time is recorded. The same TRANSACTION stops arrivals at 17.30 (see 1(b) above). The instructions are as follows:

```
* END OF DAY SEGMENT
GENERATE    1050                          TIME (1440)
SAVEX       OPEN,0                        @ STOP ARRIVALS
GATE        SE,BUILDING
SAVEX       DAY,X$DAY+1                   @ DAY COUNTER
MSAVEX      DAILY.RECORD(X$DAY,2),V$V2    @ RECORD TIME
TERMINATE
```

4(b) The daily total number of vehicles of each type

Two counters are required. Number1 and Number2 are two SAVEXES one of which is increased by 1 after each vehicle is generated between 08.00 and 17.30. (Number1 is for E.T. and Number2 for F.D.S.)

4(c),(d),(e),(f),(g) Distribution of time spent in the system

To obtain the data required three counters will be necessary; TT, MAX and N (three SAVEXES). Whenever any vehicle leaves the building (is terminated) the time spent in the system is added to TT (total time). If the individual time exceeds the previously recorded maximum time then MAX is replaced so that MAX will record the maximum time spent in the system by any vehicle. If the time spent in the system exceeds 2.5 hours then N is increased by 1.

At days end the value of TT is divided by the total number of vehicles (NUMBER 1 + NUMBER 2 of 4(b) above) to give the average system time. This together with MAX and N is inserted in the DAILY.RECORD matrix. This is done in the end of day segment referred to in 4(a) above anywhere between the GATE and TERMINATE blocks. The SAVEXES called TT, MAX, N,NUMBER1 and NUMBER2 must be reset to zero for the next day. Values must also be added to the summary row (row 31).

5. Replication Using Different Random Numbers

After simulating 30 days one wants to repeat or replicate using different random numbers. In effect this means that one wants to do another 30 days in order to compare the results to see by how much one month could differ from another. This can be done by redefining the FUNCTION definition statements using different generators. One could re-run the whole program having made the desired changes i.e. having changed RF$1 to RF$5, RF$2 to RF$6 etc. on the FUNCTION definitions. This is not necessary. Both runs can be done at once by using a CLEAR statement followed by the redefined FUNCTION definitions, followed by another START statement

Thus to simulate 30 days once only, the last two blocks would be

```
        .

        .

        .

        START 1

        END
```

To rerun with redefined FUNCTIONS this becomes

```
        .

        .

        .

        START 1                                  @ FIRST RUN

        CLEAR                                    @CLEAR ALL STATISTICS

        E.T.ARRIVAL    FUNCTION,C     RF$5        ......

        F.D.S.ARRIVAL  FUNCTION,C     RF$6        ......

        CHECKIN.TIME   FUNCTION,C     RF$7        ......

        UNLOAD.TIME    FUNCTION,C     RF$8        ......

        START 1                                  @ SECOND RUN

        END
```

The original generators 1 to 4 were replaced by generators 5 to 8 for the second run.

An alternative to using different random number generators is to use the same ones but to start at a different point in the sequence. This can be accomplished by using a SEED block (see GPSS mini-manual) for each generator. The seed of a generator is the first number of the sequence to be generated. Each generator in GPSS has its own seed, multiplier and increment (For a definition of these terms see Appendix A). However the user may if he wants specify his own values for these by using a SEED block, or that in fact an arbitrary value be selected in a random manner by GPSS. This is described in the mini-manual. The format is as follows

```
SEED    n, o, SAME, MULT
```

The four items n, o, SAME, MULT refer respectively to the Generator, Multiplier, Increment and Seed.

n	is the number of the generator $1 \leqslant n \leqslant 10$
o	indicates that GPSS should assign a random multiplier
SAME	indicates that the increment should not be changed
MULT	indicates that the seed should equal the multiplier

It is normal to have the seed equal to the multiplier although it is not necessary.

It is not necessary for the user to understand the relationship between seed, increment and multiplier. It sufficies to understand that the SEED block above has the effect of changing the starting point in the sequence of random numbers to be generated by random number generator n.

The following sequence of instructions will also carry out two independent replications.

```
        .
        .
        .
1     START  1                    @ FIRST RUN


2     CLEAR                       @ CLEAR FOR SECOND RUN
3     SEED   1,o,SAME,MULT
4     SEED   2,o,SAME MULT
5     SEED   3,o,SAME MULT
6     SEED   4,o,SAME,MULT
7     START  1                    @ SECOND RUN


8     END
```

The blocks numbered 2-7 above can be repeated any number of times for as many replications as one needs.

6. Change in Arrival Rate

This can be achieved in the same way as that described on page 231 for redefining FUNCTIONS. In that case the random number generators were changed. In this case the actual distribution (i.e. the data) would be changed. Of course only the arrival rate distributions need be changed. In this case since all that is required is to multiply the inter-arrival times by 2/3 one could use the same distribution and simply multiply each value returned by 2/3 by suitably altering the GENERATE statements.

```
          .

          .

          .

          .

          .

          START     1                     @ FIRST RUN


          START
     L20  GENERATE  480              TIME(FN$E.T.ARRIVAL*2/3)
     L21  GENERATE  480              TIME(FN$F.D.S.ARRIVAL*2/3)

          START     1                     @ SECOND RUN
```

In the above sequence between the two START statements the GENERATE statements of the original model(i.e. as defined before the first START statement) have been replaced or overlaid with arrival times now multiplied by 2/3.

10.2.9 Comparison of Results from Different Groups

Suppose that 10 student groups each simulated one month and that each group was asked specifically to find

(a) The average system time (averaged over all vehicles for the month)

(b) The maximum time spent in the system by any vehicle

(c) The maximum daily average system time

(d) The percentage of vehicles which spent more than 2.5 hours in the system

(e) The maximum time for 95% of vehicles

The 10 groups could then collate their results and estimate confidence ranges for the five values. Depending on the background of the students these confidence ranges could be simply commonsense estimates based on observation or they could be statistical intervals e.g. 95% confidence intervals.

The owner of the system when asked to state criteria which must be met states the following

1. The average daily system time must never exceed 90 minutes
2. No vehicle should have to spend more than 3 hours in the system
3. Most vehicles should be cleared in under 2.5 hours.

Students should be asked to state whether the system meets these criteria, and whether in view of their findings they regard the stated criteria as being reasonable. If not, how would they specify the criteria in forms more suited to sampling experiments.

Comment on Stated Criteria

The students should see that absolute criteria such as 1 "The average ... NEVER exceed 90 minutes" and 2 "NO vehicle ... system", are very difficult to satisfy and can never be verified by sampling. Even if one samples 100 or 1000 days and never observes a vehicle spending more than 3 hours in the system one cannot conclude that therefore no vehicle will ever spend more than 3 hours in the system. In practice although people may talk in absolutes they usually do not mean them. Thus if pressed the owner might agree that 1 and 2 can be replaced by:

1. It must be reasonably certain that on 95 out of 100 days the average daily system time will be less than 90 minutes.

2. It must be reasonably certain that 99% of vehicles are cleared in less than 3 hours

3. It must be reasonably certain that 95% of vehicles are cleared in less than 2.5 hours.

The students can now be asked whether the system satisfies these modified criteria. If not by what factor would the arrival rates need to be reduced so that the system does meet the criteria for successful operation? If the system does satisfy the modified criteria by what factor can the arrival rates be increased before the system fails to meet the criteria and which criterion is the first to be failed?

<u>Results of Four Replications (Student Groups) Compared</u>

Each replication consists of 30 simulated days.

Replication	Average System Time	Maximum Time	Maximum Daily Average	Percentage Exceeding 2.5 Hours	Maximum Time for 95%	Percentage Exceeding 3.0 Hours
1	71	188	91	3.2	130	0.2
2	72	193	85	2.8	130	0.1
3	71	190	88	2.5	120	0.1
4	73	180	84	2.4	130	0.0

Are the stated criteria met?

1. The average daily system time on one day (out of 120) exceeded 90 minutes so this criterion was met.
2. Several vehicles (0.1%) exceed 3 hours in the system so ths criterion was not met.
3. The percentage exceeding 2.5 hours was only 2.7% (average of four replications) so that this criterion was met.

If one considers the modified, more reasonable criteria then they are met.

10.2.10 Program

A suggested program for carrying out all of the steps required follows.

On page 1 of the program output are the various data definitions which precede the program proper as well as two program segments.

Segment 1   which starts traffic and opens the check-in.

Segment 2   which closes the check-in for lunch.

On page 2-3 of the program output is the daily report segment which stores all of the information collected during a simulated day in a row of the matrix called DR (for Daily Record). This segment also closes the check-in at 17.30 and stops the simulation after the specified number of days have been analysed (30 in the example, see block number 39).

Page 4 of the program contains the model proper that is the generation and processing of traffic and the compilation of statistics required for the daily report segment.

Note that the final statement in the output program is START 1 although in the program as input this was followed by a sequence of instructions similar to those on page 199 for the purpose of carrying out subsequent replications. In the output the first START statement is followed by the output statistics i.e. the results for the first run. Then follows the CLEAR statement, the SEED statements and the next START statement. Then follows the output for the second run or replication and so on for as many replications as have been requested.

Following the program is a sample of a thirty day report. The final row (31) is a summary. The replications referred to on page 237 were abstracted from four such reports.

```
  *
  *
  *
  *
  *
  *
  JOB
  *
  **********   MODEL FOR UNIT 8    **********
  *
  *
  * TABLE AND MATRIX DEFINITIONS
     MATRIX DR(031,10)
     CHECK.TABLE QTABLE 15,15,20,CHECKIN.LINE
     AV.SYS.TIME TABLE  MX$DR(X$DAY,5),60,5,20
     SYSTEM.TIME TABLE M$1,60,30,10
  *
  *
  * DEFINING THE CAPACITIES OF THE VARIOUS STORAGES
      E.T.BAY CAPACITY 3
      F.D.S.BAY CAPACITY 2
      BUILDING CAPACITY 9
  *
  *
  *  VARIABLE DEFINITIONS
  T1      VARIABLE   C$1//1440                  @ TIME WITHIN THE DAY
  T       VARIABLE,I (1-V$T1/480+V$T1/960)*1440+V$T1
  *    ABOVE IS TO SHOW TIMES BEFORE 8 A.M. AS IF ON THE PREVIOUS DAY AS
  *    THIS WOULD BE FOR CLEARING VEHICLES WHICH ARRIVED ON THAT DAY
  *    THUS 7 A.M. ON DAY 3 WOULD APPEAR AS 31.00 ON DAY 2
  ARR.T   VARIABLE,I V$T/750-V$T/810+1    @ WILL BE 2 IF 749<T<810 ELSE 1
  V2      VARIABLE,I (V$T/60)*40+V$T            @ THE TIME IN 24 HOUR FORMAT
  *
  *
  * FUNCTION DEFINITION STATEMENTS
  *
    E.T.ARRIVAL FUNCTION,C RF$1,0,0 .1,17.5 .4,22.5 .7,27.5 .9,32.5 1,37.5
    F.D.S.ARRIVAL FUNCTION,C RF$2,0,0 .2,22.5 .5,37.5 .8,52.5 1,67.5
    CHECKIN.TIME  FUNCTION,C RF$3,0,0 .1,3.5 .25,4.5 .65,5.5 .9,6.5 1,7.5
    UNLOAD.TIME   FUNCTION,C RF$4,0,0 .1,22.5 .25,37.5 .45,52.5 .75,67.5
  +.8,82.5 1,97.5
  *
  *
  *
  * SEGMENT TO START TRAFFIC AT 8 A.M. AND OPEN CHECK-IN AT 8.30 A.M.
  *
1      GENERATE 480,,1  TIME(1440)  @ GENERATE A PRIORITY 1 TR. AT 8 DAILY
2      SAVEX    OPEN,1                  @ ALLOW ARRIVALS TO COMMENCE
3      HOLD CHECK.IN TIME(30)           @ KEEP CHECK-IN CLOSED UNTIL 8.30
4      TERMINATE
  *
  *
  * SEGMENT TO CLOSE CHECK-IN FOR LUNCH BETWEEN 12.30 AND 13.30
  *
5      GENERATE 750,,1  TIME(1440)
6      HOLD CHECK.IN TIME(60)
7      TERMINATE
  *
  *
  *
  *
  *
  *
  *
  *
  *
  *
  *
  *
  *                                                           PAGE....2
  *
  * SEGMENT TO COMPILE DAILY REPORT
  *
  *
 8        GENERATE 1050 TIME(1440)
 9        SAVEX     OPEN,0                     @ CLOSE THE CHECK-IN
10        GATE SE,BUILDING
11        SAVEX NUMBER3,X$NUMBER1+X$NUMBER2  @ TOTAL NUMBER OF VEHICLES
12        SAVEX DAY,X$DAY+1                 @ DAY UPDATE
13        SAVEX  LT,V$V2                  @ TIME AT WHICH LAST VEHICLE LEFT
  *
  * FOR COLUMN 8  5.1  IS THE MEAN CHECK-IN TIME , V$T-540 IS THE
  *  DURATION FOR WHICH THE CHECK-IN WAS OPEN  SIMILARLY COLS 9,10
  *
  *  AS MSAVEX LOCATIONS HOLD VALUES TRUNCATED TO INTEGERS, IN ORDER TO
  *  ENSURE ROUNDING TO THE NEAREST INTEGER ADD 0.5 AS IN COLUMNS 5 8 9 10
  *  BELOW
  *
14        MSAVEX DR(X$DAY,1),X$LT        @ TIME AT WHICH LAST VEHICLE LEFT
15        MSAVEX DR(X$DAY,2),X$NUMBER1  @NUMBER OF E.T. VEHICLES
16        MSAVEX DR(X$DAY,3),X$NUMBER2  @ NUMBER OF F.D.S. VEHICLES
17        MSAVEX DR(X$DAY,4),X$NUMBER3  @ TOTAL NUMBER OF VEHICLES
  *
18        MSAVEX DR(X$DAY,5),X$TT/X$NUMBER3+.5 @AVERAGE TIME IN THE SYSTEM
19        MSAVEX DR(X$DAY,6),X$MAX       @MAXIMUM TIME IN THE SYSTEM
20        MSAVEX DR(X$DAY,7),X$N         @ NUMBER WHICH EXCEEDED 150 MINS.
21        MSAVEX DR(X$DAY,8),X$NUMBER3*510/(V$T-540)+.5 @ UTILIS. OF CHECKIN
22        MSAVEX DR(X$DAY,9),X$NUMBER1*1775/(V$T-480)+.5 @ UTILIS. OF ET BAY
23        MSAVEX DR(X$DAY,10),X$NUMBER2*2662.5/(V$T-480)+.5 @ UTILIS. OF FDS
  *
  *
  *
  *
  *
  *
  *
  *
  *
  *
  *
  *
```

```
*
*
*
*
*
*
*
*
*
*
*
*
*
*
                                                                      PAGE....3
******** SUMMARY DETAILS  ***********
*
*
*  ROW 31 IS A SUMMARY ROW GIVING OVERALL TOTALS, MAXIMA OR AVERAGES
*  AS APPROPRIATE
        ADVANCE        GO TO(L1,L2)
 L1     COMPARE X$LT GT MX$DR(031,1)
        MSAVEX DR(031,1),X$LT
 L2     MSAVEX DR(031,2),MX$DR(031,2)+X$NUMBER1
        MSAVEX DR(031,3),MX$DR(031,3)+X$NUMBER2
        MSAVEX DR(031,4),MX$DR(031,4)+X$NUMBER3
        SAVEX  DR5,X$DR5+MX$DR(X$DAY,5)  GO TO(L3,L4)   @  SUM TOTAL TIMES
 L3     COMPARE X$MAX GT MX$DR(031,6)
        MSAVEX DR(031,6),X$MAX
 L4     MSAVEX DR(031,7),MX$DR(031,7)+X$N
        SAVEX  DR8,X$DR8+MX$DR(X$DAY,8)          @  ACCUMULATE FOR AVERAGE
        SAVEX  DR9,X$DR9+MX$DR(X$DAY,9)          @  ACCUMULATE FOR AVERAGE
        SAVEX DR10,X$DR10+MX$DR(X$DAY,10)        @  ACCUMULATE FOR AVERAGE
 *
        TABULATE AV.SYS.TIME      @ RECORD AVERAGE TIME FOR ANALYSIS
        ADVANCE   GO TO(L8,L10)
 *
 *
 L8     COMPARE  X$DAY GE 30         @ CHECK IF THIS IS THE END OF THE RUN
 *
 *      COMPUTE AVERAGES
        MSAVEX  DR(031,5),X$DR5/X$DAY+.5    @ .5 IS TO ROUND TO THE
        MSAVEX  DR(031,8),X$DR8/X$DAY+.5    @ NEAREST WHOLE NUMBER
        MSAVEX  DR(031,9),X$DR9/X$DAY+.5
        MSAVEX  DR(031,10),X$DR10/X$DAY+.5
 *
        STOP                                 @ END OF RUN
 *
 *
 *            ZERO COUNTERS FOR THE NEXT DAY
 *
 L10    SAVEX MAX,0
        SAVEX  TT,0
        SAVEX N,0
        SAVEX NUMBER1,0
        SAVEX NUMBER2,0
        TERMINATE
 *
 *
 *
 *
```

```
*
*
*
*
*
*
*
*
*
**********    MODEL PROPER   ************
*
*
* SEGMENT TO GENERATE EXPRESS TRANSPORT VEHICLES
*
51 L20   GENERATE 480 TIME(FN$E.T.ARRIVAL*V$ARR.T)
52       COMPARE X$OPEN GT 0
53       ASSIGN TYPE,1
54       ASSIGN BLOCK,E.T.  GO TO(L7)
*
* SEGMENT TO GENERATE FAST DELIVERY SERVICE VEHICLES
*
55 L21   GENERATE 480 TIME(FN$F.D.S.ARRIVAL*V$ARR.T)
56       COMPARE X$OPEN GT 0
57       ASSIGN TYPE,2
58       ASSIGN BLOCK,F.D.S.
*
*
* SEGMENT TO PROCESS VEHICLES
*
*        CHECK-IN SEGMENT
*
59 L7    MARK
60       ASSIGN ARRIVAL.TIME,V$V2
61       QUEUE CHECKIN.LINE
62       HOLD CHECK.IN TIME(FN$CHECKIN.TIME)
63       ASSIGN ENT,V$V2
64       ENTER BUILDING
65       ADVANCE TIME(1) GO TO(*BLOCK)
*
*              BLOCKS SPECIFIC TO E.T. VEHICLES
66 E.T.  SAVEX NUMBER1,X$NUMBER1+1
67       QUEUE E.T.LINE
68       STORE E.T.BAY TIME(FN$UNLOAD.TIME) GO TO(L9)
*
*              BLOCKS SPECIFIC TO F.D.S. VEHICLES
69 F.D.S. SAVEX NUMBER2,X$NUMBER2+1
70       QUEUE F.D.S.LINE
71       STORE F.D.S.BAY TIME(FN$UNLOAD.TIME)
*
*             COMMON BLOCKS        UPDATING RECORDS
72 L9    LEAVE BUILDING
73       TABULATE SYSTEM.TIME   GO TO(L5,L6)
74 L5    COMPARE M$1 GT X$MAX     @ IS SYSTEM TIME GREATER THEN PREVIOUS MAX
75       SAVEX MAX,M$1                 @ IF SO STORE NEW MAX
76 L6    SAVEX TT,X$TT+M$1 GO TO(L12,L13) @ ADD SYSTEM TIME TO TOTAL
77 L12   COMPARE M$1 GT 150            @ IS SYSTEM TIME GREATER THAN 150
78       SAVEX N,X$N+1                 @ IF SO INCREMENT COUNT
79 L13   TERMINATE
*
 START 1      @ SIMULATE   TERMINATING WHEN STOP BLOCK ENCOUNTERED
```

	CLOSING TIME	E.T. TOTAL	F.D.S. TOTAL	ALL VEHICLES	MEAN SYSTEM TIME	MAXIMUM SYSTEM TIME	NUMBER EXCEEDING 2.5 HOURS	CHECK-IN UTILISATION	E.T. BAY UTILISATION	F.D.S. BAY UTILISATION
ROW 1	1847	24	13	37	66	103	0	32	66	53
ROW 2	1754	23	15	38	75	149	0	36	69	67
ROW 3	1828	22	16	38	85	183	3	34	62	68
ROW 4	1818	23	14	37	68	126	0	34	66	60
ROW 5	1931	22	15	37	75	137	0	30	57	58
ROW 6	1809	23	14	37	79	129	0	34	67	61
ROW 7	1847	23	18	41	91	170	1	36	63	74
ROW 8	1837	26	15	41	67	139	0	36	72	63
ROW 9	1826	23	12	35	67	128	0	32	65	51
ROW 10	1915	21	18	39	72	168	1	32	55	71
ROW 11	1832	20	15	35	69	118	0	31	56	63
ROW 12	1828	23	17	40	60	159	1	36	65	72
ROW 13	1850	23	14	37	77	143	0	32	63	57
ROW 14	1847	24	17	41	69	154	1	36	66	70
ROW 15	1827	21	15	36	61	123	0	32	59	64
ROW 16	1922	24	18	42	73	152	1	34	62	70
ROW 17	1839	24	15	39	74	145	0	34	67	63
ROW 18	1825	23	15	38	71	105	0	34	65	64
ROW 19	1836	22	15	37	73	137	0	33	61	63
ROW 20	1835	24	16	40	63	111	0	35	67	67
ROW 21	1925	23	17	40	81	174	3	33	60	66
ROW 22	1928	23	17	40	75	154	2	32	59	66
ROW 23	1814	22	13	35	66	125	0	32	64	56
ROW 24	1911	22	16	38	63	118	0	32	58	63
ROW 25	1842	21	19	40	79	188	1	35	58	79
ROW 26	1846	21	16	37	73	146	0	32	58	66
ROW 27	1839	23	17	40	67	125	0	35	64	71
ROW 28	1822	21	16	37	59	123	0	34	60	68
ROW 29	1923	27	16	43	69	147	0	35	70	62
ROW 30	1859	24	16	40	67	157	1	34	65	65
ROW 31	1931	685	470	1155	71	188	15	34	63	65

CHAPTER 11

UNIT NINE OF TEACHING PACKAGE

11.1 INTRODUCTION TO UNIT 9 OF TEACHING PACKAGE

Having completed units 1-8 students should be able to write reasonably complex simulations. This unit introduces a real-life system requiring simulation. The system itself is described together with simplified traffic rules. Suggestions are made on the modelling of various physical entities such as ramps. Having completed the material in this unit students should write the required program themselves.

Appendix F which is a report on the actual simulation describes how the model may be made more realistic in a sequence of steps proceeding from the basic model to more complex ones. The material in this unit describes what is referred to in Appendix F as the preliminary model. Appendix F gives a description of the proposed building and contains engineering drawings of the building and floor plan. At the time at which the simulation was requested most design decisions had already been made e.g. the possibility of taking material up and down on lifts rather than driving had already been considered and rejected. The purpose of the simulation was to examine the operation of the system as it had been designed.

The only new GPSS concept required for this chapter is the indexing of entities (see GPSS mini-manual)

Additional statements required are :

ORDER INQUEUE OUTQUEUE

11.2 UNIT NINE - SIMULATION OF A REAL SYSTEM

Previous units have been of an introductory nature, introducing ideas of simulation and GPSS. In this unit models will relate to the actual case-study which forms the basis of this package.

11.2.1 Description of the Problem

11.2.1.1. The Building

A company in Hong Kong was planning the construction of a nine-storey warehouse (called a "godown" in Hong Kong). Each floor above the ground would contain eight parking bays where vehicles could be loaded or unloaded. The ground floor was to be solely for checking in, maneouvering and general storage. In addition there was to be an open roof area which could be used for temporary parking but not for loading/unloading. Access to the floors would be via a spiral ramp between floors divided into two lanes with one-way traffic on each. A simulation was required in order to assess the relationship between arrival rate of vehicles and queueing delays at entry and within the system and also to compare different sets of traffic rules.

11.2.1.2. Vehicles

Two types of vehicle were considered, a lorry and a container. Loading/unloading times would be different for the two types as would rules for progress within the system. In actual practice it is probable that vehicles would belong to different customers who might rent a whole floor, part of a floor or several floors so that arrival rates for different floors might be different. However at the early planning stage the floors had not been let and it was decided to regard all floors as having the same arrival rate. A vehicle on arrival (in the simulation) would be allocated a random number in the range 1 - 8, being the destination floor.

11.2.1.3. Traffic Rules

Initially these rules were quite simple. Then as more experience was gained both from the simulation and from physical trials which were carried out simultaneously they became more complex. This is a good method for doing simulation in any case i.e. start with a simple model and then add on sophistications as required. It is certainly the easiest way to proceed.

As far as traffic was concerned there were a number of considerations particularly the following:

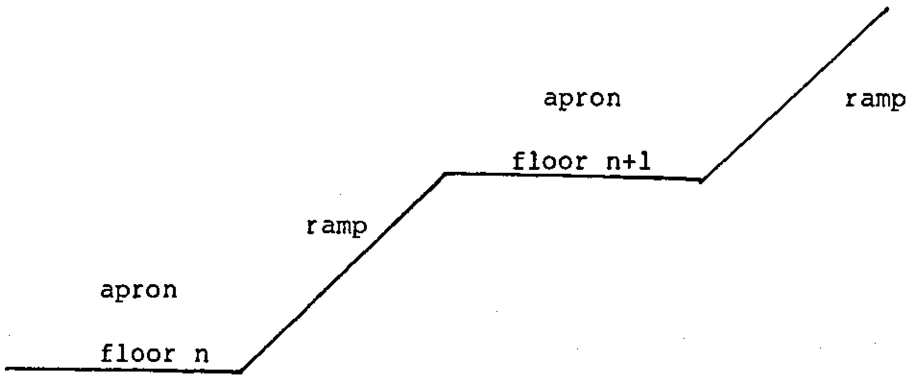
A. It would obviously be impossible for one vehicle to overtake another travelling in the same direction but could vehicles travelling in opposite directions pass each other on a ramp particularly if both were containers?

B. How does a vehicle get from a ramp to a bay? This would presumably involve stopping and manoeuvering on an apron area between the ramp and bays. While this was happening would vehicles following the manoeuvering vehicle have to stop and would vehicles trying to leave the same floor have to wait?

C. Could a vehicle, particularly a container, stop on a sloping ramp or should it only be allowed to stop on an apron area?

D. Could more than one vehicle park on the same apron (the flat area between ramps)?

The diagram here is schematic.
The actual ramps spiral.



E. If a vehicle arrives at the check-in and finds that all of the bays on its assigned floor are occupied what does it do

(a) wait, blocking following traffic

(b) move aside (if possible) allowing following traffic to pass

(c) go up to the floor and wait on the apron

(d) go up to the roof and wait there?

(a) is impractical, (b) and (c) are impossible in the actual situation so one is left with (d). This leads to consequent problems e.g. what happens if a vehicle leaves the fully occupied floor while the arriving vehicle is ascending? Should it still proceed to the roof? Obviously it would have to if it had passed its destination floor since it cannot turn except on the roof but what if it had not? Suppose that an arriving vehicle (call it A) required a space on floor 1 but floor 1 was fully occupied so A was directed to the roof. Just after A passed floor 1 a vehicle left floor 1 but A must continue to the roof as it cannot turn. Then a second vehicle B arrives also requiring a space on floor 1 which now has one vacant space. Should the space be reserved for A which arrived first but is now ascending to the roof or should it be allocated to B? The possibilities appear to be endless. In addition, the rules must appear reasonable to the driver who can only see part of the system and who is not concerned with overall optimisation but only personal optimisation.

F. Where either an ascending or descending vehicle must wait should there be any priority for the descending vehicle (or for the ascending one)?

There are not necessarily any answers to the questions posed in 11.1.3.1 - 11.1.3.6. In some cases physical considerations will impose an answer. In other cases it is up to the simulation to provide the most efficient and/or acceptable rule. The most efficient rule would not necessarily be acceptable. For instance it is possible that the most efficient rule is to say that all arriving traffic should go directly to the roof which contains a very large parking area, parking not being allowed on ascent. Vehicles could be checked in on the roof (there would be no street queue) and sent down to their required floor. This would not be acceptable as a driver would be very angry if he had to pass his floor where he could see several empty spaces. A good solution which is implementable is better than an optimum one which is not!

Since in practice vehicles could not be allowed to wait while ascending or at the check-in (as this would block following traffic) it was decided that in the model any vehicle whose requested floor was not available when the vehicle was at the check-in must go to the roof and there await clearance to descend to its required floor.

11.2.1.4. Arrival Rates

Obviously arrival rates will depend on how the floors are ultimately disposed of, that is to say the type of customer who takes them up, his traffic flow etc. The best one can do is to observe other similar warehouses to estimate traffic distribution over the day e.g. what percentage of total daily traffic is in the period 12.00 - 13.00 and then apply this distribution to different total daily figures. This is what was done. Observations lead to the conclusion that for a large part of the day the arrival rate was constant. i.e. no significant peaks or troughs.

11.2.1.5. Transit times within the building

A. Check-in Time

This would vary little from vehicle to vehicle and a constant value was used viz. 30 seconds. It was envisaged that if check-in was a problem it might be possible to check-in several vehicles simultaneously by using several clerks. This can be tried in later models as is the case in models in Appendix F.

B. Drive-time between floors

This was estimated to be 30 seconds based on trials. Since there is no overtaking and since it was decided that no vehicle could stop on a ramp (except in case of an accident) this should be relatively constant. The program at the end of this unit assumes that 80% take 30 seconds and 20% take 1 minute (irrespective of type).

C. Time to manoeuvre from ramp to bay across the apron or vice versa

This was estimated to require 2.5 minutes for all vehicles in the initial model but was subsequently modified to different values for lorry and container as a result of trials.

D. Loading/unloading

From observation of similar operations it was estimated that the dock-time for a lorry would be 30 minutes and for a container 90 minutes.

11.2.1.6. <u>Times of operation of the system</u>

No definite decision had been made as to whether the sytem would operate on a 24-hour basis or whether it would only operate for say 8 hours per day. In practise most simulation runs assumed a constant arrival rate for 8 hours after which no more were accepted but the system would continue to process vehicles which were already inside. This would be the case if the system operated on an open for 8 hours only basis and would approximate the situation if the operation was for 24 hours since in the latter case it could be expected that the number of arrivals during the night would be very small and there would be no traffic problems. Whatever problems were to occur would occur during the 8 hours 9.00a.m. - 5.00p.m. when most traffic would arrive no matter what the opening hours were to be.

11.2.2 Modelling Physical Entities

11.2.2.1 Floors

Each floor can be represented by a STORAGE with capacity eight. Instead of named STORAGES GPSS allows one to use indexed storage locations using the mnemonic S for a storage location, thus S(1) is the first floor S(2) the second and so on. S must be dimensioned on an ORDER statement as follows:

```
                ORDER,S 9            this specifies 9 STORAGE locations.
```

The ninth STORAGE is the roof. If one wishes to refer to this specifically as ROOF it is possible to do this by using two ORDER statements as follows:

```
    ORDER S 8,  ROOF

    ORDER S 9
```

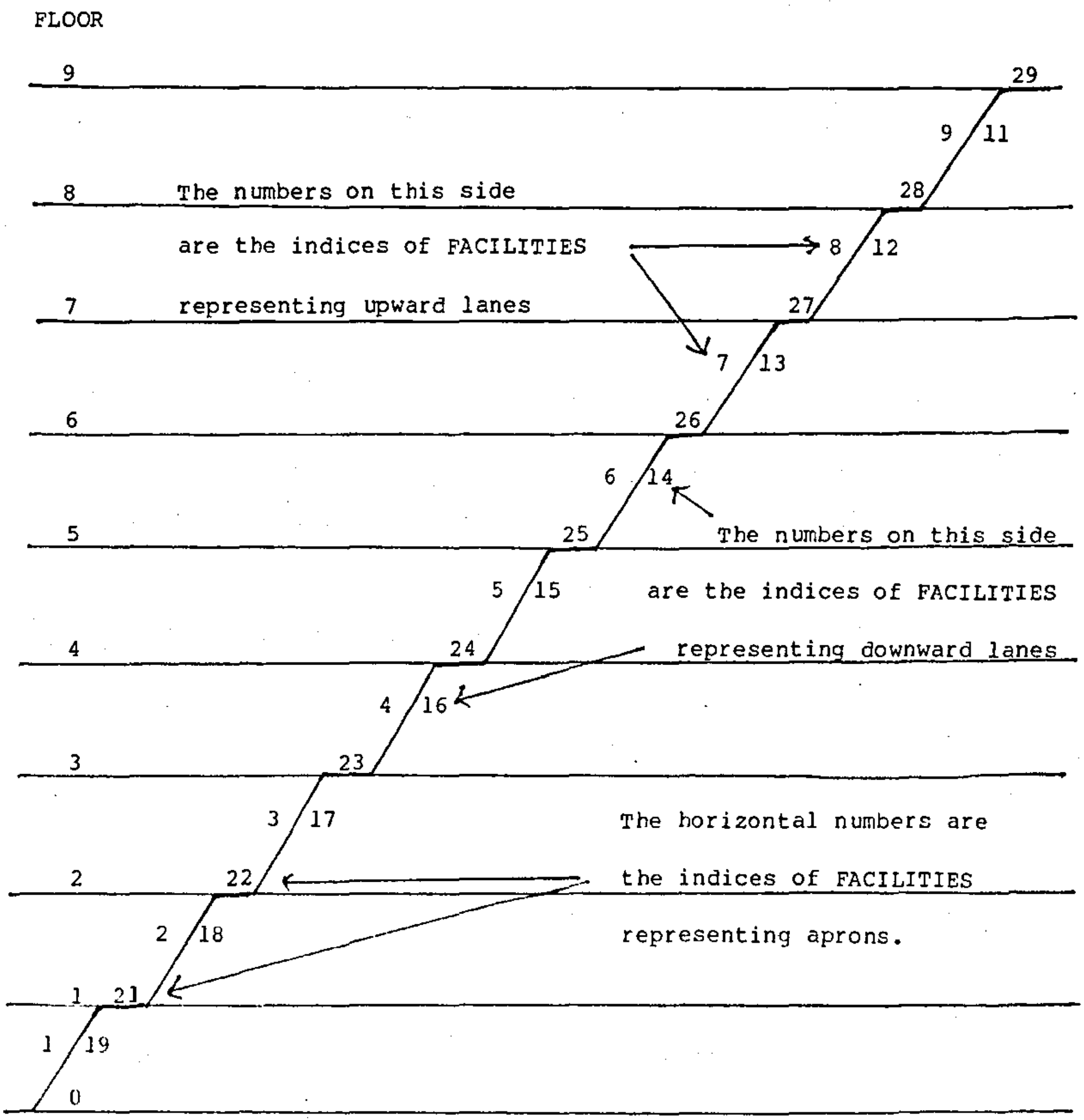
The first ORDER statement allocates the first 9 storage locations to S(1) .. S(8), ROOF.

The second ORDER statement allocates the first 9 storage locations to S(1) ... S(9) so that implicitly ROOF and S(9) are synonymous. The capacity of ROOF was set at 200. This is deliberately too large as the intention was to see what was the greatest number which would be on the roof at one time. All STORAGE locations [S(1), S(2) etc.] do not have to have the same capacity. Note that only one indexed STORAGE can be used and it must be called S(i), where i = index, so that one could not have another indexed STORAGE called say ST(i). Thus if one has a batch of eight STORAGES on floor one say and a batch of eight STORAGES on floor two, one would like to call them S1(1-8) and S2(1-8) but this is not possible, one must call the STORAGES On floor one S(1-8) and the STORAGES on floor two S(9-16). Refer here to the section on indexing in the GPSS mini-manual.

11.2.2.2 Ramps and Forecourts (Aprons)

As it was decided that only one vehicle at a time could occupy one lane of a ramp each lane (2 per ramp) can be represented by a FACILITY. There are 9 ramps (ground-floor 1, floor 1 - floor 2, ... floor 8 - floor 9) with 2 lanes each so 18 FACILITIES would be required for this purpose. Again these are indexed. Each apron or forecourt could also be occupied by only one vehicle at a time (at least in the earlier simpler models). So they could also be represented by FACILITIES. The indexed FACILITIES were allocated as follows:

F(i) = ramp from floor i - 1 up to floor i	i=1, 2 ... 9
F(i) = ramp from floor 20 - i down to floor 19 - i	i=11,12...19
F(i) = apron on floor i - 20	i=21, ...29


FLOOR
9
29
9
11
8
The numbers on this side
are the indices of FACILITIES
representing upward lanes
28
8
12
7
27
7
13
6
26
6
14
5
25
The numbers on this side
are the indices of FACILITIES
representing downward lanes
5
15
4
24
4
16
3
23
3
17
The horizontal numbers are
the indices of FACILITIES
representing aprons.
2
22
2
18
1
21
1
19
0

11.2.2.3 Vehicles

Vehicles are represented by "TRANSACTIONS" each of which carries a set of PARAMETERS relating to the vehicle. These PARAMETERS are mainly assigned immediately on generation (arrival). The PARAMETERS required are as follows:

TYPE	This is 1 or 2 depending on whether it is a container or lorry
FLOOR	This is the destination, a number in the range 1 to 8.
ARRIVAL.TIME	This is the arrival time
POSITION	This indicates the position at any time i.e. the floor. It is initially assigned a value of zero and is updated as the vehicle proceeds up or down through the building.

There are some other PARAMETERS to be described later which are used as file numbers for recording purposes. They are in fact combinations of TYPE, FLOOR and ARRIVAL.TIME.

11.2.3 Modelling Progress Through the System

The progress through the system is basically a series of SEIZE and RELEASE statements as the TRANSACTION (vehicle) successively occupies FACILITIES (ramps and aprons) to the exclusion of other vehicles for stated periods of time. There will also be QUEUES at points where a vehicle must wait.

11.2.4 Data Collection

Summary statistics mainly on the total time spent in the system will be required broken down into several categories as follows:

(1) By vehicle type (lorry, container, all)

(2) By destination floor and vehicle type

(3) By time of day and vehicle type

Intermediate waiting time distributions as follows:

(1) Waiting time at check-in

(2) Waiting time on apron n for permission to proceed up the next ramp

(3) Waiting time on apron n for permission to descend to n - 1

(4) Waiting time for apron prior to docking

(5) Waiting time for apron on completion of loading/unloading

(6) Waiting time on roof for available space on required floor

All of the above can be obtained by specifying appropriate QUEUES. It should be understood that "QUEUES" in this sense may or may not correspond to physical system queues. They are being used merely as recording devices. The following example illustrates this.

Example 11.1: Two types of vehicle arrve at a service facility and queue for service. Service takes time Y. Statistical information on the time spent queueing is required separately for each type and also for all vehicles.

Two GENERATE statements will be used one for each vehicle type (Blocks 1 and 4 below). Those generated by Block 1 will be called TYPE 1 and this number (1) is recorded in a PARAMETER called TYPE by Block 2. Similarly those generated by Block 4 will have the PARAMETER called TYPE set equal to 2. Those generated at Block 1 will join a QUEUE called Q(1) and those generated by Block 4 will join Q(2). Then all join Q(3) at Block 7.

```
BLOCK
NUMBER

  1        GENERATE . . . .    @                         a type 1 vehicle
  2        ASSIGN TYPE,1       @  This puts the PARAMETER called TYPE,=1
  3        INQUEUE Q(1),ARRIVAL.TIME  GO TO (L1)

  4        GENERATE . . . .    @                         a type 2 vehicle
  5        ASSIGN TYPE,2       @  This puts the PARAMETER called TYPE,=2
  6        INQUEUE Q(2),ARRIVAL.TIME

  7   L1   QUEUE Q(3)
  8        SEIZE FACILITY.X    @  FACILITY.X is the name assigned to
                                  the service facility
  9        OUTQUEUE Q(P$TYPE),ARRIVAL.TIME    @ TYPE will be 1 or 2,
                                                see blocks 2,5
 10        ADVANCE TIME(Y)     @  Y is the service time
 11        RELEASE FACILITY.X  @  Service is now complete
 12        TERMINATE,R         @  Vehicle leaves
```

An ORDER statement specifying that the entity QUEUE (mnemonic Q) will be indexed and will have dimension 3 must appear among the data definition statements.

Immediately a TRANSACTION seizes the FACILITY (block 8) it leaves Q(3). When it enters block 9 (after zero time in block 8) it leaves Q(1) or Q(2). It may be considered that Q(3) corresponds to the physical queue, containing both types of vehicle but Q(1) and Q(2) are used simply to provide data analysed separately for the two vehicle types. Note that for Q(3) a QUEUE block suffices. The INQUEUE, OUTQUEUE are only required if there is some block processing required while the TRANSACTION is on the QUEUE.

Distinction between QUEUE and INQUEUE

Consider the blocks

```
    1    QUEUE     Q1
    and
    2    INQUEUE   Q1,T1
```

when a TRANSACTION enters block 1 above it enters the QUEUE called "Q1". When it leaves the block it leaves "Q1". When a TRANSACTION enters block 2 above it enters the QUEUE called "Q1" but when it leaves the block it does not leave "Q1". It will not leave "Q1" until it reaches the corresponding OUTQUEUE block

```
              OUTQUEUE    Q1,T1.
```

The PARAMETER called T1 is used to store the arrival time onto Q1 so that when the TRANSACTION reaches the OUTQUEUE block it has a record in T1 of the time at which it arrived on Q1. It is not necessary to assign this value to T1, it is done automatically by block 2 above. Thus for every INQUEUE block there must be a corresponding OUTQUEUE block or the TRANSACTION would never leave the QUEUE. For complete details on QUEUE, INQUEUE, OUTQUEUE blocks see the GPSS mini-manual.

Example 11.2: Rewrite the previous program so that the time recorded on the various QUEUES includes the service time

```
1          GENERATE . . .                                  @  a type 1 vehicle
2          ASSIGN TYPE,1
3          INQUEUE Q(1),ARRIVAL.TIME     GO TO (L1)
4          GENERATE . . .                                  @  a type 2 vehicle
5          ASSIGN TYPE,2
6          INQUEUE Q(2),ARRIVAL.TIME
7    L1    INQUEUE Q(3),ARRIVAL.TIME
8          HOLD FACILITY.X     TIME(Y)
9          OUTQUEUE Q(3)
10         OUTQUEUE Q(P$TYPE)
11         TERMINATE,R
```

Note that here Q(3) requires an INQUEUE block (block 7). That is because the TRANSACTION should not leave the QUEUE at the time of entry to block 8 but at the time of departure from block 8. In this example a HOLD block suffices for FACILTIY.X for the same reason as that for Q(3) in example 11.1, that it to say, no block processing is required between entry to and exit from the FACILITY. The single block 8 is exactly the same as the following three blocks

```
    SEIZE FACILITY.X
    ADVANCE TIME(Y)
    RELEASE FACILITY.X
```

The equivalence of course only holds if no blocks intervene between SEIZE and RELEASE other than ADVANCE TIME.

11.2.5 Modelling Traffic Rules

1. A vehicle cannot overtake another vehicle going in the same direction at any stage.
2. If a container is moving either up or down within the building then no other container will be allowed to move until the first one has reached its destination. This rule is very restrictive and is relaxed in later models.
3. A vehicle cannot move from floor x to floor y(y = x ± 1) if another vehicle is moving from floor x to floor y or is occupying the apron on floor y i.e. for a vehicle to commence moving from floor x to floor y both the ramp from x to y and the apron on y must be free. This is to ensure that a vehicle will never have to stop on a ramp.
4. Only one vehicle can occupy the apron on any floor at one time.

Adherence to the above rules can be achieved as follows:

If each unidirectional ramp is represented by a FACILITY then 1 will be obeyed and 3 partially i.e. with respect to ramps.

If each apron is represented by a FACILITY then rule 4 is obeyed.

According to 3 a vehicle cannot ascend to the next floor unless the apron there is free. In GPSS this means that it should not SEIZE the FACILITY representing the ramp until it has first successfully SEIZED the FACILITY representing the apron. The same applies to a descending vehicle.

To force adherence to rule 2 a SAVEX called CONTAINER is used as a "flag" to indicate if movement is possible. This SAVEX initially has a value zero. When a container starts to move CONTAINER is set to 1. When the container reaches its destination CONTAINER is re-set to zero. A container cannot commence moving unless CONTAINER is zero.

11.2.6 Modelling Stochastic Elements

11.2.6.1 Arrival Rate

It is assumed that arrivals are random. The simulator defined negative exponential distribution is used to generate random arrivals. The data definition statement used was:
ARRIVAL FUNCTION,EXP RF$1,1,1. Thus the mean time returned was 1. This was then multiplied by the required average inter-arrival time which could vary over the day and was obtained via a second FUNCTION called RATE. The actual GENERATE statement is

```
    GENERATE 1            TIME(FN$ARRIVAL*FN$RATE)
```

"ARRIVAL" will return a random time with mean 1. This is then multiplied by an appropriate value returned by "RATE" which will depend on the time of day as specified in the FUNCTION definition statement for RATE.

In the first model arrivals were distributed according to Table 11.1 on the next page. Note that times in the program are based on a 30-second unit, 30 seconds being the highest common denominator of all times used.

It was assumed, based on estimates of usage, that at peak times the arrival rate would be about 60/hour and off-peak about 40/hour. Peaks were to be simulated at 9.30 and 14.30 assuming an opening time of 8.00 and closing at 17.30. Troughs were to occur at opening time, closing time and mid-way between peaks with a linear increase/decrease in the rate between troughs and peaks.

TABLE 11.1

1	2	3	4
TIME OF DAY	AVERAGE INTER-ARRIVAL TIME (minutes)	TIME OF DAY IN SIMULATOR UNITS (30 secs.) (T)	AVERAGE INTER-ARRIVAL TIME IN SIMULATOR UNITS (RATE)
08.00	1.5	960	3
09.30	1.0	1140	2
12.00	1.5	1440	3
14.30	1.0	1740	2
17.30	1.5	2100	3

The required function definition for RATE is

```
RATE    FUNCTION,C    V$T,960,3    1140,2    1440,3    1740,2    2100,3
```

The variable called T will contain the time within the day in 30-second units when reference is made to the function RATE i.e. T is the independent variable (column 3 above). If T is between two table values the simulator will linearly interpolate between the two corresponding "RATE" values since a continuous function is used (FUNCTION,C)

11.2.6.2 Assigning Vehicle Type

The GENERATE statement generates TRANSACTIONS (vehicles) which are indistinguishable. The first thing to be done is to assign an identity to the vehicle. Vehicles need to be distinguished by type and by destination (floor). This is done by using FUNCTIONS with the appropriate probability distributions.

Example 11.3

Vehicles on arrival are to be assigned a type number which should be either 1 or 2 (1 = container, 2 = lorry). Twenty-five percent should be type 1 and seventy-five percent type 2. Each vehicle is to be assigned a destination which is a number between 1 and 8 inclusive (the floor number). Equal numbers (on average) are to be assigned to each floor. Assuming that all vehicles are generated by the same GENERATE statement write the appropriate FUNCTION definitions and ASSIGN statements.

The probability distribution (in cumulative form) for "type" is as in Table 11.2

TABLE 11.2

TYPE	Cumulative Probability
1 = container	0.25
2 = lorry	1.00

The required FUNCTION definition using random number generator number 3 is

```
TYPE.ASSIGN   FUNCTION   RF$3,.25,1   1,2
```

There is no particular reason for choosing random number generator number 3 as all are equivalent. If there are several FUNCTION definitions using different generators for each can make manual checking of the simulation easier. Thus RF$1 can be used for the arrival-rate FUNCTION, RF$2 for the floor-assignment FUNCTION, RF$3 for the type-assignment FUNCTION etc. Different student groups could and indeed should use different generators so that their results are independent. Alternatively they can use SEED statements as described in Unit 8.

11.2.6.3 Assigning Destination

There are 8 possible destination floors. If all are equally likely the distribution is as follows:

TABLE 11.3

Floor	Cumulative Probability
1	.125
2	.250
3	.375
4	.500
5	.625
6	.750
7	.875
8	1.000

The FUNCTION definition statement is as follows:

```
FLOOR.ASSIGN   FUNCTION   RF$2,.125,1   .250,2   .375,3   .5,4
     .625,5   .75,6   .875,7   1,8
```

The actual assignment of type and destination is done by two ASSIGN statements following the GENERATE statement viz.,

```
    ASSIGN    TYPE,FN$TYPE.ASSIGN      @    This puts TYPE = 1 or 2
    ASSIGN    FLOOR,FN$FLOOR.ASSIGN    @    This puts FLOOR = 1,2...8
```

11.2.7. <u>Allocation of Queues to Collect Data as Specified in 11.2.4</u>

On arrival a TRANSACTION (vehicle) is entered onto several QUEUES for the purpose of data collection and analysis. An ORDER statement is used to indicate that QUEUES will be referenced by indexing and specifying the number of queues that will be required. The statement is

```
ORDER,Q  84
```

These queues are allocated as follows:

$Q(1 \leq i \leq 9)$	= queue for ramp from floor n to floor n+1 (i=n+1)
Q(10)	= queue of containers held up because another container is moving
$Q(11 \leq i \leq 19)$	= queue for ramp from floor n to floor n-1 (i=20-n)
Q(20)	= queue for check-in
$Q(21 \leq i \leq 29)$	= queue for apron prior to docking on floor n (i=n+20)
Q(30)	= queue on roof for space at destination
$Q(31 \leq i \leq 38)$	= queue for apron on departure from floor n after docking (i=n+30)

The above QUEUES Q(1 - 38) are actual queues. Subsequent ones (39-85) are for recording the total time spent in the system under different headings depending on time of arrival and destination floor as follows:

A	Q(39)	all vehicles
B(1)	$Q(41 \leq i \leq 50)$	containers where i = hour of arrival +40
B(2)	$Q(51 \leq i \leq 60)$	lorries where i = hour of arrival +50
C(1)	$Q(61 \leq i \leq 68)$	containers where i = floor +60
C(2)	$Q(69 \leq i \leq 76)$	lorries where i = floor + 68
D	$Q(77 \leq i \leq 84)$	all vehicles where i = floor +76

Example 11.4:

A container arrives at time 09.15 (day started at 08.00) and is proceeding to floor 5. On which QUEUES should its record be entered for recording the total time spent in the system?

Solution:

See the definition of QUEUES Q(39-55) on the previous page. Note that since actual arrival time = 09.15 the hour of arrival, counting the first hour (08.00) as one, is hour two. Thus we have the vehicle profile as follows

Type = Container (type number 1)

Hour = 2

Floor = 5

A All vehicles must be entered on Q(39)

B Containers must be entered on a QUEUE in the range Q(41) - Q(49) depending on i where i = hour + 40 = 42 hence Q(42).

C Containers must also be entered on a QUEUE in the range Q(60) - Q(67) depending on i where i = floor + 59 = 64 hence Q(64).

D All vehicles must be entered on a QUEUE in the range Q(78) - Q(85) depending on i where i = floor + 77 = 82 hence Q(82).

Thus the QUEUES on which it should be entered immediately on arrival are Q(39), Q(42), Q(64), Q(82). These can be considered simply as files for recording information on vehicles with a particular profile.

For the program formulae must be written for obtaining these numbers from information on arrival time, vehicle type and destination floor.

Given that the type (1 = container 2 = lorry) is stored in parameter TYPE and the floor in parameter FLOOR the required QUEUE numbers can generally be given as

A. 39

B. (V$T-X$DAY.START)/120+P$TYPE*10+31

C. P$FLOOR+P$TYPE*9+50

D. P$FLOOR+77

where V$T returns time within the day (in 30 sec. units) at arrival.

X$DAY.START returns the start of the day

In the above example	V$T =	1110	i.e. 09.15
	X$DAY.START =	960	i.e. 08.00
	P$TYPE =	1	
	P$FLOOR =	5	

So Formula B gives (1110 + 960)/120 + 1(10) + 31 = 42

Formula C gives 5 + 1(9) + 50 = 64

Formula D gives 5 + 77 = 82

The values returned by formulae B and C should be stored in a PARAMETER to avoid having to re-calculate them on departure.

11.2.8 The Program

11.2.8.1 Effect of Restriction on Container Movement

If the program is written on the basis of the assumptions in this unit it will be evident that queue times are intolerably long even though utilisation of bays is small. Students should be required to discover why this is happening. The reason (this should not be revealed to students) is that the restriction that no container may enter or leave the building while another container is moving either in or out of the building is causing large queues to build up at the checkin.

In the version of the program which follows the restriction on container movement has been removed. This has been done by putting asterisks in front of the relevant instructions. This converts them to comments which are not compiled as program blocks. The asterisks can be removed if one wants to make the blocks operative. The relevant instructions are those between blocks 74 and 75, between 85 and 86 between 92 and 93, between 107 and 108, between 111 and 112 and between 125 and 126. It can be seen that queue times are quite reasonable. Further investigation could be carried out to discover why so many vehicles must go to the roof.

11.2.8.2 <u>REPORT Section</u>

A REPORT section has been included. This produces an edited version of the standard statistical output. The first statement must be REPORT and the last statement ENDREPORT. The OUTPUT statement lists the items to be included using the mnemonics listed in Table 11.4. The SECTION statement indicates which rows and columns of the standard output are required, e.g. the statement

```
SECTION   F(ALL)   F1/F3/F4
```

specifies that all rows of the FACILITY detail statistics are to be printed but only columns 1,3 and 4.

The statement EJECT causes the printer to go to a new page. The statement TEXT 'ABCD...' causes the printer to print the message or title between the quotation marks. The statement SPACE causes the printer to skip a line. The statement SECTION,MX relates specifically to SAVEX MATRICES. It lists the SAVEX MATRICES which are to be included and can cause suppression of the standard title which is useful if one wants to provide ones own title or column headings as in this case. The statement

```
SECTION,MX     DR     TITLE(SUPPRESS)
```

indicates that only one SAVEX MATRIX viz. DR is required and that the title should be suppressed.

Only the standard statistical output sections which have names and abbreviated names shown in Table 11.4 may be used in the SECTION statement.

TABLE 11.4

NAME	ABBREVIATED NAME
Clock Times and Termination Counts	C
Block TRANSACTION Counts	B
SAVEX Values	X
FACILITY Statistics	F
STORAGE Statistics	S
QUEUE Statistics	Q
User Chain Statistics	UC
Group Summary Statistics	G
TABLE and QTABLE Summary Statistics	T
Random Number Generators	RN

```
*
*
*
*
*
*
*
*
*
*
*
*
*     SET UP AND  REPORT SECTION
*
*
*  FIRST MODEL WITH SIMPLE ASSUMPTIONS (UNIT 9)
*
JOB
*
*
*
*
REPORT
*
OUTPUT C,B,X,MX,F,S,Q,T,TD
*
SECTION F(ALL) F1/F3/F4
*
SECTION S(ALL) S1/S2/S3/S4/S7/S8/S10/S11
*
SECTION T(ALL) T1/T2/T3/T4/T5
*
EJECT
*
TEXT '       CHECK-IN           NUMBER    OF     VEHICLES
+                   NUMBER WITH       UTILISATIONS      BUILDING'
*
SPACE
*
TEXT '       CLOSING TIME  CONTAINERS   LORRIES         TOTAL   AVERAGE T
+IME   MAXIMUM TIME   LONG DELAY     CHECK-IN    BAYS  CLOSING TIME'
*
SECTION,MX DR TITLE(SUPPRESS)
*
ENDREPORT
*
*
*
*
*
*
*
*
*
*
*
*
*
*
*
```

```
*
*
*                                                           PAGE....2
*
*                     ORDER  STATEMENTS
*
ORDER,Q 84
*
*
* THE FOLLOWING ORDER STATEMENTS ARE TO GIVE NAMES TO ALL QUEUES FOR
*
*  EASE OF INTERPRETATION OF OUTPUT, TWO STATEMENTS ARE REQUIRED
*
*  AS ONE WOULD BE TOO LONG.  THE FIRST NAMES QUEUES 1-46,THE SECOND
*
*  NAMES QUEUES 47-84
*
*
ORDER,Q RAMP.UPTO.01,RAMP.UPTO.02,RAMP.UPTO.03,RAMP.UPTO.04,RAMP.UPTO.05
+,RAMP.UPTO.6,RAMP.UPTO.07,RAMP.UPTO.08,RAMP.UPTO.09,BLANK,RAMP.DOWN.08,
+RAMP.DOWN.07,RAMP.DOWN.06,RAMP.DOWN.05,RAMP.DOWN.04,RAMP.DOWN.03,RAMP.D
+OWN.02,RAMP.DOWN.01,RAMP.DOWN.00,CHECK.IN,UP.APRON.01,UP.APRON.02,UP.AP
+RON.03,UP.APRON.04,UP.APRON.05,UP.APRON.06,UP.APRON.07,UP.APRON.08,ROOF
+..APRON,ROOF.QUEUE,DOWN.APRON.1,DOWN.APRON.2,DOWN.APRON.3,DOWN.APRON.4,
+DOWN.APRON.5,DOWN.APRON.6,DOWN.APRON.7,DOWN.APRON.8,ALL.VEHICLES,EMPTY,
+CONT.HOUR.1,CONT.HOUR.2,CONT.HOUR.3,CONT.HOUR.4,CONT.HOUR.5,CONT.HOUR.6
*
*
ORDER,Q 46,CONT.HOUR.7,CONT.HOUR.8,CONT.HOUR.9,CONT.HOUR.10,LORR.HOUR.1,
+LORR.HOUR.2,LORR.HOUR.3,LORR.HOUR.4,LORR.HOUR.5,LORR.HOUR.6,LORR.HOUR.7
+,LORR.HOUR.8,LORR.HOUR.9,LORR.HOUR.10,CONT.FLOOR.1,CONT.FLOOR.2,CONT.FL
+OOR.3,CONT.FLOOR.4,CONT.FLOOR.5,CONT.FLOOR.6,CONT.FLOOR.7,CONT.FLOOR.8,
+LORR.FLOOR.1,LORR.FLOOR.2,LORR.FLOOR.3,LORR.FLOOR.4,LORR.FLOOR.5,LORR.F
+LOOR.6,LORR.FLOOR.7,LORR.FLOOR.8,ALL.FLOOR.1,ALL.FLOOR.2,ALL.FLOOR.3,AL
+L.FLOOR.4,ALL.FLOOR.5,ALL.FLOOR.6,ALL.FLOOR.7,ALL.FLOOR.8
*
*
*
ORDER,S 8,ROOF
*
ORDER,S 9
*
ORDER,F 29
*
ORDER,X 4,CHECKINTIME,MANOEUVRE,DAY.START,OPEN.LENGTH,LONG.DELAY
*
ORDER,X 9
*
*
* THE LAST TWO ORDER STATEMENTS MAKE THE FOLLOWING EQUIVALENCES
*    X(5)=CHECKINTIME    X(6)=MANOEUVRE TIME
*    X(7)=DAY START      X(8)=OPEN.LENGTH
*    X(9)=WHAT IS CONSIDERED A LONG DELAY
*
*    X(1-2) SPECIFY LOAD TIMES FOR 2 VEHICLE TYPES
*    X(3-4) ARE USED IN THE PROGRAMME AS WORKING STORES
*
*      THE VALUES FOR X(I) ARE SPECIFIED ON PAGE 3
*
*
```

```
*
*
*
*
*
*
*
*
*
*
*
*
*
*    DEFINITIONS,  CAPACITIES  AND  INITIALISATIONS
*
*
*    ALL TIMES ARE IN UNITS OF 30 SECONDS
*
* TABLE AND MATRIX DEFINITIONS
  SYSTEM.TIME TABLE M$1,60,10,30
  MATRIX DR(51,10)
*
*
*
*
*
*  VARIABLE DEFINITIONS
T       VARIABLE C$1//2880              @ TIME WITHIN THE DAY
V1      VARIABLE,I V$T/120              @ HOUR PART OF THE TIME
V2      VARIABLE V$V1*40+V$T/2          @ THE TIME IN 24 HOUR FORMAT
I       VARIABLE 19-P$POSITION
*
*
*
*
* FUNCTION DEFINITIONS
TYPE.ASSIGN  FUNCTION RF$3,.25,1 1,2
DRIVE.TIME   FUNCTION RF$4,.8,1 1,2
RATE         FUNCTION,C V$T,960,3 1140,2 1440,3 1740,2 2100,3
ARRIVAL      FUNCTION,EXP RF$1,1,1
FLOOR.ASSIGN FUNCTION RF$2,.125,1 .25,2 .375,3 .5,4 .625,5 .75,6 .875,7
+1,8
*
*
*
* CAPACITY STATEMENTS
S(1) CAPACITY 8
S(2) CAPACITY 8
S(3) CAPACITY 8
S(4) CAPACITY 8
S(5) CAPACITY 8
S(6) CAPACITY 8
S(7) CAPACITY 3
S(8) CAPACITY 8
S(9) CAPACITY 200
BUILDING CAPACITY 200
*
*
* SET UP PROGRAMME PARAMETERS AS DEFINED ABOVE FOR X(1-9)
INITIAL X(1-2),120,60/X(5-9),1,2,960,1140,240
*
*
```

```
   *
   *
   *
   *
   *
   *                                                              PAGE....4
   *
   *          SEGMENT TO COMPILE DAILY REPORT
   *
   *
   *
 1        GENERATE    0     TIME(2880)            @ START DAY
   *
 2        ADVANCE             TIME(X$DAY.START)   @    ADVANCE TO OPENING TIME
 3        SAVEX    OPEN,1                         @ START ARRIVALS
 4        ADVANCE             TIME(X$OPEN.LENGTH) @ ADVANCE TO CLOSING TIME
 5        SAVEX    OPEN,0                         @ CLOSE CHECK-IN
 6        GATE     SE,BUILDING                    @ WAIT FOR BUILDING TO EMPTY
   *
   *
   *       ADJUST TIMES
   *
 7        SAVEX    CHECK.CLOSE,X$CHECK.CLOSE//2880  @ TIME WITHIN DAY
 8        SAVEX    DAYL,V$T-X$DAY.START    GO TO(L11,L12)  @ DAY LENGTH
 9 L11    COMPARE X$DAYL LE 0          @ ADJUST FOR CLOSING AFTER MIDNIGHT
10        SAVEX    DAYL,X$DAYL+2880         GO TO(L20,L12)
11 L20    COMPARE X$CHECK.CLOSE LT X$OPEN.LENGTH+X$DAY.START
12        SAVEX    CHECK.CLOSE,X$CHECK.CLOSE+2880 @ADJUST TIME AFTER MIDNIGHT
   *
13 L12    SAVEX NUMBER3,X$X(3)+X$X(4)             @ TOTAL NUMBER OF VEHICLES
14        SAVEX DAY,X$DAY+1                       @  DAY COUNTER
   *
   *
   *
   *
   * ENTER DATA IN ROW CORRESPONDING TO THE CURRENT DAY
   *
   *
15       MSAVEX DR(X$DAY,1),(X$CHECK.CLOSE*2.5-X$CHECK.CLOSE//120)/3
16       MSAVEX DR(X$DAY,2),X$X(3)
17       MSAVEX DR(X$DAY,3),X$X(4)
18       MSAVEX DR(X$DAY,4),X$NUMBER3
19       MSAVEX DR(X$DAY,5),X$TT/X$NUMBER3+.5
20       MSAVEX DR(X$DAY,6),X$MAX
21       MSAVEX DR(X$DAY,7),X$N
         MSAVEX DR(X$DAY,8),X$NUMBER3*100*X$CHECKINTIME/(X$CHECK.CLOSE-X$DAY
22 +.START)+.5
23        MSAVEX DR(X$DAY,9),(X$X(3)*X$X(1)+X$X(4)*X$X(2))/(X$DAYL*.64)+.5
24        SAVEX   XY,X$DAYL+X$DAY.START
25        MSAVEX DR(X$DAY,10),(X$XY*2.5-X$XY//120)/3
   *
   *
   *
   *
   *
   *
   *
   *
   *
   *
```

```
      *
      *
      *
      *
      *
      *
      *
      *                                                                    PAGE....5
      * UPDATE THE SUMMARY ROW
      *
 26         ADVANCE           GO TO(L41,L42)
      *
      *
      *
      *   SAVEXES DR5,DR8,DR9 BELOW ACCUMULATE TOTALS FOR SUBSEQUENT AVERAGING
      *
      *
 27 L41     COMPARE MX$DR(X$DAY,1) GT MX$DR(51,1)
 28         MSAVEX DR(51,1),MX$DR(X$DAY,1)
 29 L42     MSAVEX DR(51,2),MX$DR(51,2)+X$X(3)
 30         MSAVEX DR(51,3),MX$DR(51,3)+X$X(4)
 31         MSAVEX DR(51,4),MX$DR(51,4)+X$NUMBER3
 32         SAVEX DR5,X$DR5+X$TT/X$NUMBER3        GO TO(L43,L44)
 33 L43     COMPARE MX$DR(X$DAY,6) GT MX$DR(51,6)
 34         MSAVEX DR(51,6),MX$DR(X$DAY,6)
 35 L44     MSAVEX DR(51,7),MX$DR(51,7)+X$N
 36         SAVEX  DR8,X$DR8+MX$DR(X$DAY,8)
 37         SAVEX  DR9,X$DR9+MX$DR(X$DAY,9)        GO TO(L26,L28)
 38 L26     COMPARE MX$DR(X$DAY,10) GT MX$DR(51,10)
 39         MSAVEX  DR(51,10),MX$DR(X$DAY,10)
      *
      *
      *
      *
      *   ZERO COUNTERS FOR THE NEXT DAY
      *
      *
 40 L28     SAVEX TT,0
 41         SAVEX MAX,0
 42         SAVEX N,0
 43         SAVEX X(3),0
 44         SAVEX X(4),0
      *
      *
      *    COMPUTE AVERAGES
      *
 45         MSAVEX DR(051,5),X$DR5/X$DAY+.5
 46         MSAVEX DR(051,8),X$DR8/X$DAY+.5
 47         MSAVEX DR(051,9),X$DR9/X$DAY+.5          GO TO(L27,L29)
      *
      *
 48 L27     COMPARE X$DAY GE 50             @ END SIMULATION AND PRINT RESULTS
 49         STOP
 50 L29     TERMINATE
      *
      *
      *
      *
      *
      *
```

```
*
*
*
*
*
*
*
*
*     GENERATE VEHICLES, SET UP PROFILES, ENTER QUEUES AND CHECK IN
*
*
*
     GENERATE 1 TIME(FN$ARRIVAL*FN$RATE)
     COMPARE X$OPEN GT 0                    @ IS THE CHECK-IN OPEN?
*
*
*
* SET UP VEHICLE PROFILE IN ITS PARAMETERS
*
     MARK                                   @ ARRIVAL TIME FOR PRINTOUT
     ASSIGN AT,V$V2                         @ CURRENT FLOOR POSITION
     ASSIGN POSITION,1                      @ DESTINATION FLOOR
     ASSIGN FLOOR,FN$FLOOR.ASSIGN           @ TYPE I.E. LORRY OR CONTAIN
     ASSIGN TYPE,FN$TYPE.ASSIGN
     ASSIGN FLOOR.COPY,P$FLOOR
     ASSIGN WT,P$FLOOR+P$TYPE*8+52          @ TYPE,FLOOR INDEX
     ASSIGN TQ,(V$T-X$DAY.START)/120+31+P$TYPE*10     @ TYPE,TIME INDEX
     SAVEX  X(P$TYPE+2),X$X(P$TYPE+2)+1
*
*
*  JOIN VARIOUS QUEUES FOR DATA COLLECTION PURPOSES ACCORDING TO TYPE
*
     INQUEUE Q(P$TQ),ARRIVAL.TIME
     INQUEUE Q(P$FLOOR+76),ARRIVAL.TIME
     INQUEUE Q(P$WT),ARRIVAL.TIME
     INQUEUE Q(39),ARRIVAL.TIME
     QUEUE   Q(20)
*
*
*
*                  ENTER BUILDING AND CHECK IN
*
     ENTER   BUILDING
     SEIZE   CHECKIN
     ADVANCE         TIME(X$CHECKINTIME) GO TO(L1,L2)
L1   ENTER   S(P$FLOOR) GO TO(L6)           @RESERVE SPACE IF AVAILABLE
*
* GO TO THE ROOF IF FLOOR SPACE NOT AVAILABLE
*
L2   ENTER   ROOF                           @RESERVE ROOF SPACE
     ASSIGN  FLOOR,9                        @CHANGE DESTINATION TO ROOF
*
*
*
*
*
*
*
*
*
```

```
    *
    *
    *
    *
    *
    *
    *
    *
    *         DRIVE UP TO DESTINATION FLOOR OR TO THE ROOF
    *
    *  PREPARE TO DRIVE UP RAMP NUMBER 1
    *
73 L6     QUEUE Q(1)                          @JOIN QUEUE FOR RAMP
74        SEIZE F(1)                          @ OCCUPY RAMP
    *
    *
    *
    *
    * CONTAINER TEST
    *   ADVANCE          GO TO(L3,L15)
    *L3  COMPARE P$TYPE EQ 1                  @ IS THIS A CONTAINER  IF SO WAIT
    *    QUEUE   Q(10)                        @ Q FOR CLEARANCE FOR CONTAINER
    *    COMPARE X$CONTAINER NE 1             @ UNTIL CONTAINER FLAG IS OFF
    *    SAVEX   CONTAINER,1                  @ SET CONTAINER MOVING FLAG
    *
    *
75 L15    RELEASE CHECKIN
76        SAVEX   CHECK.CLOSE,C$1             @ STORE CLOSING TIME
    *
    *
    *
    * DRIVE UP RAMPS TO ASSIGNED FLOOR OR ROOF
    *
77 L9     ADVANCE TIME(FN$DRIVE.TIME) GO TO(L4,L10)    @ DRIVE UP
78 L10    ASSIGN POSITION,P$POSITION+1                 @ UPDATE POSITION
79        QUEUE Q(P$POSITION)                     @  JOIN QUEUE FOR NEXT RAMP
80        SEIZE F(P$POSITION)                     @  OCCUPY NEXT RAMP
81        RELEASE F(P$POSITION-1) GO TO(L9)       @  RELEASE PREVIOUS RAMP
82 L4     COMPARE P$POSITION EQ P$FLOOR           @  IS THIS FINAL DESTINATION
    *
    *
    *
    * LEAVE THE RAMP AND ENTER THE FORECOURT ON THE ASSIGNED FLOOR OR ROOF
    *
83        QUEUE   Q(P$FLOOR+20)                  @ QUEUE FOR FORECOURT
84        SEIZE   F(P$FLOOR+20)                  @ OCCUPY FORECOURT
85        RELEASE F(P$FLOOR)                     @ RELEASE RAMP
    *
    *
    *
    *  CONTAINER TEST
    * IF THIS WAS A CONTAINER (TYPE=1) SWITCH OFF FLAG
    *    SAVEX   CONTAINER,(P$TYPE-1)*X$CONTAINER
    *
    *
86        ADVANCE           GO TO(L13,L14)
    *
    *
    *
    *
    *
```

```
     *
     *
     *
     *
     *
     * PARK ON ROOF AND WAIT FOR FLOOR TO BECOME AVAILABLE
     *
 87 L13   COMPARE P$FLOOR EQ 9                   @ IS THIS THE ROOF ?
 88       ADVANCE TIME(X$MANOEUVRE)              @ MANOEUVRE INTO PARK POSITION
 89       RELEASE F(29)                          @ RELEASE THE FORECOURT
 90       ASSIGN  FLOOR,P$FLOOR.COPY             @ RESTORE DESTINATION IN FLOOR
 91       QUEUE   Q(30)                          @ JOIN QUEUE ON ROOF
 92       ENTER   S(P$FLOOR)                     @ RESERVE FLOOR WHEN AVAILABLE
     *
     *
     *
     *  CONTAINER TEST
     *     ADVANCE           GO TO(L16,L17)
     *L16 COMPARE P$TYPE EQ 1                     @ IS THIS A CONTAINER ?
     *    QUEUE   Q(10)
     *    COMPARE X$CONTAINER NE 1               @ IF SO WAIT TILL FLAG OFF
     *    SAVEX   CONTAINER,1                    @ PUT FLAG ON AND GO
     *
     *
     *
 93 L17   LEAVE   ROOF                           @ LEAVE THE ROOF
 94       QUEUE   Q(29)                          @ QUEUE FOR FORECOURT
 95       SEIZE   F(29)                          @ OCCUPY FORECOURT
 96       ADVANCE         TIME(X$MANOEUVRE)      @ MANOEUVRE TO LEAVE
 97       QUEUE   Q(11)                          @ QUEUE FOR RAMP DOWN
 98       SEIZE   F(11)                          @ OCCUPY RAMP DOWN TO FLOOR 8
 99       RELEASE F(29)                          @ RELEASE FORECOURT
     *
     *
     *
     * DRIVE DOWN TO ASSIGNED FLOOR
100 L5    ASSIGN  POSITION,P$POSITION-1          @ UPDATE POSITION
101       ADVANCE         TIME(FN$DRIVE.TIME) GO TO(L18,L19)  @ DRIVE DOWN
102 L19   QUEUE   Q(V$I+1)                       @ QUEUE FOR RAMP DOWN
103       SEIZE   F(V$I+1)                       @ OCCUPY RAMP
104       RELEASE F(V$I)  GO TO(L5)              @ RELEASE PREVIOUS RAMP
     *
105 L18   COMPARE P$FLOOR EQ P$POSITION          @ IS THIS THE FLOOR
106       QUEUE   Q(P$FLOOR+20)                  @ QUEUE FOR FORECOURT
107       SEIZE   F(P$FLOOR+20)                  @ OCCUPY FORECOURT
     *
     *  CONTAINER TEST
     *    SAVEX   CONTAINER,(P$TYPE-1)*X$CONTAINER @ IF CONTAINER,FLAG OFF
     *
108       RELEASE F(V$I)                         @ RELEASE RAMP
     *
     *
     *
     *
     *
     *
     *
     *
     *
     *
```

```
      *
      *
      *
      *
      *
      *
      *
      *
      *            UNLOAD,  DRIVE DOWN AND LEAVE
      *
      * PROCEED TO BAY AND UNLOAD
109 L14  ADVANCE  TIME(X$MANOEUVRE)                 @ MANOEUVRE INTO BAY
110      RELEASE  F(P$FLOOR+20)                     @ RELEASE FORECOURT
111      ADVANCE  TIME(X$X(P$TYPE))                 @ UNLOAD
      *
      *  CONTAINER TEST
      *   ADVANCE          GO TO(L7,L8)
      *L7  COMPARE  P$TYPE EQ 1                     @ IS THIS A CONTAINER
      *    QUEUE    Q(10)
      *    COMPARE  X$CONTAINER NE 1                @ WAIT TILL FLAG IS OFF
      *    SAVEX    CONTAINER,1                     @ SET FLAG ON
      *
112 L8   QUEUE    Q(P$FLOOR+30)                     @ QUEUE FOR FORECOURT
113      SEIZE    F(P$FLOOR+20)                     @ OCCUPY  FORECOURT
114      LEAVE    S(P$FLOOR)                        @ LEAVE BAY
115      ADVANCE  TIME(X$MANOEUVRE)                 @ MANOEUVRE ON FORECOURT
116      QUEUE    Q(V$I+1)                          @ QUEUE FOR RAMP DOWN
117      SEIZE    F(V$I+1)                          @ OCCUPY RAMP DOWN
118      RELEASE  F(P$FLOOR+20)                     @ RELEASE FORECOURT
      *
      * DRIVE DOWN TO GROUND
119 L25  ADVANCE  TIME(FN$DRIVE.TIME) GOTO(L22,L23)@   DRIVE DOWN
120 L22  COMPARE  P$POSITION NE 1                   @ IS THERE FURTHER TO GO
121      ASSIGN   POSITION,P$POSITION-1             @ UPDATE POSITION
122      QUEUE    Q(V$I+1)                          @ QUEUE FOR RAMP DOWN
123      SEIZE    F(V$I+1)                          @ OCCUPY RAMP
124      RELEASE  F(V$I) GO TO(L25)                 @ RELEASE PREVIOUS RAMP
125 L23  ADVANCE
      *
      *   CONTAINER TEST
      *     SAVEX    CONTAINER,(P$TYPE-1)*X$CONTAINER @ IF CONTAINER,FLAG OFF
126         RELEASE F(19)                            @ RELEASE RAMP TO GROUND
      *
      * LEAVE ALL QUEUES  AND LEAVE THE SYSTEM
127         OUTQUEUE Q(39),ARRIVAL.TIME
128         OUTQUEUE Q(P$FLOOR+76),ARRIVAL.TIME
129         OUTQUEUE Q(P$TQ),ARRIVAL.TIME
130         OUTQUEUE Q(P$WT),ARRIVAL.TIME
131         LEAVE BUILDING
      * SEGMENT TO UPDATE RECORDS
132         TABULATE SYSTEM.TIME  GO TO(L45,L46)
133 L45     COMPARE M$1 GT X$MAX
134         SAVEX    MAX,M$1    @ UPDATE RECORD OF MAXIMUM TIME
135 L46       SAVEX TT,X$TT+M$1 GO TO(L32,L33)
136 L32     COMPARE M$1 GT X$LONG.DELAY
137         SAVEX N,X$N+1   @ UPDATE RECORD OF NO. WITH LONG DELAYS
138 L33     TERMINATE
      *
            START 1
```

SYSTEM TIME

The table below shows the distribution of "system time" i.e. the total time that a vehicle spends in the godown. The total number of vehicles was 28094. Time is measured in 30 second units so that the mean time of 108.365 is 54 minutes (to the nearest minute).

```
 TABLE          NON-WEIGHTED    NON-WEIGHTED    NON-WEIGHTED   NON-WEIGHTED
 NAME          NO. OF ENTRIES SUM OF ARGUMENTS MEAN ARGUMENT   STD. DEV.
SYSTEM.TIME        28094        3044397.0000      108.365        34.826

TABLE NAME: SYSTEM.TIME

    UPPER         OBSERVED  PERCENT   CUMULATIVE    CUMULATIVE  MULTIPLE   DEVIATION
    LIMIT        FREQUENCY  OF TOTAL  PERCENTAGE    REMAINDER   OF MEAN    FROM MEAN
   60.0000             0       .00         .00        100.00      .554       -1.389
   70.0000          1321      4.70        4.70         95.30      .646       -1.102
   80.0000          5580     19.86       24.56         75.44      .738        -.814
   90.0000          5728     20.39       44.95         55.05      .831        -.527
  100.0000          3026     10.77       55.72         44.28      .923        -.240
  110.0000          1709      6.08       61.81         38.19     1.015         .047
  120.0000          1122      3.99       65.80         34.20     1.107         .334
  130.0000          1252      4.46       70.26         29.74     1.200         .621
  140.0000          2487      8.85       79.11         20.89     1.292         .908
  150.0000          2341      8.33       87.44         12.56     1.384        1.196
  160.0000          1259      4.48       91.92          8.08     1.476        1.483
  170.0000           761      2.71       94.63          5.37     1.569        1.770
  180.0000           519      1.85       96.48          3.52     1.661        2.057
  190.0000           351      1.25       97.73          2.27     1.753        2.344
  200.0000           221       .79       98.52          1.48     1.846        2.631
  210.0000           150       .53       99.05           .95     1.938        2.918
  220.0000            98       .35       99.40           .60     2.030        3.206
  230.0000            79       .28       99.68           .32     2.122        3.493
  240.0000            35       .12       99.80           .20     2.215        3.780
  250.0000            20       .07       99.88           .12     2.307        4.067
  260.0000            15       .05       99.93           .07     2.399        4.354
  270.0000             9       .03       99.96           .04     2.492        4.641
  280.0000             3       .01       99.97           .03     2.584        4.928
  290.0000             3       .01       99.98           .02     2.676        5.216
  300.0000             3       .01       99.99           .01     2.768        5.503
  310.0000             2       .01      100.00           .00     2.861        5.790
 REMAINING FREQUENCIES ARE ALL ZERO
```

The FACILITIES below are defined in the text on page 254.
The STORAGES represent the floors.

```
   FACILITY     NUMBER      AVERAGE
     NAME       ENTRIES   TIME/TRANS
F (   1)        28094          1.52
F (   2)        25128          1.48
F (   3)        22142          1.46
F (   4)        19191          1.45
F (   5)        16293          1.47
F (   6)        13477          1.51
F (   7)        10577          1.60
F (   8)         7763          1.82
F (   9)         4926          2.54
F (  10)            0           .00
F (  11)         4926          1.95
F (  12)         7763          1.85
F (  13)        10577          1.81
F (  14)        13477          1.76
F (  15)        16293          1.71
F (  16)        19191          1.64
F (  17)        22142          1.53
F (  18)        25128          1.39
F (  19)        28094          1.20
F (  20)            0           .00
F (  21)         7138          2.08
F (  22)         7088          2.17
F (  23)         7002          2.24
F (  24)         6956          2.28
F (  25)         6826          2.29
F (  26)         7014          2.28
F (  27)         6974          2.25
F (  28)         7190          2.24
F (  29)         9852          2.16
CHECKIN         28094          1.45
  STORAGE       MAXIMUM    AVERAGE   MAXIMUM    TOTAL   TOTAL    AVERAGE   CURRENT
    NAME        CONTENTS   CONTENTS  CAPACITY  ENTRIES  TRANS   TIME/ENT  CONTENTS
S (   1)              8       2.03        8     3569     3569     81.58      0
S (   2)              8       2.05        8     3544     3544     82.86      0
S (   3)              8       2.05        8     3501     3501     83.86      0
S (   4)              8       2.06        8     3478     3478     84.97      0
S (   5)              8       2.05        8     3413     3413     86.19      0
S (   6)              8       2.12        8     3507     3507     86.81      0
S (   7)              8       2.13        8     3487     3487     87.76      0
S (   8)              8       2.22        8     3595     3595     88.58      0
ROOF                 18       1.02      200     4926     4926     29.59      0
BUILDING            109      21.24      200    28094    28094    108.36      0
```

QUEUE Details

The QUEUES are defined in the text on page 267.
See comments on page 288.

QUEUE NAME	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	ZERO ENTRIES	ZEROS PERCENT	AV. TIME/ENT (ALL)	AV. TIME/ENT (NON ZERO)	CURRENT CONTENTS	TABLE NAME
RAMP.UPTO.01	1	.09	28094	17872	63.62	.45	1.24	0	
RAMP.UPTO.02	1	.06	25128	19022	75.70	.32	1.34	0	
RAMP.UPTO.03	1	.04	22142	17993	81.26	.27	1.45	0	
RAMP.UPTO.04	1	.03	19191	16276	84.81	.25	1.62	0	
RAMP.UPTO.05	1	.03	16293	14168	86.96	.24	1.81	0	
RAMP.UPTO.6	1	.02	13477	11855	87.96	.25	2.06	0	
RAMP.UPTO.07	1	.02	10577	9310	88.02	.29	2.46	0	
RAMP.UPTO.08	1	.02	7763	6633	85.44	.41	2.84	0	
RAMP.UPTO.09	1	.03	4926	3726	75.64	.77	3.15	0	
BLANK	0	.00	0	0	.00	.00	.00	0	
RAMP.DOWN.08	1	.01	4926	4547	92.31	.31	4.06	0	
RAMP.DOWN.07	2	.03	7763	6344	81.72	.62	3.39	0	
RAMP.DOWN.06	2	.04	10577	8194	77.47	.60	2.66	0	
RAMP.DOWN.05	2	.06	13477	9961	73.91	.59	2.25	0	
RAMP.DOWN.04	2	.06	16293	11489	70.51	.57	1.92	0	
RAMP.DOWN.03	2	.07	19191	13132	68.43	.53	1.68	0	
RAMP.DOWN.02	2	.07	22142	15485	69.93	.45	1.49	0	
RAMP.DOWN.01	2	.06	25128	18733	74.55	.33	1.31	0	
RAMP.DOWN.00	2	.04	28094	23409	83.32	.19	1.13	0	
CHECK.IN	1	.00	28094	28094	100.00	.00	.00	0	
UP.APRON.01	2	.01	3569	2855	79.99	.31	1.53	0	
UP.APRON.02	2	.01	3544	2860	80.70	.33	1.69	0	
UP.APRON.03	2	.01	3501	2823	80.63	.33	1.70	0	
UP.APRON.04	2	.01	3478	2800	80.51	.36	1.86	0	
UP.APRON.05	2	.01	3413	2741	80.31	.37	1.89	0	
UP.APRON.06	2	.01	3507	2776	79.16	.41	1.98	0	
UP.APRON.07	2	.01	3487	2757	79.07	.42	2.01	0	
UP.APRON.08	2	.01	3595	2781	77.36	.43	1.92	0	
ROOF..APRON	8	.09	9852	5907	59.96	1.32	3.29	0	
ROOF.QUEUE	13	.40	4926	1794	36.42	11.63	18.29	0	
DOWN.APRON.1	3	.00	3569	3094	86.69	.19	1.41	0	
DOWN.APRON.2	2	.01	3544	2985	84.23	.24	1.52	0	
DOWN.APRON.3	3	.01	3501	2964	84.66	.26	1.67	0	
DOWN.APRON.4	3	.01	3478	2949	84.79	.28	1.87	0	
DOWN.APRON.5	3	.01	3413	2885	84.53	.32	2.05	0	
DOWN.APRON.6	3	.01	3507	2964	84.52	.32	2.06	0	
DOWN.APRON.7	3	.01	3487	2948	84.54	.38	2.48	0	
DOWN.APRON.8	3	.01	3595	3045	84.70	.35	2.31	0	
ALL.VEHICLES	109	21.24	28094	0	.00	108.36	108.36	0	

QUEUE Details Continued.

```
EMPTY            0      .00      0     0    .00      .00       .00     0
CONT.HOUR.1     21      .63    645     0    .00   140.93    140.93     0
CONT.HOUR.2     30      .94    892     0    .00   150.83    150.83     0
CONT.HOUR.3     29      .85    769     0    .00   158.70    158.70     0
CONT.HOUR.4     23      .73    700     0    .00   149.78    149.78     0
CONT.HOUR.5     22      .64    636     0    .00   143.95    143.95     0
CONT.HOUR.6     25      .84    827     0    .00   146.08    146.08     0
CONT.HOUR.7     27      .98    883     0    .00   158.88    158.88     0
CONT.HOUR.8     28      .91    789     0    .00   165.39    165.39     0
CONT.HOUR.9     23      .77    679     0    .00   161.57    161.57     0
CONT.HOUR.10    13      .32    302     0    .00   150.92    150.92     0
LORR.HOUR.1     44     1.14   2032     0    .00    80.13     80.13     0
LORR.HOUR.2     64     1.70   2645     0    .00    91.90     91.90     0
LORR.HOUR.3     59     1.56   2277     0    .00    98.27     98.27     0
LORR.HOUR.4     50     1.20   1888     0    .00    90.95     90.95     0
LORR.HOUR.5     46     1.06   1818     0    .00    83.75     83.75     0
LORR.HOUR.6     51     1.35   2246     0    .00    86.13     86.13     0
LORR.HOUR.7     62     1.84   2708     0    .00    97.64     97.64     0
LORR.HOUR.8     57     1.67   2291     0    .00   104.73    104.73     0
LORR.HOUR.9     65     1.51   2140     0    .00   101.03    101.03     0
LORR.HOUR.10    28      .60    927     0    .00    92.81     92.81     0
CONT.FLOOR.1     8      .87    878     0    .00   142.47    142.47     0
CONT.FLOOR.2     9      .90    892     0    .00   145.34    145.34     0
CONT.FLOOR.3     9      .91    884     0    .00   147.53    147.53     0
CONT.FLOOR.4    12      .94    879     0    .00   152.56    152.56     0
CONT.FLOOR.5    11      .96    881     0    .00   156.23    156.23     0
CONT.FLOOR.6    11      .97    890     0    .00   156.54    156.54     0
CONT.FLOOR.7     9      .99    889     0    .00   160.30    160.30     0
CONT.FLOOR.8    11     1.06    929     0    .00   163.65    163.65     0
LORR.FLOOR.1    15     1.57   2691     0    .00    83.74     83.74     0
LORR.FLOOR.2    13     1.57   2652     0    .00    85.09     85.09     0
LORR.FLOOR.3    16     1.61   2617     0    .00    88.44     88.44     0
LORR.FLOOR.4    15     1.67   2599     0    .00    92.01     92.01     0
LORR.FLOOR.5    15     1.67   2532     0    .00    94.79     94.79     0
LORR.FLOOR.6    15     1.78   2617     0    .00    97.47     97.47     0
LORR.FLOOR.7    15     1.82   2598     0    .00   100.47    100.47     0
LORR.FLOOR.8    19     1.92   2666     0    .00   103.51    103.51     0
ALL.FLOOR.1     19     2.44   3569     0    .00    98.19     98.19     0
ALL.FLOOR.2     21     2.48   3544     0    .00   100.25    100.25     0
ALL.FLOOR.3     20     2.52   3501     0    .00   103.36    103.36     0
ALL.FLOOR.4     22     2.60   3478     0    .00   107.31    107.31     0
ALL.FLOOR.5     21     2.63   3413     0    .00   110.65    110.65     0
ALL.FLOOR.6     20     2.75   3507     0    .00   112.46    112.46     0
ALL.FLOOR.7     19     2.81   3487     0    .00   115.72    115.72     0
ALL.FLOOR.8     25     2.99   3595     0    .00   119.05    119.05     0
```

Daily Report Details (see comment on page 289).

	CHECK-IN	NUMBER	OF	VEHICLES			NUMBER WITH	UTILISATIONS		BUILDING
	CLOSING TIME	CONTAINERS	LORRIES	TOTAL	AVERAGE TIME	MAXIMUM TIME	LONG DELAY	CHECK-IN	BAYS	CLOSING TIME
COLUMN	1	2	3	4	5	6	7	8	9	10
ROW 1	1730	131	416	547	100	198	0	48	50	1833
ROW 2	1737	155	451	606	130	247	1	52	54	1855
ROW 3	1730	142	440	582	117	260	2	51	54	1834
ROW 4	1732	140	446	586	106	230	0	51	53	1838
ROW 5	1733	144	396	540	108	251	1	47	49	1849
ROW 6	1727	143	445	588	104	205	0	52	54	1838
ROW 7	1722	127	409	536	99	211	0	48	50	1823
ROW 8	1731	142	412	554	112	248	3	48	51	1840
ROW 9	1730	136	453	589	105	222	0	52	53	1840
ROW 10	1755	172	455	627	134	304	13	53	54	1930
ROW 11	1729	147	385	532	102	216	0	47	50	1837
ROW 12	1730	136	407	543	99	177	0	48	50	1842
ROW 13	1729	134	398	532	104	212	0	47	49	1839
ROW 14	1729	116	392	508	96	174	0	45	46	1834
ROW 15	1729	124	383	507	100	211	0	45	47	1830
ROW 16	1732	143	429	572	110	235	0	50	52	1842
ROW 17	1734	131	435	566	104	210	0	49	50	1857
ROW 18	1730	159	414	573	118	227	0	50	52	1854
ROW 19	1736	152	431	583	118	299	4	51	52	1903
ROW 20	1734	141	411	552	117	235	0	48	50	1856
ROW 21	1731	132	444	576	104	214	0	50	52	1837
ROW 22	1728	131	404	535	101	203	0	47	49	1834
ROW 23	1729	143	409	552	105	236	0	49	49	1900
ROW 24										

Daily Report (continued)

	1	2	3	4	5	6	7	8	9	10
Row 24	1744	171	423	594	124	296	4	51	53	1915
ROW 25	1739	150	471	621	121	257	1	54	56	1840
ROW 26	1730	144	422	566	106	216	0	50	52	1840
ROW 27	1729	156	400	556	106	221	0	49	52	1838
ROW 28	1730	152	400	552	105	213	0	48	52	1830
ROW 29	1731	154	394	548	115	252	3	48	51	1848
ROW 30	1733	144	421	565	108	243	1	49	52	1845
ROW 31	1729	143	429	572	104	232	0	50	52	1838
ROW 32	1729	154	456	610	124	301	9	54	56	1842
ROW 33	1749	161	426	587	124	260	1	50	53	1902
ROW 34	1732	128	440	568	98	186	0	50	51	1835
ROW 35	1733	134	458	592	113	265	2	52	52	1849
ROW 36	1727	119	402	521	96	183	0	46	47	1836
ROW 37	1732	156	446	602	110	214	0	53	54	1852
ROW 38	1754	152	484	636	122	244	3	54	56	1900
ROW 39	1729	125	394	519	98	219	0	46	48	1830
ROW 40	1729	144	391	535	100	181	0	47	50	1838
ROW 41	1732	156	400	556	106	216	0	49	51	1854
ROW 42	1730	142	419	561	102	218	0	49	52	1837
ROW 43	1729	133	399	532	102	201	0	47	49	1839
ROW 44	1731	142	415	557	100	210	0	49	52	1831
ROW 45	1729	136	426	562	100	199	0	49	51	1837
ROW 46	1729	132	383	515	101	200	0	45	48	1833
ROW 47	1730	135	370	505	99	219	0	44	47	1841
ROW 48	1728	135	418	553	110	255	7	49	50	1846
ROW 49	1728	151	398	549	103	180	0	48	52	1830
ROW 50	1730	152	422	574	108	223	0	50	53	1842
ROW 51	1755	7122	20972	28094	108	304	55	49	51	1930

Row 51 above is a summary giving (1) latest closing time for check-in (2) total containers (3) total lorries (4) total all vechicles (5) average time (6) maximum time in the godown (7) total number delayed more than 2 hours. (8) average check-in utilisation (9) average bay utilisation (10) latest building closing time.

11.2.8.3 Comment on Output

The significant parts of the output are the QUEUE details and the SAVEX MATRIX (the Daily Report). The comments relate to the version with no special restrictions on container movement.

Queue Details

ROOF.QUEUE	The total entries was 4926 (out of 28094 vehicles) indicating that 18% of vehicles had to proceed to the roof before parkig.
ALL.VEHICLES	The maximum contents were 109 indicating that at one time there were 109 vehicles in the building. The average time in the system for all vehicles was 108 i.e. 54 minutes. (Program time is in 30 second units).
CONT.HOUR 1-10	The average time in the system is slightly lower for vehicles which arrived in the first hour but does not vary much subsequently, indicating that steady-state conditions are reached quickly.
ALL.FLOOR 1-8	The average time in the system for vehicles increases as the destination floor increases but not a great deal. The difference between floor 1 and floor 8 is about 10 minutes on average.

Daily Report Details

The number of vehicles per day varied between 505 and 627 (column 4) the total being 28094 (row 51). The daily average time in the system (column 5) varied between 96 and 134. The maximum time for any vehicle was 304 (column 6). Day 10 was particularly bad with 13 vehicles being unduly delayed i.e. more than 2 hours (column 7). Columns 8 and 9 indicate utilisation of check-in was 49% and for bays was 51%. It should be noted that the utilisations calculated as standard output would not be correct as the total time would be taken as the total elapsed time on the basis of 24 hour days. The utilisation figures in the Daily Report matrix are however correct since they are calculated on the basis of the actual hours of operation.

CHAPTER TWELVE

USE OF PACKAGE WITH STUDENTS

12.1 STUDENT AND COURSE DETAILS

Units 1-6 plus the relevant appendices were tried out on a group of 50 students. These students were full-time students of the Department of Mathematical Studies, Hong Kong Polytechnic. They were doing a three year full-time course leading to a Higher Diploma in Mathematics, Statistics and Computing. In their second and third years they take a number of Operations Research courses one of which is Simulation.

The Simulation course for which this package was used took five weeks. Each week students had one lecture and one tutorial, each of an hour's duration. The class was split into three groups for tutorials. Each student thus had ten hours of class contact plus some informal sessions whenever they had problems.

It is the author's opinion that ten hours is not nearly enough but it is all that was available. As it happens since the students concerned had considerable mathematical ability and computer experience (mainly FORTRAN) they were able to cover the material with good project results. To allow sufficient time for analysis of project results which is a very important part of a Simulation course they should have had at least 20 hours. For non-mathematical students e.g. business students 30 to 40 hours of class contact would be required to cover the material in units 1-6.

The material in the teaching package is divided into nine units corresponding to Chapters 3-11 respectively of this thesis. Students studied these nine units consecutively. They completed individual questionnaires on each of the first three units (Chapters 3-5) which are introductory and then a further questionnaire on GPSS (Units 4,5 and Appendix E). The following sections deal with these questionnaires.

12.2 UNIT 1 PROBABILITY DISTRIBUTIONS

12.2.1 Questionnaire

Question 1 : Were the notes difficult to understand?

Sixty-five percent found them "easy", twenty-seven percent "very easy" and eight percent "moderate". Nobody found them difficult.

Question 2 : Should there be more Mathematics?

Fifty-eight percent thought that there should be and forty-two percent thought that there should not.

Question 3 : Should there be less mathematics?

Ninety percent thought that there should not be less, ten percent thought that there should be less.

Question 4 : Which terms did you not understand?

Nobody had any difficulties (apparently).

Question 5 : Was the material already familiar?

One hundred percent replied yes.

Question 6 : Was this unit useful?

Sixty-six percent found it useful, thirty-four percent did not

Question 7 : Give reasons why you found this unit useful/not useful.

Sixty percent gave reasons why they found it useful. They found it useful because of

(i) the revision it provided (36%)

(ii) a slightly different approach (17%)

(iii) an introduction to simulation (7%)

The forty percent who gave reasons for not finding it useful all gave the same reason viz. the material was already familiar to them.

Question 8 (a) : On what topics would you like more coverage?

Forty percent specified topics. The most common topics mentioned were

(i)	negative exponential/Poisson distributions	(14%)
(ii)	more examples	(12%)

Question 8 (b) : On what topics would you like less coverage?

Twenty-two percent specified topics. Virtually all were the same viz. frequency distributions or probability distributions

Question 9 : Could you follow the notes without any lectures?

Eighty-eight percent said yes and twelve percent no.

12.2.2 Analysis of Results of Questionnaire on Unit 1

As all of the students were "mathematical" the results were as might be expected i.e. all were already familiar with the material however the approach was different. For these students there was no difficulty with the material. Almost all felt that there should not be less mathematics. They were divided as to whether or not there should be more. In spite of the material being familiar two thirds still found it useful particularly because of the opportunity for revision using a more practical approach than that to which they were accustomed.

One can conclude that even for students with a mathematical/statistical background the unit should be retained although with such students it can be covered much more quickly than with non-mathematical students.

Many wanted more material on negative exponential/Poisson distributions. This would only be appropriate for mathematical-minded students who can be directed to a suitable text such as Maisel and Gnugnoli(1972). Some wanted more examples on Simulation. However this was really just anticipating subsequent units which they had not seen when completing the questionnaires on Units 1 and 2. Thus in reply to Question 7 only 7% found the unit on probability useful as "an introduction to simulation". They had not as yet seen the connection between probability and simulation which comes in Unit 2. However, as a consequence the author decided to expand on secondary examples such as the hair-salon example (see Appendix B)

12.3 UNIT 2 PROBABILITY AND RANDOM NUMBERS

12.3.1 Questionnaire

Question 1 : Were the notes difficult to understand?

Sixty-one percent said "easy" twelve percent very easy and twenty-seven percent moderate. This was almost the same as for Unit 1.

Question 2 : Should there be more mathematics?

Fifty-nine percent though that there should be and forty-one percent thought not (Again the same as Unit 1)

Question 3 : Should there be less mathematics

Ninety-four percent thought that there should not be less and six percent thought that there should be.

Question 4 : Which terms did you not understand?

Again, there was no response to this question.

Question 5 : Was the material already familiar?

Fifty-six percent replied no and forty-four percent yes.

Question 6 : Was this unit useful?

Eighty-four percent found it useful, sixteen percent did not.

Question 7 : Give reasons why you found this unit useful/not useful

Eighty percent gave reasons why they found it useful. Mostly these reasons concerned the following:

(i) new material (particularly simulation)

(ii) useful introduction

(iii) practical examples

(iv) random number treatment.

Twenty percent gave reasons for finding it not useful.

These all referred to the fact that the material was already familiar.

Question 8(a) : On what topics would you like more coverage?

Thirty-six percent wanted more on random numbers.

Sixteen percent wanted more examples.

Twenty percent wanted more on GPSS

Sixteen percent wanted more on mathematical/statistical topics

Question 8(b) : On what topics would you like less coverage?

Only six percent replied and all specified probability distributions.

Question 9 : Could you follow the notes without any lectures?

Sixty-nine percent said yes and thirty-one percent no

12.3.2 Analysis of Results of Questionnaire on Unit 2

The results were similar to those obtained for Unit 1. Few had any difficulties. Many (59%) felt that there should be more mathematics. Almost half were already familiar with the material. However eighty-four percent found it useful (compared to 66% for Unit 1).

A substantial minority (36%) wanted more on random numbers. This is natural for mathematics students. Appendix A partly caters for this. If a fuller treatment is required students could be referred to a suitable text such as Tocher (1963). It is to be expected that non-mathematics students would not require more mathematical detail.

12.4 UNIT 3 ANALYSIS OF REAL SYSTEM

12.4.1 Questionnaire

Question 1 : Do you think that the level of material in this unit was difficult, simple, medium?

Sixty-three percent found it medium, thirty-one percent simple and six percent difficult.

Question 2 : Do you think that a real time-study (i.e. with stop-watches) carried out by students to establish say an arrival rate distribution would be time wasting, very useful, interesting but not useful?

Forty-four percent said it would be time wasting, forty-four percent said it would be interesting but not useful and twelve percent said it would be very useful.

Question 3 : Can you suggest any improvements to the example used?

There was no response to this question.

Question 4 : What extra items would you like included in this unit?

There was only one reply to this. He suggested more "special techniques".

Question 5 : Could you follow this unit easily without lectures?

Sixty percent replied yes, thirteen percent no and twenty-seven percent "don't know".

12.4.2 Analysis of Results of Questionnaire on Unit 3

There seeemed to be no problems with this unit. Only six percent found it difficult and no-one had any suggestions for improvement. Sixty percent felt that they could follow it without lectures which is fewer than Unit 1 (89%) or Unit 2 (69%). This is understandable as this unit is less mathematical and more descriptive. Chinese students (as are all of the students at Hong Kong Polytechnic) have little difficulty with mathematics but as English is not their mother tongue they have some difficulty with descriptive material.

The number who thought that a time-study would be useful was disappointingly low, at least from the author's point of view, at twelve percent. However students here are particularly examination oriented and unless something has direct and obvious relevance to their final grade it is of little interest. If it was a marked project their reaction might be different. It is the author's opinion that some direct contact with reality (as distinct from abstract model building) however slight would be useful even if only to remind students that there is a real world out there to which all of these models apply.

The comments above on difficulties with descriptive material partly explain the lack of response to questions 3 and 4 which require comment rather than a tick.

12.5 UNIT 4 INTRODUCTION TO GPSS

UNIT 5 FUNCTIONS

MINI GPSS MANUAL

12.5.1 Questionnaire on Units 4, 5 and Mini GPSS Manual (Appendix E)

Question 1 : Did you find the material easy to follow?

Sixty-six percent answered "yes" and thirty-four percent "no".

Question 2 : Were the GPSS concepts explained, well, badly, reasonably?

Forty percent answered "well" and sixty percent "reasonably"

Question 3 : Did you find learning GPSS to be easy, difficult?

Seventy-three percent said "easy" and twenty-seven percent "difficult".

Question 4 : If you had not known FORTRAN would learning GPSS have been, the same, easier, more difficult?

Sixty-four percent answered "the same", ten percent "easier" and twenty-six percent "more difficult".

Question 5 : Which GPSS concept did you find most difficulty in understanding?

Only seven answered this. They suggested the following:

TABLE (2), MSAVEX (2), ORDER, PRIORITY, BLOCKS

Question 6 : Did you have any difficulty with the material in the appendices dealing with the GPSS language?

Seventy-one percent answered "no" and twenty-nine percent "yes".

Question 7 : What items in the appendices caused difficulty?

The items mentioned were as follows:

VARIABLE (2), PRINT-OUT, INDEXING, ORDER (2), ERROR MESSAGE, GATE, TABLE DESCRIPTION OF STATEMENT

Question 8 : How would you compare this material with using a conventional computer manual, much easier, easier, similar, more difficult?

Eleven percent answered "much easier", fifty-one percent "easier", twenty-nine percent "similar" and nine percent "more difficult".

12.5.2 Analysis of Results of Questionnaire on Units 4, 5 and Mini GPSS Manual

Most found the material easy to follow but a substantial minority (about one third) did have some difficulty as indicated in the answers to questions 1,2,3 and 6. To some extent this was because they had apparently tried to learn material from the mini GPSS manual which had not been covered in the units at that stage. This is obvious in the answers to questions 5 and 7 where almost all items referred to such as TABLE, MSAVEX etc. are not in fact dealt with until Unit 8 although they appeared in the manual. As a result the manual was rewritten. Originally there was a single manual, a mini GPSS manual for all students. This was rewritten as two separate manuals one for students only proceeding to Unit 6 and one for those who intend to complete the whole package. In fact it might be a good idea to supply all students even those proceeding to Unit 9 with the simple manual (Appendix D) only until they have completed Unit 6 when they could be supplied with the more detailed manual (Appendix E) even though the material in Appendix D is a sub-set of the material in Appendix E.

It is of interest that only twenty-six percent believed that a previous knowledge of a computing language (FORTRAN) was an advantage in learning GPSS. This is presumably because the structure of GPSS is so dissimilar from non-specialised languages (see discussion on page 18 of the Introduction).

Sixty-two percent found the method easier or much easier than using a conventional computer manual. With the revised appendices it is expected that this percentage should be much increased. In fact trying to learn from the old appendix was much the same as learning from a GPSS manual.

CHAPTER 13

SUMMARY, EXPERIENCE AND IDEAS FOR FURTHER RESEARCH

13.1 SUMMARY

The usefulness of simulation not only as a tool for problem solving but also as a means of teaching subjects which involve complex stochastic systems has been much enhanced by the increasing availability of computers and/or terminals to individual students. Simulation does not necessarily involve computers. However manual or physical simulation is severely limited because of the inherent slow speed of operation. Stochastic systems in particular require many repetitions in order to estimate the distributions of the variables involved. Consequently a considerable amount of work has been done recently in the field of "computer simulation". There is no doubt but that applications will grow at an increasing rate with the increasing availability of cheaper and faster computers. The use of simulation has been somewhat restrained because of the amount of expensive computer time required, however computer time is becoming much cheaper.

For the purpose of this thesis what is perhaps a somewhat restricted definition of simulation was adopted (see Introduction, section 1.1.1) :

"A Computer Simulation is the process of designing and operating a computer model of some system, which behaves in a way analogous to the operation of the original system. The purpose of such a computer simulation is to acquire an understanding of the behaviour of the original system under various conditions and/or to evaluate the effect of certain strategies on the operation of the system usually with a view to selecting the optimum strategy."

The thesis proposed a method of teaching simulation as defined above. The potential student was envisaged as having limited mathematical/statistical/computing background. The method advocated was to develop a simulation project into a nine unit teaching package which would simultaneously teach simulation and GPSS, the chosen simulation language (see Introduction, section 1.3.3)

Due to the presumed lack of quantitative background Units 1 and 2 introduced the required elements of probability theory in a non-mathematical manner. Such mathematical treatment which might be of benefit was relegated to Appendix A which is not essential. This has been done with the first objective in mind (see Introduction, section 1.4.2) namely :

"A student should find the package interesting enough to persevere to the desired end and should not be deterred by a lack of knowledge of, or a liking for, mathematics."

Unit 3 describes what a simulation model of a stochastic system is, by comparing a live study and a simulation study of a simple stochastic system. This together with subsequent units achieves the second objective namely :

"A student who completes the package should have an understanding of probability-based modelling as exemplified by discrete-time simulation."

Units 4 to 6 introduce GPSS and show how it can be used to model the system described in Unit 3.

The abbreviated form of the GPSS mini-manual provided (Appendix D) should be used in conjunction with these units. This describes a sub-set of the GPSS language. On completion of Unit 6 the third objective should have been achieved namely :

"The student should appreciate the value of a special simulation language such as GPSS and be able to write his own GPSS programs."

At this stage however the programs which a student could write would be relatively unsophisticated. Students who only require an introduction to the subject could stop at this point, objectives one, two and three having been achieved.

Units 7-9 intoduce further GPSS concepts and add further complexities to the model, culminating in a realistic simulation of an actual system. Once Unit 6 has been completed the more complete mini-manual should be used as the appropriate GPSS reference manual. At this point it can be claimed that the fourth objective has been achieved namely :

"The student should have had sufficient experience of handling successively more complex models to enable him to construct his own simulation models for different situations."

It should however be borne in mind that simulation is largely something which one learns by experience so it is never possible to say "this is the end".

However any simulation course must come to an end. The most one can hope to achieve is that the student has acquired the basic knowledge and skills to proceed by himself. If further experience is required then Appendix F, which is a report of the actual consultancy project which provided the basis of the material used in the package, provides many ideas on how further complexities can be added. At some stage however, it is more beneficial and more satisfying for the student to embark on projects of his own. He should be able to do this on completion of Unit 9.

Appendices B,C,F are supplementary to the package and not essential. Appendix B contains an alternative simulation exercise to be used if one wants to add some variety or to give students something extra to work on. Appendix C is similar to Appendix A in being more mathematical than the main text and is a discussion of the statistical aspects of simulation results. It would be of benefit to the more mathematically minded student. However like A it is not essential. Appendix F as has been mentioned above is a project report and could provide further experience for those who wish to proceed beyond Unit 9.

The content of the units constituting the package can be summarised as follows :

Units

1-2	Introduction to Probability
3	Introduction to Simulation
4-5	Introduction to GPSS
6	A Simple Computer Simulation
7-8	More Complex GPSS Concepts
9	A Computer Simulation of a Real System

The relevant appendices should be used as follows :

Appendices

A	Mathematical notes to be used in conjunction with Units 1 and 2 by students who could benefit thereby
B-C	Supplementary material. Appendix B can be used in parallel with Units 6-9. Appendix C can be used with mathematically minded students in conjunction with Units 8 and 9.
D	Abbreviated Mini GPSS Manual
E	Mini GPSS Manual
F	Report on actual case-study for student reading on completion of course.

13.2 EXPERIENCE WITH THE PACKAGE

The package was tried on a group of students who were asked to complete questionnaires at intervals. The results have been summarised in Chapter 12.

Generally, the package worked well and wherever shortcomings were apparent changes were made. The present package includes these changes. The main revisions made were the following :

1. The addition of Appendix B to satisfy requests for another example.

2. The writing of a simplified or rather partial manual for first use. This was because students tended to find the more complete manual somewhat daunting and tried to use certain concepts before they were ready for them.

3. Much of the explanatory material in the GPSS manuals was rewritten and expanded particularly items with which students had difficulty.

4. The addition of a section on analysis of output to Unit 6 for students who might not proceed further and of Appendix C on more detailed statistical analysis for students completing the package who have some previous statistical knowledge.

13.3 IDEAS FOR FURTHER RESEARCH

The teaching package described in Chapters 3-11 was devised as indicated in the Introduction to teach discrete simulation and GPSS to a wide (and in particular non-mathematical) audience. At the same time experience is gained of modelling and some statistics applications.

During research into the package and various side-issues some areas for future research became apparent, particularly in relation to statistical aspects of simulation in general and GPSS in particular.

It was the author's original intention to avoid statistical complications almost entirely for the following two reasons

1. Generally speaking prospective users of the package cannot be assumed to have a sufficient knowledge of statistics.

2. In many real-life applications statistical details such as confidence intervals will have little relevance.

Unit 6 contained some ideas concerning the interpretation of simulation results for those with no interest whatsoever in statistics. However especially in view of the fact that the experimental group of students did have a statistical background the author felt a little guilty at what might appear to be deliberate avoidance of a thorny issue and hence Appendix C on statistical validation was produced. This raised some problems which could be fruitfully researched. Following are four suggestions for further research.

13.3.1. The relevance of steady-state simulation of a queueing system and of the steady state solutions of queueing theory

In the discussion which follows the following notation is used:

λ = average arrival rate

s = average service time

$\rho = \lambda s$ (known as the traffic intensity)

c = number of servers.

The inter-arrival time distributions and service-time distributions will be understood to be negative exponential

In real-life it is almost impossible to find any example of a steady-state queueing system i.e. one which has an unvarying arrival-rate and an unvarying service-rate and which is in continuous operation. Most commercial systems are of the terminating type i.e. the systems close and open periodically. Some like road networks never close but have peaks and troughs of activity. A possible example would be arrivals of documents for processing by a clerk or typist. Suppose that they arrive at a constant rate between 09.00 and 17.00 and that the clerk or typist works at a constant rate. The clerk leaves at 1700 and resumes at 0900. A queue of documents which was on his desk at 1700 remains there until 0900 next day and he continues exactly where he left off. If one ignores time between 1700 and 0900 then this could approximate a steady-state system. The significant thing here is that the system resumes in the morning in exactly the state in which it finished the previous day. This would not happen in many systems (people will not remain on line all night). Law's (1983) state-of-the-art survey of statistical analysis quotes as his only example of a steady-state simulation a proposed computer system. However he does state "because the arrival rate of jobs will vary with the time of day or day of the week, there probably will not be steady-state measures for real-world computer systems. Assuming the arrival rate is constant over time in the model, however, will often allow steady-state measures to exist. In performing a steady state analysis of the proposed computer system, the model's developers are essentially trying to determine how the system will respond to a peak load of infinite duration; this is clearly a worst case analysis."

While experimenting with various models for Appendix C the author became convinced that there is a very big difference between peak-loads of finite duration (say up to 2000 observations) and of infinite duration. Convergence to steady-state values is very slow. The effect is most marked for loads which result in high system utilisations. For instance suppose that in a system* (M/M/1) where the average service time is 6 minutes there is a peak arrival rate of 9.8 arrivals per hour (i.e. $\rho = \lambda s = 0.98$) which lasts for two hours. The steady-state average queue size for $\rho = 0.98$ is 24. However in a two-hour period the average number of arrivals would equal only 19.6 and many of these would have been dealt with so at the end of two hours the maximum queue size could not be anywhere near 24 much less the average queue size. In fact it would take several days of continuous peak arrival rates before a mean of 24 was achieved.

These comments apply equally to the use of both steady-state simulation and steady-state queue formulae. It is the authors experience that many studies have used steady-state formulae in situations where in view of his recent experience they are not valid. While textbook authors allude to the fact that queueing formulae quoted are steady-state solutions the author has not come across any standard O.R. text which investigates or even points out the implications of this in particular

* using Cox's notation

the time-span over which a system must operate before the steady-state average is applicable. Many examples quoted (to illustrate steady-state formulae) are such that steady-state values would not be correct.

A number of well-known O.R. textbooks were selected at random in order to examine how quoted examples using steady-state formulae compared with a simulation of the terminating type and to demonstrate the inadvisability of using steady-state formulae.

Example from Operations Research by H.A. Taha (1976)

Example 13.1

On page 500 there is an example (13, 9) of a queue at a drive-in bank where λ =10 per hour, s=5 minutes c=1 server. The system is M/M/1. One question asked is "how long is an arriving customer expected to wait before starting service?" The answer given is 0.417. This has been obtained from the steady-state formula.

$$\text{average waiting time} = \frac{\lambda s^2}{(1-\lambda s)} = \frac{10}{12 \times 12(1-\frac{10}{12})} = 0.417 \text{ hour}$$
$$= 25 \text{ minutes}$$

On the assumption that the window would be open for 5 hours at a stretch one thousand independent simulations gave the following values (each replication simulated one 5 hour period)

average waiting time = 14.5 minutes with standard deviation 12.0 minutes

Based on these a 95% confidence interval would be 14.5 $\pm$ 0.76

It is obvious that the mean is much less than 25 minutes. Increasing the days length to 8 hours increases the mean to 17 minutes still much less than 25 minutes.

Examples from Operations Research for Management Decisions by S.B. Richmond (1968)

Example 13.2

A power company receives complaints at a rate of 2 per hour which are handled by a cruising radio controlled truck at a rate of 3 per hour.

The textbook gives "average wait before service" = 40 minutes.

Simulation of an 8-hour day gives 27 minutes.

(averaged over 500 days)

The standard error is 1.36

Example 13.3

Aircraft arrive at a rate of 9 per hour at an airport which can land 12 aircraft per hour.

The textbook gives "average wait before service" = 15 minutes

Simulation of a 4 hour period (it is unlikely that a peak rate would be maintained for more than 4 hours) gives 10 minutes based on 500 periods. The standard error is 0.38 minutes.

Examples from Quantitative Analysis by M. Hosking (1983)

Example 13.4

A team of 12 workers is employed to unload material from lorries delivering agricultural produce from different sources. The team works for a four-hour period each day during which 16 lorries arrive (4 in each hour). It takes 12 minutes to unload a lorry.

The formula is given as :

Average time a customer is in the sytem = $\frac{s}{1-\lambda s}$

(using this author's notation)

The formula given is valid only for Poisson arrivals, negative exponential service-time and steady-state conditions. The question seems to imply that exactly 16 lorries arrive each day. If that is the case arrivals are not Poisson since if the arrivals were Poisson with mean 4 per hour the actual number in a 4-hour period could vary considerably from 16. However if one assumes that what is intended is an average of 16 each day then arrivals could be Poisson.

In a 4 hour period with an arrival rate of only 4 per hour steady-state conditions would certainly not be achieved. In fact whereas the formula gives the average time in the system as 60 minutes, simulation gives 31 minutes, a very different answer indeed, in fact just looking at the waiting time (i.e. excluding service of 12 minutes) simulation gives 19 minutes and the steady-state formula gives 48 minutes!

What is required is some further research into the way in which steady-state formulae have been used by various authors of textbooks and in project work. It would also be of interest to determine for various models how long a time is required for a system to operate (in terms of numbers of arrivals) before it becomes reasonable to use steady-state formulae.

13.3.2 Quantities to be estimated from a simulation

In many studies (cf Ladany and Turban (1978) or Taylor and Keown (1980)) authors have tended to concentrate on average waiting time as the quantity to be estimated. However generally speaking variances are so large that in a study such as the godown study which formed the basis of this thesis, the average waiting time is not of much value. To a user of a system the average is not apparent. This is obvious when one considers how many observations must be made in order to establish the mean with any accuracy (typically several thousand). It would be more useful to specify some limit on waiting times and use the simulation to determine what proportion of observations exceed this limit. This measure is much more meaningful to a user. The owner of a system such as the godown referred to is concerned with minimising complaints from users. Users will not complain on the basis of a large average waiting time which they cannot perceive anyway but on the basis of occasional (or perhaps frequent) long delays. The quantity of interest would therefore seem to be the percentagé which suffer long delays. Alternatively it may be that what one should do is simulate say one month's operation of a system and then say to management "look, this is what happened in a typical month of operation" and let management draw its own conclusion without the analyst going to great lengths to establish say a 95% confidence interval for the mean waiting time. However this whole area of obtaining results from a simulation and presenting them in a useful format is worthy of further research.

13.3.3 Improvements to GPSS

While in general GPSS is a useful and easy to use language it does have some shortcomings particularly in relation to replication and statistical analysis. There should be an easy way to inform the simulator to repeat a simulation n times using different random numbers each time, to collect data from each replication and then to tabulate the n sets of data and produce a statistical analysis. For instance it could collect the mean waiting time Q (or any other useful measure) from each replication, tabulate the n values, determine whether they are normally distributed and produce a 95% confidence interval for the mean waiting time. It could decide on a value of n such that the width of the confidence interval is less than a value set by the programmer. It could determine how many observations there should be in each replication to ensure that Q is normally distributed for a specified n. In other words the type of analysis discussed in Appendix C should be built into the language.

13.3.4 Research into the further use of the suggested package and methodology

Some extensions are obvious such as changing the language to some other language which some might prefer to GPSS while retaining the same model. Other extensions might involve using a completely different model to demonstrate continuous as distinct from discrete simulation. In this context a useful development would be an economic model to teach economics, continuous simulation, a continuous simulation language such as DYNAMO and statistics. This would be most useful for students of economics.

The author's idea in using the godown model was to use a situation which is readily understandable to a wide range of users. Specific packages applicable to narrower areas such as the economics model just referred to could be developed. Another obvious teaching area is biology (e.g. ecological systems).

A model designed specifically to teach statistics would also be useful. This became apparent during the author's research on this thesis. In a relatively simple model there are useful applications for hypothesis testing e.g. is the mean queueing time equal to or less than the steady state value, is a certain distribution negative exponential etc., correlation analysis of output data, central limit theorem applied to batch size, design of experiment and analysis of variance and so on. Virtually the whole field of statistics could be covered starting with frequency distributions, mean and variance. Narrower uses have been suggested by other authors such as Reinhardt and Loftsgaarden (1978) who suggested using simulation to teach probability.

REFERENCES AND BIBLIOGRAPHY

Atkins, M.S.	'A Comparison of SIMULA and GPSS for Simulating Sparse Traffic' in Simulation 34, 3, 93-100, 1980
Beasley, J.E. and Whitchurch, G.	'O.R. Education - A Survey of Young O.R. Workers' in J. Opl. Res. Soc., 35, 4, 281-288, 1984
Bork, A.	'Learning with Computer Simulations' in Computer (IEEE), 12, 12, 75-84, 1979
Bronson, R.	'Teaching Simulation in Industrial and Academic Settings' in Simulation, 30,6,203-205, 1978
Carroll, J.M. and Laurin, P.E.	'Using Simulation to Assign Police Patrol Zones' in Simulation 36,1, 1-12, 1981
Dunn, S.	'Research and Mathematics Education' in Int. J. Math. Educ. Sci. Technol., 12, 2, 181-186, 1981
Fishman, G.S.	PRINCIPLES OF DISCRETE EVENT SIMULATION Wiley-Interscience, 1978
Gaither, N.	'Using Computer Simulation to Develop Optimal Inventory Policies' in Simulation 39, 2, 81-87, 1982
Gordon, G.	SYSTEM SIMULATION Prentice-Hall, 1978

Greilich W.	'Making Better Use of Jurors Through Computer Simulation' in Simulation, 33, 6, 195-202, 1979
Henize, J.	'Can a Shorter Workweek Reduce Unemployment?' in Simulation 37, 5, 145-156, 1981
Higgins, J.C.	'Teaching and Research in Quantitative Methods in University Management Schools' in Omega, 10, 1, 51-59, 1982
Hosking M.	Quantitative Analysis Financial Training Publications Ltd. (1983)
Ladany S.P. and Turban E.	'A Simulation of Emergency Rooms' in Computers and Operations Research 4, 89-100, 1978
Law, A.M.	'Statistical Analysis of Simulation Output Data' in Operations Research 31, 6, 983-1029, 1983
McMillan, C. and Gonzalez, R.F.	SYSTEMS ANALYSIS - A COMPUTER APPROACH TO DECISION MODELS Richard D. Irwin Inc., 1973
Maisel, H. and Gnugnoli, G.	SIMULATION OF DISCRETE STOCHASTIC SYSTEMS SRA, 1972

Markland, R.E.	'A Simulation Approach to Real Estate Investment Analysis' in Computers and Ops. Res. 6, 121-128, 1979
Moore, L.J. and Taylor, B.W. and Cattanach, K.A.	'A Network Model and Simulation Analysis of a Journal Review Process' in Computers and Ops. Res., 7, 267-275, 1980
Naylor, T.H. and Balintfy, J.L. and Burdick, D.S. and Chu K.	COMPUTER SIMULATION TECHNIQUES John Wiley and sons, 1968
Patel, K.P. and Janahanlal, P.S. and Ghista, D.N.	'Mathematical Simulation of Impact of Birth Control Policies on Indian Population System' in Simulation 41, 3, 103-117, 1983
Pidd M.	COMPUTER SIMULATION IN MANAGEMENT SCIENCE J. Wiley and Sons, 1984
Pritsker, A.A.B.	'Compilation of Definitions of Simulation' in Simulation, 33, 2, 61-63, 1979
Pritsker, A.A.B. and Pegden, D.P.	INTRODUCTION TO SIMULATION AND SLAM John Wiley and Sons, 1979
Reinhardt, H.E. and Loftsgaarden, D.O.	'Using Simulation to Resolve Probability Paradoxes' in Int. J. Math. Educ. Sci. Technol. 10, 2, 241-250. 1979

Reitman, J.	'The Use of Enhanced GPSS (GPSS/H)' in Simulation, 38, 1, 23-25, 1982
Richmond S.B.	Operations Research for Management Decisions The Ronald Press Company, 1968
Shannon, R.E.	SYSTEMS SIMULATION - THE ART AND SCIENCE Prentice-Hall, 1975
Taha H.A.	Operations Research Collier MacMillan, 1976
Taylor B.W. and Keown A.J.	'A Network Analysis of an Inpatient/ Outpatient Department' in J. Opl. Res. 31, 169-179 1980
Tocher K.D.	THE ART OF SIMULATION The English Universities Press Ltd., 1963
Wensley, J.R.	'Relative Success of Marketing Decision Aids A Survey' in Omega, 7, 1, 15-23, 1979

APPENDIX A

MATHEMATICAL NOTES

A.1 CHI-SQUARE TEST

To test whether a set of observed frequencies agrees with a set of expected or theoretical frequencies sufficiently well so that the differences can be attributed to chance, statisticians use a test called the chi-square test (or X^2-Test). "Chi-square" is a statistic which measures the difference i.e. the bigger the difference between observed and expected values the bigger will be Chi-square. The procedure in calculating Chi-square is as follows:

1. For each observed frequency (O_i) calculate the difference between it and the corresponding expected frequency (E_i) i.e. calculate $O_i - E_i$
2. Square this difference i.e. get $(O_i - E_i)^2$
3. Express this squared difference as a fraction of the expected value i.e get $\dfrac{(O_i - E_i)^2}{E_i}$
4. Add up the expressions $\dfrac{(O_i - E_i)^2}{E_i}$ for all observed frequencies i.e. get $\sum \dfrac{(O_i - E_i)^2}{E_i}$. This is chi-squared.

Note that chi-square is got from squared differences hence its name. It must of course be positive. For technical reasons the expected frequencies should not be too small. In practise take this to mean that E_i should always be at least 5. If any E_i is less than 5 then combine that class with an adjacent one so that for the new larger class the expected frequency is at least 5. Following is an extract from Table 3.9 and it can be seen that the expected frequencies for classes 5, 6, 7 and 8 are too small. Classes 5 and 6 and also 7 and 8 should be combined as shown.

TABLE A.1 (Extracted from TABLE 3.9)

Class Number	Time Between Arrivals	Observed Frequencies	Expected Frequencies
4	15 up to 20	12	7
5	20 up to 25	4	4
6	25 up to 30	4	3
7	30 up to 35	1	2
8	35 and over	0	4

TABLE A.2

(With small classes combined)

Class Number	Time Between Arrivals	Observed Frequencies	Expected Frequencies
4	15 up to 20	12	7
5 (old 5 + 6)	20 up to 30	8	7
6 (old 7 + 8)	30 and over	1	6

Chi-Square is now calculated for Example 3.4

TABLE A.3 (TABLE 3.9 with combined classes)

Class Number	Observed Frequencies O_i	Expected Frequencies E_i	$O_i - E_i$	$(O_i - E_i)^2$	$\frac{(O_i - E_i)^2}{E_i}$
1	10	24	-14	196	8.17
2	25	16	+ 9	81	5.06
3	15	11	+ 4	16	1.45
4	12	7	+ 5	25	3.57
5	8	7	+ 1	1	0.14
6	1	6	- 5	25	4.17
TOTAL	71	71	0		22.56

Thus chi-square is 22.56.

Is this too big? The answer to this depends on two factors:

(a) The degrees of freedom. This is a statistical concept related to the sample size. It is the number of independent expected frequencies which in this type of example equals "the number of classes -2" or 4 in this case. Note that there are not 6 independent values for E_i because since the total is fixed at 71 and the mean is fixed at 11.866 if any four are known then the other two can be obtained from the relationships $\sum E_i = 71$ and $\sum E_i x_i = 71 \times 11.866$

(b) How unlikly a result is one willing to accept and still atribute it to chance? Generally speaking the following criterion may be used; if the observed value of chi-square is so big that the probability of it being observed by chance is less than 0.05 one accepts that it is too big i.e. one accepts that the differences between observed and expected values are too great to have occured by chance and the hypothesis on which the expected frequencies are based must be wrong. Note, that this value of .05 is arbitrary and may not always be appropriate but it is a commonly accepted criterion.

How does one find this maximum value of chi-square? By consulting chi-square tables an extract from which is given below.

TABLE A.4 (Extract from Chi-square Tables)

Degrees of Freedom	PROBABILITY OF AN OBSERVED VALUE EXCEEDING THE TABULATED VALUE				
	.10	.05	.01	.005	.001
1	2.706	3.841	6.635	7.879	10.828
2	4.605	5.991	9.210	10.600	13.816
•					
•					
4	7.779	9.488	13.277	14.860	18.467
•					
•					

With 4 degrees of freedom and using .05 the maximum value of chi-square consistent with chance differences is 9.488. The observed chi-square (22.56) is much greater, too great to attribute the differences to chance so reject the hypothesis on which the expected frequencies were based, which was that the observed distribution is negative exponential. This means that for the process observed at least one of the conditions for a negative exponential distribution must not have been present.

However consider which of the following two questions is more appropriate.

1. Is the observed distribution a negative exponential distribution?

2. Does the negative exponential distribution give a sufficiently close approximation to the observed distribution in order for it to be used in place of the observed distribution?

The chi-square test endeavours to answer question 1. If the answer to question 1 is "yes" then there is no need to ask question 2. If the answer to question 1 is "no" as in the example just described then question 2 is relevant. Unfortunately it is not so easy to answer. It is largely judgemental.

A.2 FUNCTIONAL RELATIONSHIPS

A.2.1 Forms in which Functional Relationships May be Expressed

A.2.1.1 Table form

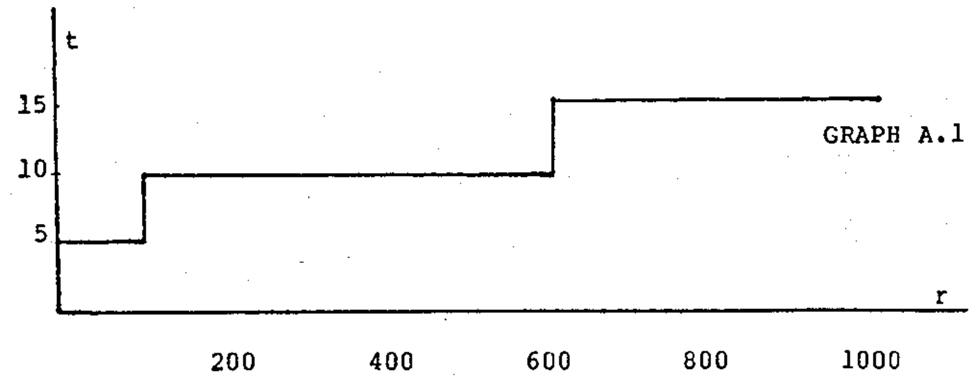
TABLE A.5

r	t
000 - 099	5
100 - 699	10
700 - 999	15

Given a value of r to find the corresponding t value simply look down the r column to find the row containing the given r then select the t on that row.

A.2.1.2 Graphical form

Given a value of r to find the corresponding t value look along the r axis to find the given r value. From that point draw vertical line to meet the curve and thence a horizontal line to the t-axis to give the appropriate t value.



GRAPH A.1

A.2.1.3 Explicit equation form

$$t = 5\left\{1 + \left[\frac{r + 500}{600}\right]\right\}$$

where the division is integer division i.e. on division take only the whole number part of the quotient.

Given a value of r, to find the corresponding value of t substitute the given r in the right hand side of the equation and evaluate it to get t.

As an exercise students should be asked to substitute each of the ten values for r in Table 4.3 into the above equation in turn and find the ten corresponding t values.

A.2.1.4 Implicit equation form

$$t^2 - 10t - 25\left[\frac{r + 500}{600}\right]^2 + 25 = 0 \qquad t > 0$$

Given a value of r one could substitute it into the above equation and then solve for t. In the example given this is easy since we have a quadratic equation but it would not be so simple if the implicit relationship were of a form such as $r^2 + art - br \log t = C$.

In all of the above it is assumed that to any given r there corresponds one and only one value of t. This is not true of functional relationships in general. If $t = r$ (form 3 above) then for any positive r greater than zero t will have 2 values. e.g. if $r = 9$ $t = +3$ or -3, if $r = 0$ t has only one value, if r is negative t has no value. We shall assume however that we are only dealing with relationships which give a unique value of t to any value of r. This is why the condition $t > 0$ was necessary in the implicit relationship used in A.2.1.4 above.

Exercise A.1

Using the graph find t corresponding to each of the ten values for r in Table 3.3.

Exercise A.2

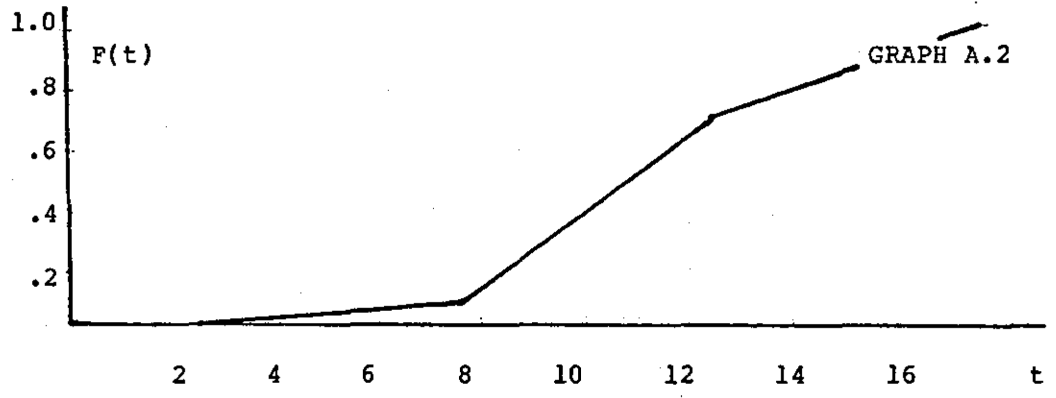
Students can be asked to substitute the ten values for r from Table 3.3 into the equation.

$$t^2 - 10t - 25\left[\frac{r + 500}{600}\right]^2 + 25 = 0 \qquad t > 0$$

and solve for t in each case.

A.2.2 Functional Relationship Between Time (t) and Random Number (r)

By the conventions of probability theory the graphical relationship between F(t) and t is shown as follows



The axes above are reversed if compared with the example in Graph 4.2 For the moment let us refer to F(t) as y and t as x since that is in accordance with the usual graphical convention. It is usual in probability theory to express the cumulative probability (F(t) or y) as a function of the variable x. Thus we have for three common theoretical distributions.

Binomial Distribution $y = \sum_{i=0}^{i=x} \frac{n!}{i!(n-i)!} p^i (1-p)^{n-i}$

Negative Exponential Distribution $y = 1 - e^{-\lambda x}$

Normal Distribution $y = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} \left\{ e^{-1/2 \left[\frac{x-\mu}{\sigma}\right]^2} \right\} dx$

It is not necesary to worry about the above formulae, the point is simply being made that the usual way for writing the formula is y = f(x) hence given a value for x by substituting in the right hand side of any of the above formulae one could get y.

The problem for simulation is that what one wants to do is the reverse of the above i.e. given a y value get x and that is not so easy. Ideally one should like to have an equation $x = f(y)$ the inverse of the above situation. If one has an equation $y = f(x)$ one could write the inverse symbolically as $x = f^{-1}(y)$ f^{-1} is the inverse of the function f.

Example A.1 What is the inverse of the function $(\quad)^2$?

if $y = (x)^2$

then $x = \sqrt{y}$ thus $\sqrt{}$ is the inverse of $(\quad)^2$

Example A.2 If $y = ax^2 + bx + c$ find the inverse.

Above can be written $ax^2 + bx + (c - y) = 0$

Applying the formula for a quadratic equation one obtains

$$x = 1/2a \left\{ -b \pm \left[(b^2 - 4a(c - y) \right]^{1/2} \right\}$$

In the above two examples inverting the function was easy. In general it is not so easy. Hopefully though the examples will convey an idea of what is meant by inverting a function.

To come back to simulation, one is usually given F(t) the cumulative probability as a function of the variable t but what one requires is t as a function of F(t).

Given $r = F(t)$

One requires an expression $t = F^{-1}(r)$

Example A.3 $r = 3t + 4 = F(t)$

$$t = F^{-1}(r) = \frac{r - 4}{3}$$

Example A.4 $r = F(t) = 1-e^{-\lambda t}$ invert this i.e. find an expression for t in terms of r

$$r = 1-e^{-\lambda t}$$

$$e^{-\lambda t} = 1-r$$

$$-\lambda t = \log_e(1-r)$$

$$t = -\frac{1}{\lambda}\log_e(1-r)$$

To show that actual algebraic inversion is rarely so straightforward a simple example will suffice.
Given $r = at^3 + bt^2 + ct + d + e/t$ What is t equal to?

Graphically all inversion amounts to is a reversal of axes (see Graphs 4.2 and 4.3) and graphical or tabular representation of functions is the only form to be used in this package.

A.3 GENERATION OF PSEUDO-RANDOM NUMBERS

To describe the usual procedure it is necessary first to define the use of the term "modulo", sometimes abbreviated "mod.

x(modulo a) means the remainder when x is divided by "a". Thus 9(modulo 6) = 3 500(modulo 13) = 6 821(modulo 100) = 21

Consider the formula $x_{n+1} = (3x_n + 7) \bmod 100$
Given any number x_n the formula gives the next one x_{n+1}

In non-mathematical language this procedure may be defined by the following steps

(a) multiply the current number by 3

(b) add seven to the result

(c) divide by 100 and take the remainder

(d) this remainder is the next number

For example suppose that the current number is 82

(a) $82 \times 3 = 246$

(b) $246 + 7 = 253$

(c) $253 \div 100 = 2$ with remainder 53

(d) the next number is 53

The table below shows how, starting with $x_0 = 25$ a series of "random" numbers 25, 82, 53, 66 ... is obtained.

TABLE A.6

x	3x	3x + 7	(3x + 7) mod 100
$x_0 = 25$	75	82	$82 = x_1$
$x_1 = 82$	246	253	$53 = x_2$
$x_2 = 53$	159	166	$66 = x_3$
$x_3 = 66$	etc.		

Obviously if a number is divided by 100 the remainder must be in the range 0 to 99 so x_n as generated above is always in the range 0 to 99. In GPSS the numbers generated are modulo 2^{35} so they will cover a range 0 to $2^{35} - 1$ or 0 to 34359738367. To convert to a range 0 to 1 they can be divided by 2^{35}.

In the above example three numbers were set arbitrarily; the "multiplier" 3, the "increment" 7 and the first value for x known as the "seed" 25. The proper choice of these numbers is important. They are chosen on the basis of rules derived from number theory so that the requirements of randomness are met. For a particular generator therefore a set of 3 numbers must be specified, the multiplier, generator and seed (3, 7 and 25 respectively in the above example.)
GPSS has in fact ten different (pseudo) random number generators each with a different set of parameters. These are given in Table A.7.

TABLE A.7

RANDOM GENERATOR	MULTIPLIER	INCREMENT	SEED
1	1220703125	0	6526266081
2	3141592653	2718281829	32828076357
3	2718281829	3141592653	5570918525
4	10604499373	7261067085	17801426661
5	17249876309	7261067085	17249876309
6	30517578125	7261067085	30517578125
7	2565727293	35981228	2565727293
8	107936437	4292354	107936437
9	22438762221	6891	22438762221
10	621444377	92111326	621444377

Example A.5

Use the random number generator defined by the formula

$$r_{n+1} = [(300r_n+7) \bmod 100]/100$$

to select a random sample of five arrival times from the distribution specified below. Use as seed (r_o) the value 0.25.

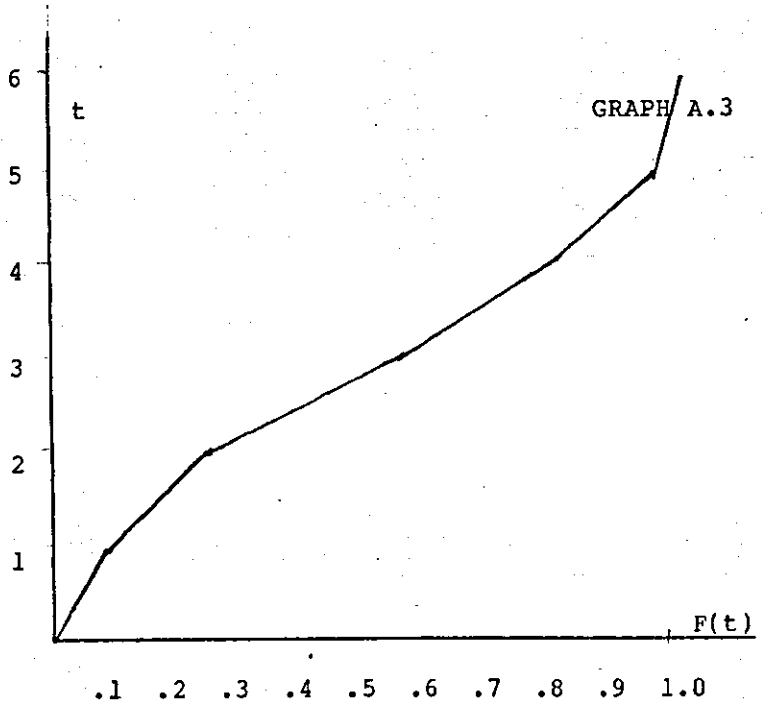
TABLE A.8

Inter-arrival time (minutes) t (Times between successive arrivals)	Relative Frequency Observed
0 up to but not including 1	.10
1 up to but not including 2	.15
2 up to but not including 3	.30
3 up to but not including 4	.25
4 up to but not including 5	.15
5 up to but not including 6	.05

First form the cumulative frequencies (probabilities) F(t) as in Table A.9 below and then plot as in Graph A.3.

TABLE A.9

t	F(t)
0	.00
1	.10
2	.25
3	.55
4	.80
5	.95
6	1.00



Now generate random numbers as on page 334. This gives the values r_i shown below. Read off the values of t corresponding to F(t) = r, from the Graph A.3. This gives Table A.10 below.

TABLE A.10

r_i	t (from GRAPH A.3)
.25	2.0
.82	4.1
.53	2.9
.66	3.5
.05	0.6

TABLE A.11

Inter-arrival Times	Arrival times (with respect to an initial t = 0)
2.0	2.0
4.1	6.1
2.9	9.0
3.5	12.5
0.6	13.1

The procedure just outlined is performed automatically by GPSS. The distribution (values of the variable and associated cumulative probabilities or relative frequencies) is specified in a FUNCTION definition statement. At the same time one specifies which of the ten available random number generators is to be used in connection with this FUNCTION. Then whenever in the program a request is made for a sample value for the function (e.g. inter-arrival time) the next random number in the series being used is obtained and the corresponding value of the FUNCTION is obtained in a manner similar to that just described and the value is returned as the required sample value.

APPENDIX B.

HAIR-SALON EXAMPLE

An example like this could be tackled in several ways depending on the type of student and in particular on how much guidance they require. If they are relatively mature and can act on their own initiative they should be forced to elicit any required information themselves i.e. the lecturer could play the role of the hair-salon owner and answer any questions put by the students but without volunteering any information. Other students could be given some guidance. The level of detail to be simulated is at the discretion of the lecturer but some suggestions are given below.

It is important that students write an analysis of the output in non-technical form and should not merely present the computer output as if that were "THE ANSWER".

B.1 SIMPLE SYSTEM

B.1.1 Service : There are two hairdressers each of whom performs exactly the same services with the same service time distribution. It is not necessary to distinguish between them. The service-time distributions are as follows :

HAIR-CUT ONLY

TIME	RELATIVE FREQUENCY
10 - 15 mins.	0.2
15 - 20 mins.	0.3
20 - 25 mins.	0.3
25 - 30 mins.	0.2

HAIR-CUT AND SHAMPOO

TIME	RELATIVE FREQUENCY
20 - 25 mins.	.1
25 - 30 mins.	.4
30 - 35 mins.	.3
35 - 40 mins.	.2

B.1.2 Customers : Sixty per-cent require hair-cut only, forty per-cent require hair-cut and shampoo. Arrivals are random at an average rate seven per hour, however if the queue size exceeds five, fifty per-cent of arriving customers would not wait.

B.1.3 Operation of System : The salon opens at 9:00 a.m. Assume that no customers would have arrived before that time. One server goes for lunch at 12:30 or if she is busy she will go when she is finished her current customer. She will return after one hour. The second server then leaves. You may assume that the second server can leave immediately the first one returns i.e. if the second server is occupied the first server will relieve her. (Consider the effect of not assuming this i.e. suppose that the second server had to complete her service before leaving). No customers will be admitted to the salon after 16:45. No customers will be admitted after 16:30 if both servers are occupied. The salon will close at 17:00 or as soon thereafter as all customers have been served.

B.1.4 Duration of Simulation : Simulate one day's operation.

B.1.5 Output Required

Average utilisation of the servers

Maximum queue size

Average queue size

Average time on queue

(standard outputs at 16:45 and at final close will provide all of the above)

Having run the above model successfully students should be asked to criticize the paucity of the output. What other information would be desirable and how could it be obtained?

e.g. (a) what percentage of customers had to queue for more than 1 hour?

(b) how many customers were lost because of the queue size?

(c) how many customers were turned away after 16:30?

(d) how bad was the situation at lunch time with only one server?

(e) what would be the effect of having one extra server?

(f) is one day's simulation sufficient?

B.2 MORE COMPLEX SYSTEM

To give more experience extra complications can now be added to the model. Below are some suggestions.

B.2.1 Service

(i) Servers

Three hairdressers

Two assistants who only wash hair

One manicurist

One cashier/receptionist

(ii) Service provided in the following order

(a) Hair washed - by assistant (distribution required)

(b) Hair dried - no server required (distribution required)

(c) Hairdressing (cut, set, whatever) by hairdresser (distribution required)

(d) Manicure - by manicurist (distribution required) - this may be done simultaneously with (b) and/or (c).

B.2.2 Customers

Some have appointments and will get preference over random arrivals. Their actual arrival time will be their appointment time plus or minus a small random time (distribution provided) - as people rarely arrival exactly on time. Appointment times can be specified e.g. one every half-hour (9:00, 9:30, 10:00 etc.). There will also be random arrivals.

Customers can request any of the services (a), (b), (c), (d) above except that (b) is compulsory if (a) is requested.

Each customer requires the receptionist/cashier for two minutes (constant) on arrival and for two minutes (constant) at departure. Assume that 50% of random arrivals would not wait if they observed a queue of more than 3. Assume also that no random arrivals would be admitted if the queue exceeds 6 and in this case they go away (i.e. they do not wait outside).

B.2.3 Operation of System

The salon opens at 9:00 am but customers may arrive from 8:45 and wait outside.

Each server gets one hour lunch break. Only one hairdresser at a time can go for lunch. Only one of the two assistants plus receptionist/cashier can go at one time (if the receptionist/cashier is absent an assistant will substitute and then will not be available for hair-washing). The manicurist goes for lunch from 1:00 to 2:00 pm and during that time no manicure service is provided so none is requested.

No customer will be admitted after 17:00 unless the server he/she requires is free. After 17:45 nobody will be admitted and the salon closes as soon thereafter as the last customer leaves. Individual servers may leave as soon as they are no longer required.

B.2.4 Duration of Simulation

Simulate 100 days of operation

B.2.5 Output Required

In addition to the output required for the simple model some extra information might be requested such as the following

(a) What is the average length of a working day for each server type?

(b) What was the latest time at which the salon closed?

(c) Given salary rates and a maximum achievable arrival rate of customers find the optimum staff distribution.

B.3 PROGRAM AND OUTPUT FOR SIMPLE SYSTEM

```
@HKU*USER.GPSS
 GPSS 4.1   -05/19-13:36-(000)

   *
   *
    JOB
   * PROGRAM TO SIMULATE HAIRDRESSING SALON
   *
   * TIMES ARE IN MINUTES AND C$1=0 IS 09.00 A.M.
   *
   *  LIST OF FUNCTIONS, VARIABLES AND STORAGES
   *
   INTERARRIVAL FUNCTION,EXP  RF$3,7,60
   HAIR.ONLY    FUNCTION,C    RF$1,0,10 .2,15 .5,20 .8,25 1,30
   HAIR.AND.SH  FUNCTION,C    RF$2,0,20 .1,25 .5,30 .8,35 1,40
   *
   REMAIN       BVARIABLE     (C$1 LT 465)  AND
   +                          (C$1 LT 450 OR S$HAIRDRESSERS NE 2) AND
   +                          (Q$Q1 LT 6 OR RI$4 GT 500)
   *
   HAIRDRESSERS CAPACITY        2
   *
   *
   *  MAIN PROGRAM
   *
 1     GENERATE 0 TIME(FN$INTERARRIVAL)  GO TO(L1,L2)
 2 L1  COMPARE BV$REMAIN EQ 1
 3     ASSIGN  C,RI$5/600                  @ C=0 (P=.6) C=1 (P=.4)
 4     QUEUE Q1
 5     STORE HAIRDRESSERS  TIME(FN$HAIR.AND.SH*P$C+FN$HAIR.ONLY*(1-P$C))
 6 L2  TERMINATE,R
   *
   *
   *       LUNCH SEGMENT
   *
 7     GENERATE 180,1,1                    @ TIME NOON
 8     STORE HAIRDRESSERS  TIME(120)       @ REDUCE CAP. TO 1 FOR 2 HRS.
 9     TERMINATE
   *
   *
   *       END OF DAY SEGMENT
   *
10     GENERATE 480,1                      @ TIME 5 P.M.
11     GATE SE,HAIRDRESSERS                @ ALL CUSTOMERS DEPARTED
12     STOP 1                              @ END SIMULATION
13     TERMINATE
   *
   START 500,,,240

STORAGES:
HAIRDRESSERS
```

PRINTOUT AT 1 p.m.

```
 RELATIVE     ABSOLUTE   TERMINATION
CLOCK TIME   CLOCK TIME     COUNT
      240          240        19

BLOCK  CURR TOTAL    BLOCK  CURR TOTAL    BLOCK  CURR TOTAL    BLOCK  CURR TOTAL    BLOCK  CURR TOTAL
  #    TRAN  TRAN      #    TRAN  TRAN      #    TRAN  TRAN      #    TRAN  TRAN      #    TRAN  TRAN
  1      0    26       2      0    23       3      0    23       4      6    23       5      1    17
  6      0    19       7      0     1       8      1     1       9      0     0      10      0     0
 11      0     0      12      0     0      13      0     0

  STORAGE     MAXIMUM   AVERAGE  MAXIMUM   AVERAGE   AVERAGE    TOTAL  TOTAL  AVERAGE    AVERAGE  CURRENT
   NAME      CONTENTS  CONTENTS CAPACITY  CAPACITY UTILIZATION ENTRIES TRANS ENT/TRANS  TIME/ENT CONTENTS
HAIRDRESSERS     2       1.94      2        2.00     .9708       18     18      1.00     25.89       2

  QUEUE      MAXIMUM   AVERAGE     TOTAL    ZERO     ZEROS   AV. TIME/ENT AV. TIME/ENT  CURRENT     TABLE
  NAME       CONTENTS  CONTENTS   ENTRIES  ENTRIES  PERCENT     (ALL)      (NON ZERO)   CONTENTS    NAME
Q1               6       2.44        23        4     17.39      25.48         30.84        6
```

PRINTOUT AT 5 p.m.

```
 RELATIVE     ABSOLUTE   TERMINATION
CLOCK TIME   CLOCK TIME     COUNT
      480          480        50

BLOCK  CURR TOTAL    BLOCK  CURR TOTAL    BLOCK  CURR TOTAL    BLOCK  CURR TOTAL    BLOCK  CURR TOTAL
  #    TRAN  TRAN      #    TRAN  TRAN      #    TRAN  TRAN      #    TRAN  TRAN      #    TRAN  TRAN
  1      0    59       2      0    43       3      0    43       4      7    43       5      2    36
  6      0    50       7      0     1       8      0     1       9      0     1      10      1     1
 11      0     0      12      0     0      13      0     0

  STORAGE     MAXIMUM   AVERAGE  MAXIMUM   AVERAGE   AVERAGE    TOTAL  TOTAL  AVERAGE    AVERAGE  CURRENT
   NAME      CONTENTS  CONTENTS CAPACITY  CAPACITY UTILIZATION ENTRIES TRANS ENT/TRANS  TIME/ENT CONTENTS
HAIRDRESSERS     2       1.97      2        2.00     .9854       37     37      1.00     25.57       2

  QUEUE      MAXIMUM   AVERAGE     TOTAL    ZERO     ZEROS   AV. TIME/ENT AV. TIME/ENT  CURRENT     TABLE
  NAME       CONTENTS  CONTENTS   ENTRIES  ENTRIES  PERCENT     (ALL)      (NON ZERO)   CONTENTS    NAME
Q1               9       4.59        43        4      9.30      51.19         56.44        7
```

CLOSING TIME (6.21 p.m.) PRINTOUT

```
 RELATIVE     ABSOLUTE   TERMINATION
CLOCK TIME   CLOCK TIME     COUNT
      561          561        73

BLOCK  CURR TOTAL    BLOCK  CURR TOTAL    BLOCK  CURR TOTAL    BLOCK  CURR TOTAL    BLOCK  CURR TOTAL
  #    TRAN  TRAN      #    TRAN  TRAN      #    TRAN  TRAN      #    TRAN  TRAN      #    TRAN  TRAN
  1      0    73       2      0    43       3      0    43       4      0    43       5      0    43
  6      0    73       7      0     1       8      0     1       9      0     1      10      0     1
 11      0     1      12      1     1      13      0     0

  STORAGE     MAXIMUM   AVERAGE  MAXIMUM   AVERAGE   AVERAGE    TOTAL  TOTAL  AVERAGE    AVERAGE  CURRENT
   NAME      CONTENTS  CONTENTS CAPACITY  CAPACITY UTILIZATION ENTRIES TRANS ENT/TRANS  TIME/ENT CONTENTS
HAIRDRESSERS     2       1.96      2        2.00     .9813       44     44      1.00     25.02       0

  QUEUE      MAXIMUM   AVERAGE     TOTAL    ZERO     ZEROS   AV. TIME/ENT AV. TIME/ENT  CURRENT     TABLE
  NAME       CONTENTS  CONTENTS   ENTRIES  ENTRIES  PERCENT     (ALL)      (NON ZERO)   CONTENTS    NAME
Q1               9       4.23        43        4      9.30      55.19         60.85        0
```

B.4 CONSIDERATION OF OUTPUT

The program is a simple one, deliberately so, in order to show how few instructions are required in GPSS for a realistic simulation.

A "Boolean Variable" (see GPSS mini-manual) has been used to test whether an arriving customer should remain. These conditions are as follows:

(a) The time must be less than 16:45

(b) If the time is between 16:30 and 16:45 at least one server must be free

(c) If the queue length exceeds 5 there is a 50% chance of not remaining.

A single Boolean variable can test all three. It will have a value 1 if all clauses are true, zero if any one is false.

B.4.1 Output Analysis

Simulation started at time O (representing 9:00a.m.). Note that if one used t=540 to represent 9:00 a.m. i.e. if one generated the first customer at t=540 some of the output statistics would be incorrect. The program would assume that the hairdressers and queue had been there since t=0 and hence average utilization and average contents would be wrong.

As far as average utilization of hairdressers is concerned one should be aware that the lunch break would have been treated as "working" since the corresponding storage was occupied. If one wants utilization to exclude the lunch period one can adjust the figures as follows:

Length of day = 561 minutes (realtive clock time at end of simulation)

Length of working day (excluding lunch) = 561-60 = 501

Time spent working = 561 x utilization = 561x.9813 = 550.5

Time spent working (excluding lunch) = 550.5-60 = 490.5

Utilization (excluding lunch) = (490.5 501)x100 = 97.9%

The important facts shown by the output are as follows:

From the intermediate printout at 5:00p.m. (t=480) it can be seen that fifty-nine customers arrived between 9:00a.m. and 5:00p.m. Forty-three of these were served and sixteen left without being served. If one assumes that between 4:30p.m and 5:00p.m. four of these would have arrived (an average rate of seven per hour) then twelve were "lost" earlier because the queue exceeded five and they decided not to remain. Since it may be assumed that no customers would have been accepted after 4:30p.m. (there was such a long queue at 5:00p.m. that both servers must have been occupied at 4:30p.m.) a customer who arrived at 4:30p.m. (the last customer) did not leave until 6:21p.m. (closing time) - almost two hours later. It definitely seems that an extra server is required. The current average queue size (4.59 as of 5:00p.m.) is too big. This suggests re-running the simulation with an increased STORAGE capacity of three to see what happens.

B.5 SUGGESTION FOR EXTENDING THIS EXAMPLE

Assuming that one extra hairdresseer would cost $20 per day including increased overheads and that the average customer pays $5.00 of which $0.50 is for materials used would one be justified financially in hiring the extra hairdresser? Assume that the arrival rate is unchanged although in practise one could expect the improvement in service to cause an increase in the arrival rate.

APPENDIX C

ACCURACY OF OUTPUT

C.1 STATISTICAL CONSIDERATIONS

Conventional statistical formulae for confidence limits assume that individual sample values are independent. Thus if a sample of 100 independent values of x give as an estimate of the population mean a value $\bar{x}$ and as estimate of the population standard deviation a value s then the standard deviation of $\bar{x}$ is given by $s/\sqrt{n}$ and a ninety-five percent confidence interval for the population mean is given approximately by $\bar{x} - \frac{2s}{\sqrt{n}} \leqslant \mu \leqslant \bar{x} + \frac{2s}{\sqrt{n}}$.

This analysis cannot easily be applied to simulation output. Suppose that $\bar{Q}$ = 45 minutes is the mean queueing time of 100 customers in a simulation run, and that s = 15 minutes is the estimated standard deviation, one could form as a 95% interval for the mean queueing time of all customers the interval

$$45 - \frac{15 \times 2}{\sqrt{100}} \text{ to } 45 + \frac{15 \times 2}{\sqrt{100}}$$

i.e. 42 to 48

However this is not valid because the 100 values are not independent. Just consider two consecutive customers numbered n and n+1. Are their queueing times independent? Clearly not, as it is obvious that if customer n queues for a long time customer n+1 will probably do so also. Similarly if customer n has zero queueing time then customer n+1 will have a short queueing time if any. The consequence of this is that although the standard deviation of individual queueing times is calculated as s the standard deviation of $\bar{Q}$ is not $\frac{s}{\sqrt{n}}$ and in fact cannot easily be obtained from s.

One way around this dilemma is to carry out a number n of independent simulation runs (or replications as they are called) say n=30. Suppose that the first run gives $\bar{Q}_1$, the second $\bar{Q}_2$ and so on. Then one has 30 independent estimates of the mean queueing time. These will be independent because although values within a run are dependent because of the sequential nature of the process, successive runs started afresh and using different random numbers are independent. One could estimate the standard deviation of $\bar{Q}$ from the thirty values. Call this estimate s. Let $\bar{\bar{Q}}$ be the mean of the thirty values $\bar{Q}_1$, $\bar{Q}_2$ ... $\bar{Q}_{30}$ then the standard deviation of $\bar{\bar{Q}}$ is given by $s/\sqrt{30}$. A ninety-five percent confidence interval for the mean queueing time is then given by

$$\bar{\bar{Q}} - \frac{2s}{\sqrt{n}} \leq \mu \leq \bar{\bar{Q}} + \frac{2s}{\sqrt{n}} \qquad \text{(n=30 in the example)}$$

where μ is the mean queueing time.

It would be more appropriate to use $t_{.025}$ in place of 2 in the above but for n=30 this would be 2.05 so there is not much difference.

It is assumed in the construction of the confidence interval that $\bar{\bar{Q}}$ (the mean of the $\bar{Q}$'s) is normally distributed. This will be so for large n (by the central limit theorem) no matter what distribution $\bar{Q}$ has. Frequently however n cannot be large due to the cost of replications. In many cases it may not be reasonable to do more than say 4 replications of a large-scale simulation. In this case $\bar{\bar{Q}}$ can be assumed to be normal if $\bar{Q}$ is approximately normal and this will be so provided that the number of "customers" in each replication is large. A comparison of $\bar{Q}$ and s will give some idea of whether this assumption ($\bar{Q}$ being approximately normal) is unreasonable. E.g. if $\bar{Q}$ = 25 minutes and s = 28 minutes (s is the standard deviation of $\bar{Q}$) then the assumption would be unreasonable since a normal distribution has about 16% of its values more than 1 s.d. below the mean and if $\bar{Q}$ is 25 and s=28 no values can be more than 1 s.d. below the mean (25-28=-3!). In practise the standard deviation should be less than 0.4 of the mean to assume even approximate normality (this is not a sufficient condition but a necessary one).

In the case of terminating simulations replications occur naturally e.g. in the case of a system which opens and closes every day then each day is a replication. In the case of a steady state simulation the end of a single run or replication has to be chosen arbitrarily.

For steady state simulations there is the problem of starting conditions in each replication. Thus suppose that one carries out n replications with m "customers" in each replication giving $\bar{Q}_1, \bar{Q}_2 \ldots \bar{Q}_n$ as the mean queueing times in each replication. These $\bar{Q}_i$ are not estimates of the steady state mean waiting time (call this μ) but of the mean waiting time of the first m customers (call this μ_m). In general $\mu_m < \mu$ because early customers have little or no waiting time. The confidence interval one would construct would be for μ_m not μ. Increasing n (for a fixed m) would not help. It would merely get a better estimate for μ_m. Increasing m would help because as $m \rightarrow \infty$ $\mu_m \rightarrow \mu$. Also neglecting the first k ($k < m$) observations in each replication would help.

An alternative to replications in steady state simulation is to divide one large simulation run into batches and treat the batches as was suggested for replications. Thus instead of n replications with m observations or "customers", one might have a single run of mn observations divided into n batches with m in each batch. The problem of starting conditions is avoided except in the first batch which can be ignored. The problem is that just as successive "customers" are not independent, successive batches are not independent. For instance if batch k ends up with a "traffic jam" then batch k+1 will start with a traffic jam and the mean queueing times of both batches will be similarly affected. This effect is somewhat diluted if the batch sizes are large. For large m the batch means are approximately independent and also will be approximately normally distributed.

A ninety-five percent confidence interval for μ is given by

$$\bar{\bar{Q}} - \frac{ts}{\sqrt{n}} \leqslant \mu \leqslant \bar{\bar{Q}} + \frac{ts}{\sqrt{n}}$$

where $\bar{\bar{Q}}$ = mean of all replication (or batch) means

n = number of replications (batches)

s = standard deviation of replication (batch) means

t = t(n-1, .025) [obtained from t-table], approximately 2

It is difficult to say anything in general about the appropriate values for m and n, as this will depend on the structure of the particular model. In the case of terminating simulations m is determined by the "natural" termination and does not have to be chosen. n has to be chosen however and generally its value will be decided by a compromise between the following considerations:

1. n should be large enough to ensure that $\bar{Q}$ is normally distributed
2. The larger n is, the more expensive will be the simulation

3. The larger is m the smaller need n be to ensure normality
4. The larger is n the smaller is the confidence interval (i.e. the estimate is more accurate)

In the case of batches both m and n must be chosen. m must be chosen large enough to ensure that there is very little correlation between batch means. There are tests for such correlation but unfortunately they require a large number (n) of batch means. The choice of n depends mainly on 2 and 4 above.

C.2 Suggested Exercises

Write a GPSS program for the following simple system. Customers arrive onto a single queue at random times (negative exponential inter-arrival time distribution) at an average rate of λ per hour. Service time also has a negative exponential distribution with a mean service time of s hours. There are c servers and no maximum system capactiy and all customers wait until served (i.e. the system is the familiar M/M/C(∞, FIFO) system of queueing theory). The simulation is to be of the steady-state type

1. Investigate the effect of starting conditions by obtaining $\overline{Q}_m$ the mean queueing time for the first m customers for m=10, 100, 500, 1000, 10,000 for selected values of λ, s, c such as (8, 0.1, 1) (9, 0.1, 1) (47, 0.1, 5) (48, 0.1, 5) (49, 0.1, 5).

2. Investigate the effect of excluding the first k customers by obtaining $\overline{Q}_{m,k}$, the mean queueing time obtained after excluding the first k customers of a batch of m for values such as
   m=5000 k=50,100,500,1000 or
   m=10,000 k=100,500,1000,2500
   For fixed m=10000 plot $\overline{Q}_{m,k}$ against k for $0 \leq k \leq 2500$

3. Obtain a ninety-five percent confidence interval for the mean queueing time by simulating mn customers and obtaining batch means $\overline{Q}_j$ as described on page 354 (j=1,2,...n).

4. Rewrite the program as a terminating simulation which terminates after eight hours (one day's operation) or whenever the last customer has left, whichever is later (no customers being admitted after 8 hours). Use the method of replications (pages 352-3) to obtain a confidence interval for μ, the mean queueing time. Use values of n=10, 30, 50. With n=50 tabulate the fifty mean values $\bar{Q}_j$(j=1,2...50) obtained and compare with a normal distribution.

Investigate the distribution of $\bar{Q}_j$ (the daily means) and also of $\bar{\bar{Q}}$ the mean of n values of $\bar{Q}_j$ for n=10,20.

i.e. $$\bar{\bar{Q}}_1 = \frac{1}{n}\sum_{j=1}^{n} \bar{Q}_j$$

$$\bar{\bar{Q}}_2 = \frac{1}{n}\sum_{j=n+1}^{j=2n} \bar{Q}_j$$

etc.

SUGGESTED PROGRAM FOR EXERCISE 1

SINGLE SERVER MODEL M/M/1

```
@HKU*USER.GPSS
 GPSS 4.1   -11/02-16:27-(000)

    1       * FILE NO. BM1:KG5025.GPS TO SHOW CONVERGENCE OF MEAN OF FIRST M
    2        JOB
    3       *
    4       *
    5       INTERARRIVAL FUNCTION,EXP    RF$1,8,60      @ 8 PER HOUR
    6       SERVICE.TIME FUNCTION,EXP    RF$2,10,60     @10 PER HOUR
    7       QT1          QTABLE          10,10,10,Q1
    8       *
    9    1               GENERATE        0              TIME(FN$INTERARRIVAL)
   10    2               QUEUE           Q1
   11    3               HOLD            SERVER         TIME(FN$SERVICE.TIME)
   12    4               TERMINATE,R
   13       *
   14       *
   15       *
   16       *
   17       START 1000
```

MULTI-SERVER MODEL M/M/5

```
@HKU*USER.GPSS
 GPSS 4.1   -11/27-02:00-(000)

    1       * FILE NO. BM1:KG5025.GPS TO SHOW CONVERGENCE OF MEAN OF FIRST M
    2         JOB
    3       *
    4       *
    5       INTERARRIVAL FUNCTION,EXP    RF$3,49,600
    6       SERVICE.TIME FUNCTION,EXP    RF$4,10,600
    7       QT1          QTABLE          100,100,10,Q1
    8       SERVER       CAPACITY        5
    9       *
   10    1               GENERATE        0              TIME(FN$INTERARRIVAL)
   11    3               QUEUE           Q1
   12    4               STORE           SERVER         TIME(FN$SERVICE.TIME)
   13    5               SAVEX           NO,X$NO+1      GO TO(L2,L3)
   14    6               COMPARE         X$NO//100 EQ 0
   15    7               PRINT           X$NO,QT$Q1     @INTERMEDIATE RESULTS
   16    8               TERMINATE,R
   17       *
   18       *
   19       *
   20       *
   21       START 1000
```

SAMPLE RESULT

To Demonstrate the Convergence of $\overline{Q}_m$ to μ as $m \rightarrow \infty$

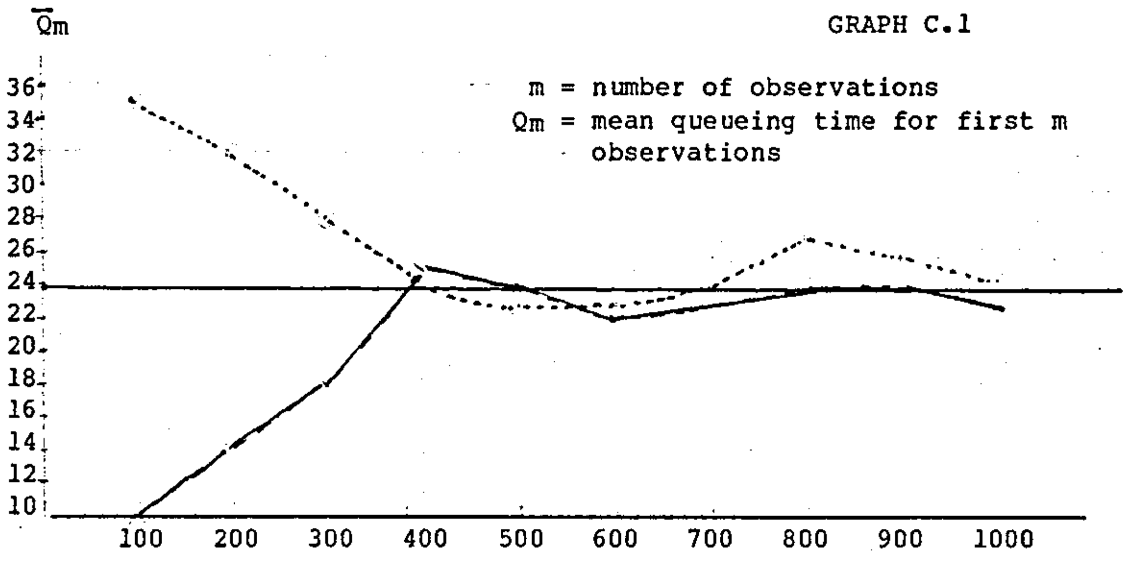
$\overline{Q}_m$ = Mean of first m observations

μ = Steady state mean queueing time

Two programs are shown on page 358 for M/M/1 and M/M/5 models. [Note, M here stands for Markovian, as the arrival and service distributions being negative exponential are examples of Markovian processes]. They produced the following results.

Sample results for $\lambda = 8$ S = 0.1 c = 1 are shown below

GRAPH C.1



Note: The two sets of data (solid line and dotted line) were obtained using different random numbers. The results are tabulated on the next page.

TABLE C.1

m	Set 1 $\overline{Q}_m$	Set 2 $\overline{Q}_m$
100	10	35
200	14	32
300	18	28
400	25	24
500	24	23
600	22	23
700	23	24
800	24	27
900	24	26
1,000	22.6	24.4
10,000	25.1	24.5
20,000	25.7	23.4
30,000	24.2	23.3
40,000	25.1	24.0
50,000	25.3	22.9
60,000	24.8	22.7
70,000	25.3	23.0
80,000	25.0	23.0
90,000	24.7	23.3
100,000	24.6	23.6

The two sets were obtained in two independent runs using different random number generators.

Note that because of the large variance convergence to the theoretical steady state value 24 is slow.

Steady state formula

λ = arrival rate (inter-arrival times are negative exponential)

S = average service time (service times are negative exponential)

Average queueing time $\mu = \frac{\lambda s^2}{1-s\lambda}$ where $s = 0.1$ hour, $\lambda = 8$ per hour

$$= \frac{(.1)^2(8)}{1-(.1)(8)}$$

$= 0.4$ hour $= 24$ minutes

.Sample results for multi-server model with 5 servers (c=5) and average service time 6 minutes (s = 0.1 hour)

Table C.2

m	$\overline{Q}_m$		
	λ = 47	λ = 48	λ = 49
100	3	4	9
200	6	6	14
300	8	9	17
400	7	8	19
500	6	7	21
600	5	6	22
700	6	7	22
800	10	11	23
900	13	14	24
1000	13	15	23
2000	12	16	21
3000	13	16	24
4000	15	22	45
5000	14	22	59
6000	16	23	64
7000	16	24	71
8000	15	22	77
9000	14	21	76
10000	14	20	75
Steady State Formula	17	27	57

Steady State Formula $\mu = \frac{s}{(c-\rho)}\left[\frac{(c-1)!(c-\rho)}{\rho^c}\sum_{0}^{c-1}\frac{\rho^i}{i!}+1\right]^{-1}$

where $\rho = \lambda s$.

Exercise 2

Page 363 shows the program for $m = 5,000$ and

$k=50,100,500,1000$

The example is for $\lambda = 9$/hour $s = .1$ hour and $c = 1$

It can be seen that the mean for the first k is always less than for the subsequent m-k although with m = 5000 only, the agreement with the theoretical value $\mu = 54$ is not very good. This is partly due to the fact that for relatively high arrival rates ($\rho = \lambda s \geqslant .9$) the mean μ is very sensitive to actual arrival rates and the actual time simulated was 32977 minutes for 5000 arrivals which is an actual rate of 9.097 per hour for which μ would be 60.4 minutes. The reason for the difference between the specified rate of 9 per hour and the actual rate of 9.097 per hour is that GPSS generates arrivals randomly in such a way that the average rate approaches the specified rate (9 per hour in this case) as m approaches infinity but in any run the actual rate observed can be slightly more or less than that specified.

```
@HKU*USER.GPSS
 GPSS 4.1   -11/07-19:10-(000)

        * FILE NO. BM1:KG5026.GPS TO SEE EFFECT OF IGNORING FIRST L OF M
          JOB
        *
        *
        INTERARRIVAL FUNCTION,EXP    RF$1,9,60
        SERVICE.TIME FUNCTION,EXP    RF$2,10,60
        QT1          QTABLE          10,10,10,Q1
        *
   1                 GENERATE        0             TIME(FN$INTERARRIVAL)
   2                 QUEUE           Q1
   3                 HOLD            SERVER        TIME(FN$SERVICE.TIME)
   4                 TERMINATE,R
        *
        *
        *
        *
        START 50
         RESET
        START 4950
         CLEAR
        *
        START 100
         RESET
        START 4900
         CLEAR
        *
        START 500
         RESET
        START 4500
         CLEAR
        *
        START 1000
         RESET
        START 4000
         END

@FIN
```

$\overline{Q}_k$ = mean of first k observations

$\overline{Q}_{m-k}$ = mean of observation k+1, k+2, ... m

The two sets were obtained using different random numbers

m = 5000

TABLE C.3

k	Set 1		Set 2	
	$\overline{Q}_k$	$\overline{Q}_{m-k}$	$\overline{Q}_k$	$\overline{Q}_{m-k}$
0	-	64	-	42.8
50	7	65	24	43
100	45	65	16	43
500	42	66	17	46
1000	43	69	24	48
Theoretical		54		54

One would not expect the effect of deleting the first k observations to be as marked in the single-server case as in the multi-server case so a further example is shown on page 365 for a multi-server situation. In a simple single-server queue situation it is not all that unusual to have an empty system. However in a complex system such as a multi-storey warehouse it would be extremely unusual to find the system completely empty.

Results for M/M/C model m = 10,000 and

k = 50,100,500,1000, 2500

The examples are for λ = 45,46,47 respectively with s=6 minutes in all cases. Two independent results are given for each value of λ .

TABLE C.4

k	λ = 45				λ = 46				λ = 47			
	Set 1		Set 2		Set 1		Set 2		Set 1		Set 2	
	$\bar{Q}_k$	$\bar{Q}_{m-k}$	$\bar{Q}_k$	$\bar{Q}_{m-k}$	$\bar{Q}_k$	$\bar{Q}_{m-k}$	$\bar{Q}_k$	$\bar{Q}_{m-k}$	$\bar{Q}_k$	$\bar{Q}_{m-k}$	$\bar{Q}_k$	$\bar{Q}_{m-k}$
50	4	10	3	9	3	9	3	13	4	11	3	18
100	3	10	8	9	2	9	8	13	3	11	8	18
500	10	10	6	10	2	9	7	13	3	11	11	18
1000	11	10	6	10	4	10	8	13	5	12	10	19
2500		11	7	10	7	10	9	14	9	12	11	20
Steady State	9				12				17			

The simulation was also carried out for m=50,000 with k=10,000 to see if much larger values of m,k would give better results. (one set only)

TABLE C.5

k	λ = 45		λ = 46		λ = 47	
	Q_k	Q_{m-k}	Q_k	Q_{m-k}	Q_k	Q_{m-k}
10,000	7.5	8.5	9.2	11.1	11.1	15.2
Steady State	9		12		17	

EXERCISE 3

For M/M/1 with λ = 8 per hour, s = 6 minutes the batch means of 100 batches of 2000 were calculated. They are tabulatd below and compared with a Normal Distribution.

TABLE C.6

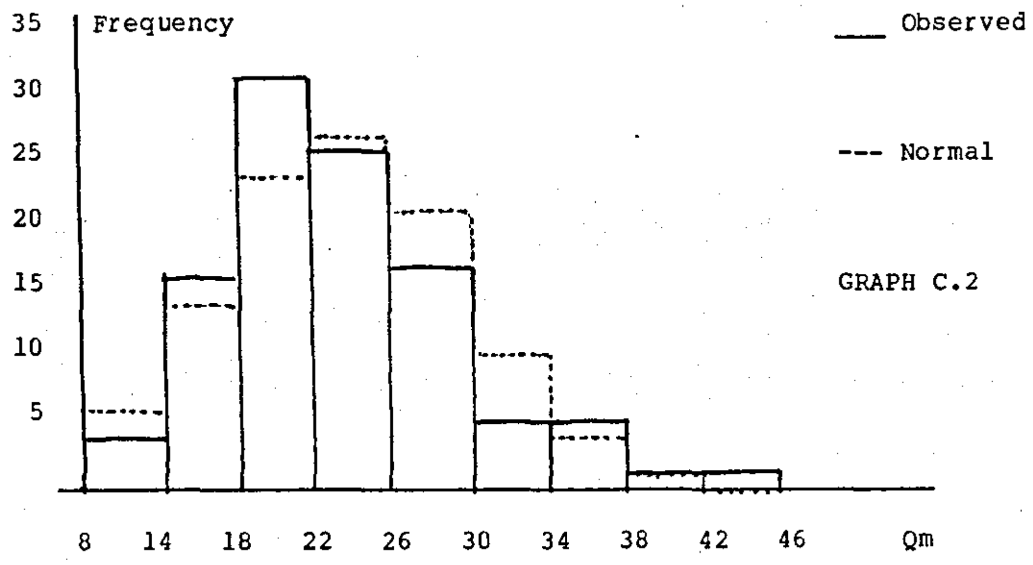
Class	Frequency (Simulation)	Expected Frequencies For a Normal Distribution
$\bar{Q}_m \leq 14$	3	5
$14 < \bar{Q}_m \leq 18$	15	13
$18 < \bar{Q}_m \leq 22$	31	23
$22 < \bar{Q}_m \leq 26$	25	26
$26 < \bar{Q}_m \leq 30$	16	20
$30 < \bar{Q}_m \leq 34$	4	9
$34 < \bar{Q}_m \leq 38$	4	3
$38 < \bar{Q}_m \leq 42$	1	1
$42 < \bar{Q}_m$	1	0
TOTAL	100	100

$\bar{Q}_m$ = 23.42

= 5/855

Calculated Chi-square = 8.5 (combining the last 3 classes)

With 4 degrees of freedom this is not significant at .05 level.



GRAPH C.2

Confidence Interval

$$\bar{\bar{Q}}_m \pm \frac{2\sigma}{\sqrt{n}}$$

is $23.42 \pm \frac{2(5.855)}{\sqrt{100}}$

or 22.25 to 24.59

The steady-state formula gives $\mu = 24$

```
@HKU*USER.GPSS
 GPSS 4.1   -11/12-19:33-(000)

       * FILE NO. BM1:KG5021.GPS  TO TABULATE BATCH MEANS
                    JOB
       *
       *
                    ORDER,X         BS
                    INITIAL         BS,2000
       INTERARRIVAL FUNCTION,EXP    RF$1,8,60
       SERVICE.TIME FUNCTION,EXP    RF$2,10,60
       WAIT.TIME    TABLE           X$MQ,12,2,20
       QT1          QTABLE          10,10,10,Q1
       *
    1               GENERATE        0           TIME(FN$INTERARRIVAL)
    2               QUEUE           Q1
    3  L1           HOLD            SERVER      TIME(FN$SERVICE.TIME)
    4               TERMINATE
       *
       *
    5               GENERATE         60           TIME(100)
    6               COMPARE          N$L1//X$BS EQ 0
    7               SAVEX            MQ,(QT$Q1*N$L1-X$NQ)/X$BS
    8               SAVEX            BATCH,X$BATCH+1
    9               PRINT            N$L1,QT$Q1,X$BATCH,X$MQ
   10               TABULATE         WAIT.TIME   GO TO(L2,L3)
   11  L2           COMPARE          X$BATCH EQ 100
   12               STOP
   13  L3           SAVEX            NQ,QT$Q1*N$L1
   14               TERMINATE
       *
       *
       START 300000
```

EXERCISE 4

598 days were simulated. The daily means $\bar{Q}_i$ are tabulated below in Table C.7.

Obviously the distribution is not Normal.

The days were divided into batches of 40 and the mean for each 40 ($\bar{\bar{Q}}$) obtained. These are shown in Table C.8, also not Normal.

TABLE C.7

CLASS	FREQUENCY
$\bar{Q}_i \leq 15$	285
$15 < \bar{Q}_i \leq 30$	175
$30 < \bar{Q}_i \leq 45$	79
$45 < \bar{Q}_i \leq 60$	26
$60 < \bar{Q}_i \leq 75$	18
$75 < \bar{Q}_i \leq 90$	7
$90 < \bar{Q}_i$	8
	598

TABLE C.8

CLASS	FREQUENCY
$19 < \bar{\bar{Q}} \leq 21$	7
$21 < \bar{\bar{Q}} \leq 23$	2
$23 < \bar{\bar{Q}} \leq 25$	3
$25 < \bar{\bar{Q}} \leq 27$	0
$27 < \bar{\bar{Q}} \leq 29$	2

The overall mean was 22.4. If one assumes that this (the mean of 598 daily means) is Normally distributed a 95% confidence interval for the daily mean is

$$22.4 \pm \frac{2(21.034)}{\sqrt{598}} \qquad (\sigma = 21.034)$$

or $\quad 22.4 \pm 1.7$

i.e. $\quad$ 20.7 to 24.1

```
@HKU*USER.GPSS
 GPSS 4.1    -11/12-19:58-(000)

        * FILE NO. BM1:KG5027.GPS  TO TABULATE DAILY MEANS
         JOB
        *
        *
        INTERARRIVAL FUNCTION, EXP   RF$1,8,60
        SERVICE.TIME FUNCTION, EXP   RF$2,10,60
        WAIT.TIME    TABLE           X$MQ,5,5,20
        N.BATCHES    TABLE           X$BT/40,15,2,20
        QT1          QTABLE          10,10,10,Q1
        *
        *
        *
  1                  GENERATE        0             TIME(FN$INTERARRIVAL)
  2                  COMPARE         X$TIMER LT 480
  3                  SAVEX           NO,X$NO+1
  4                  QUEUE           Q1
  5 L1               HOLD            SERVER        TIME(FN$SERVICE.TIME)
  6                  TERMINATE
        *
        *
  7                  GENERATE        0              TIME(1)
  8                  SAVEX           TIMER,X$TIMER+1    GO TO(L2,L3)
  9 L2               COMPARE         X$TIMER EQ 480
 10                  GATE            NU,SERVER
 11                  SAVEX           DAY,X$DAY+1
 12                  SAVEX           MQ,(QT$Q1*N$L1-X$NQ)/50
 13                  SAVEX           BT,X$BT+X$MQ
 14                  TABULATE        WAIT.TIME    GO TO(L4,L5)
 15 L4               COMPARE         X$DAY//40 EQ 0
 16                  TABULATE        N.BATCHES    GO TO(L6,L7)
 17 L6               COMPARE         X$DAY GE 2400
 18                  STOP
 19 L7               SAVEX           BT,0
 20 L5               SAVEX           NQ,QT$Q1*N$L1
 21                  SAVEX           TIMER,0
 22                  SAVEX           NO,0
 23 L3               TERMINATE
        *
        *
        START 60000,300000
```

APPENDIX D

MINI GPSS MANUAL FOR UNITS 1-6 ONLY

CONTENTS

D.1 NOTATION

D.1.1 Arithmetic Symbols

+	denotes addition
-	denotes subtraction
*	denotes multiplication
/	denotes division
//	denotes remainder division i.e. only the remainder is obtained
**	denotes exponentiation

Examples $4 ** 3 = 4^3 = 64$

252 // 26 = 18 (when 252 is divided by 26 the remainder is 18)

The order of priority is FIRST exponentiation

SECOND division (both types) and multiplication

THIRD addition and subtraction

Consider the following expression:

20 * 30 + 4 / 5 ** 3 - 20 // 6

First priority ** so do 5 ** 3 first

expression becomes 20 * 30 + 4 / 125 - 20 // 6

Second priority /, // and *

expression becomes 600 + .032 - 2

Third priority + and -

expression becomes 598.032.

D.1.2 Names, Syntax, Coding

Symbolic names must contain one to twelve characters from the set 0-9, A-Z, and the period (.) i.e. a choice of 37 characters. The first must be alphabetic and spaces are not allowed. It is convenient to use the period in place of a space e.g. CHECKIN.TIME. None of the GPSS block type names can be used as symbolic names.

In a model, statement fields are separated by one or more blanks, items within a field are separated by commas. If one item in the midst of a list of optional items is to be omitted the commas must still be included, e.g. $n_1,,n_3,n_4$ (the optional n_2 has been omitted). Generally speaking blanks may not appear within fields. One exception is the word GOTO in the ROUTING field. It may be written GO TO. Also within the DATA field of a FUNCTION statement data pairs are separated by blanks.

If a statement cannot fit on one line it can be continued on further lines. Each continuation line must have a + sign in column 1.

If a statement has an asterisk in column 1 it is understood to be a comment only (for reading, in a printout) and is not interpreted as part of the program. If within a statement the symbol @ appears then what follows it is understood to be comment only, i.e. is not part of the statement.

D.1.3 Referencing Entities

Whenever in GPSS one refers to a numeric attribute of some entity such as the value of a FUNCTION or a VARIABLE one uses a special notation consisting of three parts.

(a) a one or two alphabetic character reference or mnemonic which defines the type of entity e.g. FN = function. The reference must be one of a set specified by the simulator. A few examples are given below.

(b) a dollar sign $ (mentally one can substitute the word "called")

(c) the entity name

GPSS entities such as FACILITIES or QUEUES have mnemonics which are used in numeric attribute references (see page 375).

	mnemonic	
CLOCK TIME	C	Value of clock time C$1
TRANSACTION	TN	TRANSACTION no. = TN$1
RANDOM NUMBER GENERATOR	RF	number from generator n = RF$n
FUNCTION	FN	
FACILITY	F	
QUEUE	Q	

D.1.4 Numeric Attributes

Many entities in GPSS such as QUEUES or FACILITIES have numerical values associated with them such as the size of a QUEUE, the identity number of a TRANSACTION etc. These values are maintained by the simulator and generally cannot be changed by the programmer but the values are available by using certain designated references. The only one to be used in the simple model is the following:

RF$n = next random number from generator n ($1 \leqslant n \leqslant 10$)

There are other numeric attributes which are set and manipulated by the programmer such as the following:

FN$name = value of the FUNCTION called "name"

One needs to distinguish carefully between the name of an entity and a "numeric attribute reference" of an entity. For example suppose that there is a FUNCTION called "DAY". If in a DATA position the name of the FUNCTION is required then one should write DAY. However if a numeric attribute reference is required i.e. the value of DAY then one must write FN$DAY.

D.1.5 Random Number Generators

(The first 25 numbers produced by each generator in integer format)

RI$1	RI$2	RI$3	RI$4	RI$5	RI$6	RI$7	RI$8	RI$9	RI$10
889	135	115	84	13	348	104	611	803	907
191	702	301	265	696	511	660	732	152	749
67	713	609	669	694	89	901	941	584	593
967	839	851	696	194	697	631	276	619	515
410	824	816	348	261	293	402	498	726	363
150	90	948	203	188	900	412	192	773	138
270	649	779	369	969	593	728	102	56	255
335	968	951	179	676	836	383	998	755	531
693	875	462	167	506	64	628	84	614	780
74	238	686	762	517	820	82	974	801	394
188	615	894	763	638	75	684	389	904	103
847	206	134	315	523	509	212	740	734	295
355	762	531	617	374	855	882	40	303	441
903	412	133	842	533	366	175	58	523	785
73	869	742	691	658	884	569	144	490	17
377	333	344	130	26	659	156	94	414	484
518	117	444	161	294	651	915	398	660	964
78	397	611	941	745	582	367	841	641	866
817	891	132	629	181	438	743	280	168	107
144	351	542	807	107	754	4	829	311	637
749	364	936	97	92	201	5	228	961	558
214	212	529	122	943	944	203	998	190	977
770	160	377	881	959	599	318	134	751	624
628	665	594	174	311	362	68	152	208	434
398	444	559	131	495	354	845	68	403	531

Note RI$n denotes a random number from generator n in integer form. A reference to RF$n would return a random number in the range 0 to 1 i.e. RF$n = RI$n / 1000

D.2 MODEL STATEMENTS OR BLOCKS

D.2.1 Description

A model statement or block consists in general of five sections or fields separated by spaces. They are as follows:

LABEL TYPE DATA TIME ROUTING

Some statements have fewer than five fields e.g. some do not require DATA or TIME fields. LABEL and ROUTING fields are generally optional. All must have a TYPE field. These five fields are described in detail on pages 108 - 110.

The following pages describe in alphabetical order all of the blocks to be used or referred to in the models of units 1 - 6. It is not an exhaustive list. GPSS contains many blocks which are not required or used here. They can be found in any GPSS manual.

For the blocks to be described, the function will be explained and the DATA format. TIME and ROUTING formats are the same for all blocks which require them (see page 109). Any item in the DATA field in square brackets is optional e.g. in the list

X,[Y,]Z   Y is optional.

Entry to a block may be conditional. A TRANSACTION cannot enter a SEIZE or HOLD block unless the corresponding FACILITY is unoccupied. In this case it will remain where it is (e.g. on a QUEUE) until entry is allowed or will attempt to go to an alternative destination if one was specified in the ROUTING FIELD.

D.2.2 List of Blocks

TYPE	Function of this Block	DATA
ADVANCE	The main function is to allow the passage of time i.e when a TRANSACTION enters this block it waits for the time period specified in the TIME field and then proceeds to the next block.	none
GENERATE	To generate TRANSACTIONS and feed them into the model at a specified rate. This block cannot be entered by a TRANSACTION. The TIME field specifies the interval between the successful departure of one TRANSACTION and the creation of the next i.e. the time of creation of the next TRANSACTION depends on the actual departure time (not generation time) of the present one.	Format: $n_1[,n_2, n_3]$ n_1 time at which first TRANSACTION is generated n_2 maximum number of TRANSACTIONS n_3 priority of TRANSACTIONS n_2 and n_3 are optional. if n_2 is not specified an unlimited number can be generated.

TYPE	Function of this Block	DATA
HOLD	To occupy a FACILITY	Format: X X is the name of the FACILITY
QUEUE	To enter an item or items onto a QUEUE for statistical purposes. The item(s) leaves the QUEUE when the TRAN-SACTION leaves this block. Thus a record is made of the time spent waiting for entry to the next block.	Format: X[,Y] X is the name of the QUEUE Y is the number of units to be entered. If omitted Y is understood to be 1
RELEASE	To release a FACILITY which had been occupied via a SEIZE (q.v.) block	Format: X X is the name of the FACILITY
SEIZE	To occupy a FACILITY, which will be released when the TRANSACTION subsequently enters a RELEASE (q.v.) block	Format: X X is the name of the FACILITY

TYPE	Function of this Block	DATA
TERMINATE	To remove a TRANSACTION from the model	None
TERMINATE,R	To remove a TRANSACTION from the model and to record it as part of the total number of terminations required before the simulation stops	None

D.2.3 Examples of blocks

ADVANCE TIME(FN$XYZ)

An entering TRANSACTION will wait for a time FN$XYZ before proceeding to the next block.

GENERATE 1,100,2 TIME(FN$SPACING)

Generate 100 TRANSACTIONS starting at time 1 with priority 2. The time between one successful departure and the next attempted departure from the block is got by getting a value of the FUNCTION called "SPACING". This is recalculated for each TRANSACTION.

HOLD TELEPHONE TIME(4)

The FACILITY called "TELEPHONE" is occupied for 4 time units. A TRANSACTION cannot enter this block if "TELEPHONE" is already occupied.

QUEUE LINE1

Enters one item onto the QUEUE called "LINE1". The item will be removed from the QUEUE when the TRANSACTION leaves the block.

RELEASE TELEPHONE

Releases the FACILITY called "TELEPHONE" which had been occupied as a result of a SEIZE block.

SEIZE TELEPHONE

Occupies the FACILITY called "TELEPHONE". A TRANSACTION cannot enter this block unless "TELEPHONE" is unoccupied.

D.3 DATA DEFINITION STATEMENTS

D.3.1 Description

Data definition statements precede the model proper. Their purpose is to specify numeric values and formats to be used in the model.

In general a data definition statement consists of three sections or fields separated by one or more blanks, as follows:

```
NAME      TYPE      DATA
```

NAME:	This is the analyst assigned name for the entity to which the statement refers (see page 373 on allowable names)
TYPE:	This specifies the type of data definition statement. Examples are CAPACITY, FUNCTION,* INITIAL, MATRIX, QTABLE, TABLE, VARIABLE. TYPE will be one such word. The only data definition statement to be described here is "FUNCTION". See the official manual for a complete list if required.
DATA:	This is the data which is being assigned to the named entity. Its format depends on which of the many statements is being used.

* There are three variations of the FUNCTION statement used viz., FUNCTION FUNCTION,C FUNCTION,EXP

D.3.2 List of Data Definition Statements

TYPE OF DATA DEFINITION STATEMENT	WHAT THIS STATEMENT DOES	DATA REQUIRED
FUNCTION	Specifies the relationship between two variables in the form of a table; gives the values of the dependent variable y which correspond to the specified values of the independent variable x. It is a discrete or step-valued function i.e. if y_1 corresponds to x_1 and y_2 corresponds to x_2 then for any $x \leq x_1$ y equals y_1 and for any x in the range $x_1 < x \leqslant x_2$ y equals y_2 (See Unit 5)	Format: $x, x_1, y_1 \ \ x_2, y_2 \ \ x_3, y_3 \ldots$ x This defines what is to be used as the independent variable. It can be a numeric attribute (see page 375) or an arithmetic expression combining numeric attributes x_1 A numeric value of x y_1 The corresponding value of the dependent variable x_2 Another numeric value of x y_2 The corresponding value of the dependent variable . As many pairs as required . . . Note: Members of a pair are separated by a comma, pairs are separated by a space (or spaces). This is an exception to the rule that spaces may not appear within fields.

TYPE OF DATA DEFINITION STATEMENT	WHAT THIS STATEMENT DOES	DATA REQUIRED
FUNCTION,C	The same as FUNCTION but this is a continuous or linearly interpolated function i.e. if x is between x_1 and x_2 then $y = y_1+(\frac{x-x_1}{x_2-x_1})(y_2-y_1)$	as for FUNCTION
FUNCTION,EXP	This is a special simulator defined function. The relationship between the dependent and independent variables is that of the negative exponential distribution with mean $n_2 \div n_1$. The independent variable is a uniform random number	Format: RF\$n,n_1,n_2 RF\$n n is the number of one of the ten random number generators i.e. $1 \leqslant n \leqslant 10$. n_1 is an integer n_2 is an integer such that mean rate equals $n_2 \div n_1$

D.3.3 Examples of Data Definition Statements

```
1.  ARRIVAL.TIME   FUNCTION,C      RF$1,0,0  .3,600  .65,1200  1,2100

2.  INTERARRIVAL   FUNCTION,EXP    RF$4,2,5
```

1. This defines "ARRIVAL.TIME" as a dependent variable (y) which depends on the value of a random number generated by generator number 1 (RF$1). The relationship is as follows:

TABLE D.1

Value of random number	Value of "ARRIVAL.TIME"
0	0
.3	600
.65	1200
1.00	2100

Since the function is a continuous function if the random number generated falls between two of the table values the "ARRIVAL.TIME' will be got by interpolation between the corresponding "ARRIVAL.TIMES".

e.g. if random number = 0.4 which lies between 0.3 and 0.65 then "ARRIVAL.TIME' is got by interpolating between 600 and 1200 as follows:

$$\text{ARRIVAL.TIME} = 600 + \left(\frac{.4 - .3}{.65 - .3}\right)(1200 - 600) = 771.43$$

If it had been a FUNCTION and not a FUNCTION,C statement the value of ARRIVAL.TIME corresponding to random number 0.4 would be 1200

2. This defines INTERARRIVAL as a variable with a negative exponential distribution. Its value will depend on the value returned by random number generator number 4. Its mean value will be 2.5 ($n_2 \div n_1 = 5 \div 2 = 2.5$).

Those with a mathematical inclination might like to verify

that INTERARRIVAL = $-\frac{n_2}{n_1} \log_e r$ where r is the random number.

D.4 CONTROL STATEMENTS AND MODEL OPERATION

D.4.1 Description

Control statements are used to tell the simulator how one wants the simulation run e.g. how long to continue simulation, whether one wants re-runs and if so under what conditions. Three different control statements will be required viz.,

JOB, START, END

Control statements do not have label fields.

READING THE MODEL

When the simulator has been "called" via the @GPSS statement the first statement it should encounter is a JOB statement. This indicates the beginning of a job. The simulator then reads the subsequent statements and checks for errors. If there are no errors it is ready to start.

STARTING SIMULATION

If there are no errors the simulator will start when and if it encounters a valid START statement.

ENDING A SIMULATION

Simulation will end when the conditions for ending specified on the START statement are satisfied or when an error causes it to stop. An output report is printed. The simulator is now ready to read another START statement. In the models of these chapters only one START statement will ever be used.

ENDING A SIMULATION JOB

Simulation will end when an END control statement is encountered. This should be the last statement.

D.4.2 List of Control Statements

STATEMENT TYPE	FUNCTION	DATA
END	To indicate that the job is finished i.e. this should be the final statement	None
JOB	To indicate that what follows is a job i.e. this should be the first statement (Note: some compilers at least, apparently require that JOB should not commence in column one - this is not true of control statements in general which may start in column 1 as there is no LABEL field)	None
START	To instruct the simulator to start simulating and to give instructions on when to stop the simulation. Stopping can be on the basis of the number of TRANSACTION terminations or the time. If both are specified (n_1 and n_2) then when the first is satisfied simulation stops. By specifying either or both n_3, n_4 intermediate printouts may be obtained	Format $n_1[,n_2][,n_3][,n_4]$ n_1 the number of TERMINATIONS to be recorded before stopping n_2 the simulator time at which to stop n_3 the number of terminations between intermediate printouts n_4 the simulator time between printouts

D.4.3 Example of Start Statement

```
START          100,500,,10
```

The above statement causes simulation to stop when the first of the following two conditions is true

(a) number of recorded TERMINATIONS equals 100

(b) simulation time equals 500.

Printouts will be provided after every 10 time units.

APPENDIX E

GPSS MINI-MANUAL

CONTENTS

E.1 NOTATION

E.1.1 Arithmetic Symbols

+	denotes addition
-	denotes subtraction
*	denotes multiplication
/	denotes division
//	denotes remainder division i.e. only the remainder is obtained
**	denotes exponentiation

Examples 4 ** 3 = 4^3 = 64

252 // 26 = 18 (when 252 is divided by 26 the remainder is 18)

The order of priority is FIRST exponentiation

SECOND division (both types) and multiplication

THIRD addition and subtraction

Consider the following expression:

20 * 30 + 4 / 5 ** 3 - 20 // 6

First priority ** so do 5 ** 3 first

expression becomes 20 * 30 + 4 / 125 - 20 // 6

Second priority /, // and *

expression becomes 600 + .032 - 2

Third priority + and -

expression becomes 598.032.

E.1.2 Names, Syntax and Coding

Symbolic names such as labels or the names of FACILITIES etc. must contain one to twelve characters from the set 0-9, A-Z, and the period (.) i.e. a choice of 37 characters. The first must be alphabetic and spaces are not allowed. It is convenient to use the period in place of a space e.g. CHECKIN.TIME. None of the GPSS block type names can be used as symbolic names.

In a model, statement fields are separated by one or more blanks, items within a field are separated by commas. If one item in the midst of a list of optional items is to be omitted the commas must still be included, e.g. $n_1,,n_3,n_4$ (the optional n_2 has been omitted). Generally speaking blanks may not appear within fields. One exception is the word GOTO in the ROUTING field. It may be written GO TO. Also within the DATA field of a FUNCTION statement data pairs are separated by blanks.

If a statement cannot fit on one line it can be continued on further lines. Each continuation line must have a + sign in column 1.

If a statement has an asterisk in column 1 it is understood to be a comment only (for reading, in a printout) and is not interpreted as part of the programme. If within a statement the symbol @ appears then what follows it is understood to be comment only, i.e. is not part of the statement.

E.1.3 Referencing Entities

Whenever in GPSS one refers to a numeric attribute of some entity such as the value of a FUNCTION or a VARIABLE one uses a special notation consisting of three parts.

(a) a one or two alphabetic character reference or mnemonic which defines the type of entity e.g. V = variable, FN = function. The reference must be one of a set specified by the simulator. A partial list is provided on page 394.

(b) a dollar sign $ (mentally one can substitute the word "called")

(c) the entity name

For example if in a GPSS programme the following expression appeared

```
V$SPEED * FN$TIME
```

the simulator would multiply the current value of the "VARIABLE called SPEED" by the current value of the "FUNCTION called TIME" (this FUNCTION value would be evaluated as described in Unit 5) and substitute this numerical value for the above expression. Note that * denotes multiplication.

Indirect Referencing

A name may be stored in a PARAMETER via an ASSIGN block (q.v.). One may later refer to this name indirectly by specifying the parameter which contains it. An indirect reference must be preceded by an asterisk.

Example

ASSIGN	PAR1, BAY1	This stores the name BAY1 in the PARAMETER called PAR1
.		
.		
.		
HOLD	*PAR1	This indicates that the entering TRANSACTION should "HOLD" the FACILITY named in the PARAMETER called PAR1. For different TRANSACTIONS PAR1 could contain different names. The asterisk indicates that the word (PAR1, in this case) is not the name of the FACILITY but an address where the name can be found.

E.1.4 Mnemonics for Entities

GPSS entities such as FACILITIES or QUEUES have mnemonics which are used in numeric attribute references (see page 395) or for indexing (see below). The ones required are listed below:

	mnemonic	
CLOCK TIME	C	Value of clock time = C$1
TRANSACTION	TN	TRANSACTION no. = TN$1
MARK TIME	M	Value of MARK time = M$1
RANDOM NUMBER GENERATOR	RF	number from generator n = RF$n
FACILITY	F	
STORAGE	S	
QUEUE	Q	
PARAMETER	P	
TABLE	T	
SAVEX	X	
FUNCTION	FN	
VARIABLE	V	

All of the entities in the above list from FACILITY to VARIABLE may be indexed i.e. if one has three FACILITIES one can give them three "names" or simply refer to them as F(1), F(2), F(3). If an entity is to be referenced using indexing this fact must be declared on an ORDER statement (q.v.). It is also possible to do both i.e. reference an entity either by name or by indexing provided this is specified on an ORDER statement

e.g. one can make VAN.RATE synonymous with FN(1)

and LORRY.RATE synonymous with FN(2)

Indexing is convenient if one wants to choose a FACILITY or FUNCTION (or any other entity) on the basis of a PARAMETER or SAVEX value e.g. HOLD F(X$TYPE). This HOLDS a FACILITY, which FACILITY will depend on the value of the SAVEX called TYPE.

In fact any entity may have several synonymous names provided this is declared on an ORDER statement.

E.1.5 Numeric Attributes

Many entities in GPSS such as QUEUES, FACILITIES, STORAGES, TRANSACTIONS have numerical values associated with them such as the contents of a STORAGE, the size of a QUEUE, the identity number of a TRANSACTION etc. These values are maintained by the simulator and generally cannot be changed by the programmer but the values are available by using certain designated references. The ones to be used are lised below. They can be used in arithmetic expressions.

C$1	= simulator clock time
TN$1	= TRANSACTION identity number
FU$name	= 1 if FACILITY called "name" is occupied, otherwise = 0
S$name	= contents of STORAGE called "name"
R$name	= remainder of STORAGE called "name" (capacity-contents)
Q$name	= contents of QUEUE called name
QT$name	= average queueing time on QUEUE called "name"
M$1	= a TRANSACTION'S transit time since it entered the model or entered a MARK block (see page 408)
N$label	= number of TRANSACTIONS which have entered the block with the indicated label
RF$n	= next random number from generator n ($1 \leqslant n \leqslant 10$)

There are other numeric attributes which are set and manipulated by the programmer such as those below.

P$name	= numeric contents of the TRANSACTION PARAMETER called "name"
X$name	= numeric value of the SAVEX called "name" (see page 398)
MX$name(i,j)	= numeric value of the (i,j) element of SAVEX matrix (MSAVEX) called "name" (see page 399)
V$name	= numeric value of the VARIABLE called "name". (see page 397)
X$X(i)	= numeric value of the i^{th} element of SAVEX ARRAY X (see page 399)

One needs to distinguish carefully between the name of an entity and a "numeric attribute reference" of an entity. For example suppose that there is a VARIABLE called "DAY". If in a DATA position the name of the VARIABLE is required then one should write DAY. However if a numeric attribute reference is required i.e. the value of DAY then one must write V$DAY.

On pages 415 - 419 is a list of examples of the blocks described on pages 404 - 414 .

E.1.6 VARIABLE Entity

A VARIABLE in GPSS is a mathematical expression which has a single name by which it can be referenced. For instance if an expression such as

FN$ARRIVAL.TIME*(C$1/750-C$1/810+1) occurs several times in a model it can be given a VARIABLE name on a VARIABLE definition statement, a type of DATA definition statement. The format would be as follows:

```
ARR.T VARIABLE FN$ARRIVAL.TIME*(C$1/750-C$1/810+1)
```

This states that the VARIABLE called ARR.T is synonymous with the given arithmetic expression. If anywhere in the model reference is made to V$ARR.T the numeric value of the expression, calculated at the time the reference is made, is substituted for V$ARR.T. Note that it is not possible to directly assign a value to ARR.T in the model. It should be seen merely as a short-hand notation for the arithmetic expression.

The above format defines a decimal variable. It is also possible to define an integer variable as follows:

```
name VARIABLE,I arithmetic expression
```

In this case the arithmetic expression is evaluated as follows:

(a) each component is truncated to an integer (0 if it is negative) before any operations take place

(b) the result of any division is truncated to an integer before any further operations take place

BOOLEAN VARIABLE

A Boolean Variable has a value of either 1 or 0. It is 1 if the statement defining it is true when the reference is made. It is zero if the statement defining it is false when the reference is made. See BVARIABLE on page 426.

E.1.7 SAVEX Entity

A SAVEX is a storage location available to the programmer. Numeric values are assigned to the SAVEX by means of a SAVEX block with the following format

```
SAVEX name, arithmetic expression
```

The value of the arithmetic expression is stored in the SAVEX called name.

e.g. `SAVEX KOUNT, X$KOUNT + 1`

The value of a SAVEX is got by reference to X$name so whenever a TRANSACTION goes through this block the value of "KOUNT" is increased by 1. A SAVEX is somewhat similar to a TRANSACTION PARAMETER however a PARAMETER is part of the record of a particular TRANSACTION and is carried with it. A SAVEX is external. Compare the following two statements

```
        ASSIGN KOUNT, P$KOUNT + 1
and     SAVEX  KOUNT, X$KOUNT + 1
```

If a TRANSACTION goes through the ASSIGN block the PARAMETER called "KOUNT" of that particular TRANSACTION is increased by 1. If another TRANSACTION goes through the same block the PARAMETER called "KOUNT" of that second TRANSACTION is increased by one. If the two TRANSACTIONS had gone through the SAVEX block then two would have been added to the SAVEX called "KOUNT". One must decide carefully which one wants to use. For instance suppose one wants to record the arrival time of a TRANSACTION and one does this as follows:

```
GENERATE .......

SAVEX ARRIVAL,C$1
```

Then at some subsequent stage one wants to recall the arrival time by reference to X$ARRIVAL. The problem is that since the TRANSACTION assigned its arrival time to the SAVEX called "ARRIVAL" another or several other TRANSACTIONS could have been generated and they would have changed the value of "ARRIVAL". One should have used ASSIGN ARRIVAL,C$1. Then the PARAMETER "ARRIVAL" specific to this TRANSACTION could not be changed by another TRANSACTION.

E.1.8 <u>SAVEX Arrays</u>

It is possible to have one singly dimensioned SAVEX i.e. with a single subscript. It must be specified on an ORDER control statement. The ORDER statment has other functions which will be encountered later but for this purpose the format is

```
ORDER,X n
```

The above defines X as an n-dimensional array. Note that only one singly indexed array called X is available.

It is also possible to have any number of doubly indexed SAVEXES called MSAVEX (the number is of course subject to computer capacity).

The value is assigned by means of a MSAVEX block as follows:

```
MSAVEX name(i,j), arithmetic expression
```

e.g. MSAVEX VEHICLES(P$TYPE,P$OWNER), MX$VEHICLE(P$TYPE,P$OWNER) + 1
The above matrix or table called VEHICLES might be used to record the total number of vehicles by type and owner.

The dimension of an MSAVEX (matrix) must be specified on a MATRIX data definition statement with format

MATRIX name(m_1,n_1), name(m_2,n_2) .....

A number of matrices may be dimensioned on the same statement.

It is possible using an INITIAL data definition statement to set SAVEX or MSAVEX values before the simulation commences. The INITIAL statement must follow the MATRIX statement. The format for an INITIAL statement is as follows:

```
INITIAL item/item/item/item/ .... /item
```

Each item specifies a value or set of values. Possible formats for "item" are as follows (assume A = name of a SAVEX, B = name of a doubly indexed MSAVEX)

```
A, constant

X(i), constant

X(i - j), constant

X(i - j), constant, constant ...   (number of constants must
                                     equal j-i+1)

B(i,j), constant

B(i-j, m-n), constant

B(i-j, m-n), constant 1, constant 2 ... (number of constants
                                          = (j-i-1)(n-m+1))
```

Example

```
INITIAL SIZE,4/X(1),3/X(2-5),4/X(6-8),5,6,7
```

The SAVEX called SIZE is given a value 4.

The ARRAY X has the values 3,4,4,4,4,5,6,7 in positions 1 to 8.

E.1.9 Permissable Formats for an Index

An index can be either a numerically specified integer, a numeric attribute or a numeric attribute plus or minus an integer but nothing else.

Thus the following are acceptable

```
X(3)   MAT(3, 5)   X(P$I)   MAT(P$I+4,V$V1-2)
```

The following are NOT allowed

```
X(4*P$I)   x(20-P$I)   MAT(4,3*V$V1)
```

If one wished to calculate an index as above this would have to be done in a separate step

```
e.g.                ASSIGN J,4*P$I
        then refer to   X(P$J)
or                  SAVEX  J,4*P$I
        then refer to   X(X$J)
```

E.1.10 Random Number Generators

(The first 25 numbers produced by each generator in integer format)

RI$1	RI$2	RI$3	RI$4	RI$5	RI$6	RI$7	RI$8	RI$9	RI$10
889	135	115	84	13	348	104	611	803	907
191	702	301	265	696	511	660	732	152	749
67	713	609	669	694	89	901	941	584	593
967	839	851	696	194	697	631	276	619	515
410	824	816	348	261	293	402	498	726	363
150	90	948	203	188	900	412	192	773	138
270	649	779	369	969	593	728	102	56	255
335	968	951	179	676	836	383	998	755	531
693	875	462	167	506	64	628	84	614	780
74	238	686	762	517	820	82	974	801	894
188	615	894	763	638	75	684	389	904	103
847	206	134	315	523	509	212	740	734	295
355	762	531	617	374	855	882	40	303	441
903	412	133	842	533	366	175	58	523	785
73	869	742	691	658	884	569	144	490	17
377	333	344	130	26	659	156	94	414	484
518	117	444	161	294	651	915	398	660	964
78	397	611	941	745	582	367	841	641	866
817	891	132	629	181	438	743	280	168	107
144	351	542	807	107	754	4	829	311	637
749	364	936	97	92	201	5	228	961	558
214	212	529	122	943	944	203	998	190	977
770	160	377	881	959	599	318	134	751	624
628	665	594	174	311	362	68	152	208	434
398	444	559	131	495	354	845	68	403	531

Note RI$n denotes a random number from generator n in integer form. A reference to RF$n would return a random number in the range 0 to 1 i.e. RF$n = RI$n 1000

E.2.1 MODEL STATEMENTS or BLOCKS

E.2.1 Description

A model statement or block consists in general of five sections or fields separated by spaces. They are as follows:

LABEL   TYPE   DATA   TIME   ROUTING

Some statements have fewer than five fields e.g. some do not require DATA or TIME fields. LABEL and ROUTING fields are generally optional. All must have a TYPE field. These five fields are described in detail on pages 108 - 110.

The following pages describe in alphabetical order all of the blocks to be used or referred to in the models which will be constructed. It is not an exhaustive list. GPSS contains some additional blocks which are not required or used here. They can be found in any GPSS manual.

For the blocks to be described, the function will be explained and the DATA format. TIME and ROUTING formats are the same for all blocks which require them (see page 109). Any item in the DATA field in square brackets is optional e.g. in the list

X,[Y,]Z   Y is optional.

Entry to a block may be conditional. A TRANSACTION cannot enter a SEIZE, HOLD, ENTER, STORE block if the corresponding FACILITY or STORAGE is occupied. In this event it will remain where it is (e.g. on a QUEUE) until entry is allowed or will attempt to go to an alternative destination if one was specified in the ROUTING field.

E.2.2 List of Blocks

TYPE	Function of this Block	DATA
ADVANCE	The main function is to allow the passage of time i.e when a TRAN-SACTION enters this block it waits for the time period specified in the TIME field and then proceeds to the next block.	none
ASSIGN	Assigns data to the TRANSACTION parameter named in the DATA field	Format: X, Y X is the name of the parameter Y is the data to be assigned. it may be a numeric attribute reference or it may be a name

TYPE	Function of this Block	DATA
COMPARE	To allow a choice of routing depending on the relationship between two quantities. If the relationship specified is true a TRANSACTION may enter the COMPARE block and proceed normally. If it is not true the TRANSACTION will remain where it is until the relationship is true, or it will proceed to an alternative destination if one is specified in its current block (see p.109 on Routing Information	Format: X Y Z X is one quantity Z is another quantity X and Z may be numeric attributes or arithmetic expressions Y is the relationship to be satisfied. It must be one of the following L or LT Less than LE Less than or equal to E or EQ Equal to NE Note equal to GE Greater than or equal to G or GT Greater than Note that X,Y,Z are separated by spaces not commas
ENTER	To enter a unit or units into a STORAGE where they will remain unless removed by a TRANSACTION entering a LEAVE block	Format: X[,Y] X is the name of the STORAGE Y is the number of units to be entered. Y is optional. If omitted it is assumed to be 1. That will very often be the case.

TYPE	Function of this Block	DATA
GATE	To check the status of a FACILITY or STORAGE.	Format: X,Y X is a mnemonic specifying the condition to be checked. It must be one of the following SE STORAGE empty SNE STORAGE not empty SF STORAGE full SNF STORAGE not full U FACILTIY in use NU FACILITY not in use I FACILITY interrupted NI FACILTIY not interrupted Y the name of the STORAGE or FACILITY
GENERATE	To generate TRANSACTIONS and feed them into the model at a specified rate. This block cannot be entered by a TRANSACTION. The TIME field specifies the interval between the successful departure of	Format: $n_1[,n_2, n_3]$ n_1 time at which first TRANSACTION is generated n_2 maximum number of TRANSACTIONS n_3 priority of TRANSACTIONS

TYPE	Function of this Block	DATA
	one TRANSACTION and the creation of the next i.e. the time of creation of the next TRANSACTION depends on the actual departure time (not generation time) of the present one	n_2 and n_3 are optional. if n_2 is not specified an unlimited number can be generated.
HOLD	To occupy a FACILITY	Format: X X is the name of the FACILITY
INQUEUE	To enter an item or items onto a QUEUE for statistical purposes. The item(s) will not leave the QUEUE until the TRANSACTION reaches an OUTQUEUE block. This distinguishes an INQUEUE block from a QUEUE (q.v.) block.	Format: X,Y[,Z] X is the name of the QUEUE Y is the name of a TRANSACTION PARAMETER which will record the time of entry onto the QUEUE Z is optional. If not specified it is understood to be 1. It is the number of items to be entered onto the queue when a TRANSACTION enters this block.

TYPE	Function of this Block	DATA
INTERRUPT	To occupy a FACILITY with a higher priority than a SEIZE or HOLD block even at the expense of a TRANSACTION already occupying a FACILITY as a result of a SEIZE or HOLD	Format: X X is the name of the FACILITY
LEAVE	To remove a unit or units from a STORAGE. This is the opposite to an ENTER block	Format: X[,Y] X is the name of the STORAGE Y is the number of units to be removed. If omitted Y is understood to be 1.
MARK	To start a timer, rather like pressing a button on a stopwatch. A future reference to either M$1 or to MP$X, (if the X option on the data has been used, X being the name specified in the DATA field) will return the elapsed time since the MARK was set.	Format: [X] X is a PARAMETER name which will record the time at which the MARK was set. If omitted the time is recorded in M$1 only. The advantage of specifying a name is that several different time records can be kept in different PARAMETERS whereas there is only M$1.

TYPE	Function of this Block	DATA
MSAVEX	To store a number in a SAVEX matrix element	Format: $X(n_1,n_2),Y$ X is the name of the SAVEX matrix n_1 is the row } position of n_2 is the column } element Y is the number to be stored in $X(n_1,n_2)$ and may be a numeric attribute or an arithmetic expression.
OUTQUEUE	To remove units from a QUEUE	Format: X,Y(,Z) X is the QUEUE name Y is the name of the TRANSACTION PARAMETER which contains a record of the time at which the unit(s) entered the QUEUE as specified on the INQUEUE (q.v.) block Z is the number of units to be removed. If omitted it is understood to be 1.

TYPE	Function of this Block	DATA
PRINT	To print values of variables during the simulation. Five values per line will be printed	Format: X_1, X_2 ... X_1, X_2 ... are numeric attribute references or arithmetic expression the values of which are to be printed whenever a TRANSACTION enters the block.
PRINT,L	To print whole or partial arrays (indexed) including SAVEX matrices. Matrices are printed row by row, thus if A is a 6 x 3 matrix and DATA is $n_1=1$, $n_2=1$, $n_3=7$, $n_4=1$ the elements to be printed (5 per line) will be A(1,1), A(1,2) A(1,3), A(2,1), A(2,2) A(2,3), A(3,1)	Format: $X(n_1),n_3[,n_4]$ or $X(n_1,n_2),n_3[,n_4]$ X is a numeric attribute of the indexed array (singly or doubly indexed) n_1 (or(n_1,n_2) for a matrix) specifies the first element to be printed n_3 is the index of the last element to be printed (in the case of a matrix it is the number of values to be printed) n_4 is the increment and is assumed to be 1 if omitted (i.e. if $n_4=2$ every second element will be printed)

TYPE	Function of this Block	DATA
PRIORITY	To change the priority of an entering TRANSACTION	Format : X X is an integer (the priority)
QUEUE	To enter an item or items onto a QUEUE for statistical purposes. The item(s) leaves the QUEUE when the TRANSACTION leaves this block. Thus normally entry to the next block might require waiting some time.	Format: X[,Y] X is the name of the QUEUE Y is the number of units to be entered. If omitted Y is understood to be 1
RELEASE	To release a FACILITY which had been occupied via a SEIZE (q.v.) block	Format: X X is the name of the FACILITY
SAVEX	To store a value in a SAVEX location	Format: X, Y X is the name of the SAVEX Y is the value to be stored and may be a numeric attribute or an arithmetic expression
SEIZE	To occupy a FACILITY, which will be released when the TRANSACTION subsequently enters a RELEASE (q.v.) block	Format: X X is the name of the FACILITY

TYPE	Function of this Block	DATA
STOP	To stop the simulation when a TRANSACTION enters this block.	Format: [X] X is a number which will be printed when the simulation stops as a result of a TRANSACTION entering this block. This is to distinguish between different STOP blocks.
STORE	To put units into STORAGE. They are removed when the TRAN-SACTION leaves the block.	Format: X[,Y] X is the name of the STORAGE Y is the number of units to be entered. It is assumed to be 1 if omitted.
TABULATE	To make an entry in the named TABLE when a TRANSACTION enters this block (see TABLE data definition statement).	Format: X[,Y] X is the name of the TABLE Y is the number of entries to be made. It is understood to be 1 if omitted.
TERMINATE	To remove a TRANSACTION from the model.	None

TYPE	Function of this Block	DATA
TERMINATE,R	To remove a TRANSACTION from the model and to record it as part of the total number of terminations required before the simulation stops	None
TRANSREAD	To read data from a specified PARAMETER of another TRANSACTION and enter it into a PARAMETER of the TRANSACTION in this block. See UNGUARD on the next page.	Format: X,Y,Z X is the name of the PARAMETER into which the data is to be put Y is a number which is the internal identification number of the TRANSACTION from which the data is to be obtained Z is the name of the PARAMETER containing the data

TYPE	Function of this Block	DATA
TRANSWRITE	To transfer data from a PARAMETER of the TRANSACTION in this block and enter it into a PARAMETER of another TRANSACTION. See UNGUARD below.	Format: X,Y,Z X is the name of the PARAMETER containing the data to be transfered Y is a number which is the internal identification number of the TRANSACTION to which Z belongs Z is the name of the PARAMETER into which the data is to be put.
UNGUARD	A TRANSACTION cannot have data read from it or written onto it in TRANSREAD or TRANSWRITE blocks unless it has first gone through an UNGUARD block. This block frees the TRANSACTION for writing onto or reading from.	None

E.2.3 Examples of Blocks

ASSIGN	T1,C$1 The current clock time (C$1) is stored in the PARAMETER called T1 of the entering TRANSACTION.
ASSIGN	TYPE, VAN The name "VAN" is stored in the PARAMETER called "TYPE".
COMPARE	C$1 LT 2000 Entry to this block is allowed only if the clock time (C$1) is less than 2000.
ENTER	BASKET,2 Enter two units into the STORAGE called "BASKET". A TRANSACTION cannot enter this block unless the maximum capacity of "BASKET" exceeds current contents by at least 2.
GATE	SE,BUILDING Entry to this block is allowed only if the STORAGE called "BUILDING" is empty.

GENERATE 1,100,2 TIME(FN$SPACING)

Generate 100 TRANSACTIONS starting at time 1 with priority 2. The time between one successful departure and the next attempted departure from the block is got by getting a value of the FUNCTION called "SPACING". This is recalculated for each TRANSACTION.

HOLD TELEPHONE TIME(4)

The FACILITY called "TELEPHONE" is occupied for 4 time units. A TRANSACTION cannot enter this block if "TELEPHONE" is already occupied.

INQUEUE LINE1,T2,2

Enter two units onto the QUEUE called LINE 1 and enter the current clock time into the PARAMETER called T2.

INTERRUPT SERVER TIME(2)

Occupy the FACILITY called "SERVER" for 2 time units. If "SERVER" is already occupied as a result of a HOLD or SEIZE block and whether or not there is a queue for SERVER the current occupant is put aside (interrupted) for 2 time units before resuming. This request for service has a higher priority than HOLD or SEIZE.

LEAVE BASKET,2

Remove 2 units from the STORAGE called "BASKET". If "BASKET" does not contain 2 units the program will terminate in error.

MARK DURATION.1

Will start a timer. The PARAMETER called "DURATION.1" will at any time contain the time which elapsed since the TRANSACTION enered the MARK block.

MSAVEX RECORD(1,4), 3*X$DAY+FN$XY

This stores the value of the expression "3*X$DAY+FN$XY" in the first row, fourth column of the matrix called "RECORD"

OUTQUEUE LINE1,T2,2

Removes two units from the QUEUE called LINE1 and updates QUEUE records. The PARAMETER "T2" contains the time at which the units entered the QUEUE (see INQUEUE example above).

PRINT TN$1, C$1, M$1, P$TYPE

This will print the TRANSACTION identity number (TN$1), the clock time (C$1), the time elapsed since the last MARK block (M$1) and the value of the PARAMETER called TYPE.

PRINT,L MX$RECORD(1,3),7,4 (assume "RECORD" is a 7 x 4 matrix)

Internally the simulator stores matrices row by row so the internal order of the elements is as follows:

(1,1) (1,2) (1,3) (1,4) (2,1) (2,2) (2,3) (2,4) (3,1) et

The above instruction specifies that the first item to be printed in MX$RECORD(1,3) and then 6 more (7 in all) at intervals of 4

i.e. (1,3) (2,3) (3,3) (4,3) (5,3)
(6,3) (7,3)

that is, it prints column 3, 5 elements per line.

PRIORITY 2		This sets the priority of an entering TRANSACTION equal to two
QUEUE	LINE1	Enters one item onto the QUEUE called "LINE1". The item will be removed from the QUEUE when the TRANSACTION leaves the block.
RELEASE	TELEPHONE	Releases the FACILITY called "TELEPHONE" which had been occupied as a result of a SEIZE block.
SAVEX	TIME,C$1	Stores the clock-time (C$1) in the SAVEX called "TIME".
SEIZE	TELEPHONE	Occupies the FACILITY called "TELEPHONE". A TRANSACTION cannot enter this block unless "TELEPHONE" is unoccupied.
STOP	99	The simulation will stop, the number 99 will be printed followed by the output report.
STORE	BIN,3 TIME(FN$XYZ)	Enters 3 items into the STORAGE called "BIN" where they will remain for a time FN$XYZ plus any time the TRANSACTION spends in the block subsequently while waiting for permission to enter the next block.

TABULATE TABLE.1

When a TRANSACTION enters this block an entry will be made in the TABLE called "TABLE.1" in accordance with the data definition statement for TABLE.1, i.e. it indicates the point in time for making an entry.

TERMINATE,R

Removes this TRANSACTION from the model and adds 1 to the termination count.

TRANSREAD TYPE.2, X$NO, TYPE.1

"NO" is a SAVEX which contains a TRANSACTION number which had presumably been previously stored in the SAVEX in a SAVEX block such as SAVEX NO,TN$1 (which stores the identity number TN$1 of the entering TRANSACTION in "NO").

The above block reads the contents of the PARAMETER called "TYPE.1" of the TRANSACTION whose identity number (TN$1) is stored in "NO" and writes this into the PARAMETER called TYPE.2 of the TRANSACTION in the TRANSREAD block.

(See TRANSWRITE below for a similar example)

TRANSWRITE TYPE.1,50,TYPE.2

This reads the information contained in the PARAMETER called "TYPE.1" of the TRANSACTION entering the block and writes it into the PARAMETER called "TYPE.2" of the TRANSACTION whose identity number is 50.

E.3 DATA DEFINITION STATEMENTS

E.3.1 Description

Data definition statements precede the model proper. Their purpose is to specify numeric values of FUNCTIONS to be used in the model, to dimension MSAVEX arrays or TABLES, to initialise values, to specify STORAGE capacities or to define VARIABLES.

In general a data definition statement consists of three sections or fields separated by one or more blanks, as follows:

```
TYPE          NAME          DATA
```

NAME:	This is the analyst assigned name for the entity to which the statement refers (see page 392 on allowable names)
TYPE:	This specifies the type of data definition statement. Seven types are required. They are CAPACITY, FUNCTION*, INITIAL, MATRIX, QTABLE, TABLE, VARIABLE. "TYPE" will be one of those seven words. GPSS does include other data definition statements not used here. See the official manual for a complete list if required.
DATA:	This is the data which is being assigned to the named entity. Its format depends on which of the seven statements is being used. They will be described in alphabetical order.

The statements INITIAL and MATRIX do not have a NAME field, the required names appear in the DATA field.

* There are three variations of the FUNCTION statement used

```
viz.,    FUNCTION       FUNCTION,C       FUNCTION,EXP
```

and three variations of the VARIABLE statement

```
viz.,    VARIABLE       VARIABLE,I       BVARIABLE
```

E.3.2 List of Data Definition Statements

TYPE OF DATA DEFINITION STATEMENT	WHAT THIS STATEMENT DOES	DATA REQUIRED
CAPACITY	Assigns a maximum capacity to a STORAGE	Format n n is an integer specifying the maximum capacity
FUNCTION	Specifies the relationship between two variables in the form of a table which gives the values of the dependent variable y which correspond to the specified values of the independent variable x. It is a discrete or step-valued function i.e. if y_1 corresponds to x_1 and y_2 corresponds to x_2 then for any $x \leqslant x_1$ y equals y_1 and for any x in the range $x_1 < x \leqslant x_2$ y equals y_2 (See Unit 5)	Format: $x, x_1,y_1 \ \ x_2,y_2 \ \ x_3,y_3 \ldots$ x This defines what is to be used as the independent variable. It can be a numeric attribute (see page 395) or an arithmetic expression combining numeric attributes x_1 A numeric value of x y_1 The corresponding value of the dependent variable x_2 Another numeric value of x y_2 The corresponding value of the dependent variable . As many pairs as required . . . Note: Members of a pair are separated by a comma, pairs are separated by a space (or spaces). This is an exception to the rule that spaces may not appear within fields.

TYPE OF DATA DEFINITION STATEMENT	WHAT THIS STATEMENT DOES	DATA REQUIRED
FUNCTION,C	The same as FUNCTION but this is a continuous or linearly interpolated function i.e. if x is between x_1 and x_2 then $y = y_1 + \left[\frac{x-x_1}{x_2-x_1}\right](y_2-y_1)$	as for FUNCTION
FUNCTION,EXP	This is a special simulator defined function. The relationship between the dependent and independent variable is that of the negative exponential distribution with mean $n_2 \div n_1$. The independent variable is a uniform random number	Format: RF\$n,n_1,n_2 RF\$n: n is the number of one of the ten random number generators i.e. $1 \leq n \leq 10$. n_1: n_1 is an integer n_2: n_2 is an integer such that the mean rate $= n_2 \div n_1$

TYPE OF DATA DEFINITION STATEMENT	WHAT THIS STATEMENT DOES	DATA REQUIRED
INITIAL	This sets initial values for a SAVEX, SAVEX array or SAVEX matrix. In the case of a matrix it must follow the MATRIX (q.v.) statement	Format: item/item/item ... where item can have one of following formats A, constant X(i), constant X(i-j), constant, X(i-j), constant, constant, ... (number of constants = j-i+1) B(i, m), constant B(i-j, m-n), constant B(i-j, m-n), constant, constant,.. (number of constants = (j-i+1)(n-m+1)) A is the name of a SAVEX B is the name of SAVEX matrix (MSAVEX)
MATRIX	Sets the dimensions of SAVEX matrices to be used in the model	Format A(n_1, n_2), B(n_2, n_3), .. A,B,C.. are the names of the matrices being dimensioned n_1, n_2 ... are integers specifying the required dimensions.

TYPE OF DATA DEFINITION STATEMENT	WHAT THIS STATEMENT DOES	DATA REQUIRED
QTABLE	Names and dimensions a TABLE to be associated with a specified queue or queues for the purpose of recording statistical information on that queue (or queues). Each time an item leaves one of the queues referred to (Q_1, Q_2 ...) the time it spent on that queue is noted and a relevant entry is made in the QTABLE for subsequent analysis	Format: $x_1, x_2, n, Q_1, Q_2, Q_3$, ... x_1 upper limit of first class in the TABLE x_2 class size n number of classes Q_1 name of a queue to be recorded Q_2 name of another queue to be recorded in the same TABLE Q_3 . . . In most cases only one queue would be specified.

TYPE OF DATA DEFINITION STATEMENT	WHAT THIS STATEMENT DOES	DATA REQUIRED
TABLE	Names and dimensions a TABLE to be used to record values of some variable	Format: x, x_1, x_2, n x the variable to be recorded x_1 upper limit of first class in the TABLE x_2 class size n number of classes Thus class one is $x \leqslant x_1$ class two is $x_1 < x \leqslant x_1+x_2$ class three is $x_1+x_2 < x \leqslant x_1+2x_2$ class n is $x_1+(n-2)x_2 < x \leqslant x_1+(n-1)x_2$ An extra class $x > x_1+(n-1)x_2$ is added to take any overflow.
VARIABLE	Associates a name with an arithmetic expression	Format: x x is an arithmetic expression
VARIABLE,I	Associates a name with an arithmetic expression which is calculated by truncating to an integer after each arithmetic operation	Format: x x is an arithmetic expression

TYPE OF DATA DEFINITION STATEMENT	WHAT THIS STATEMENT DOES	DATA REQUIRED
BVARIABLE	Specifies the conditions which if true result in the BVARIABLE having a value 1. Otherwise it is zero	an expression made up of numeric attribute references connected by relational operators such as GE, LT etc. (same as for COMPARE block, see p 405) and Boolean operators such as "AND" or "OR" (there are others but they are not required here). See example 10 page 427.

E.3.3 Examples of Data Definition Statements

```
 1. BUILDING       CAPACITY        9

 2. ARRIVAL.TIME   FUNCTION,C      RF$1,0,0  .3,600  .65,1200  1,2100

 3. INTERARRIVAL   FUNCTION,EXP    RF$4,2,5

 4.                MATRIX          DAILY.RECORD(8,2),Q.ANALYSIS(20,5)

 5.                INITIAL         DAY,1/X(1-4),3/X(5-7),4,5,6
    +                                /DAILY.RECORD(1-8,1-2),1

 6. CH.IN.TABLE    QTABLE          20,10,5,CHECKIN.LINE

 7. DELAYS         TABLE           X$N,5,2,6

 8. CLOCK.TIME     VARIABLE        ((8+(X$T+29-(X$T+29)//60)/60)*100
    +                                +(X$T+29)//60)//2400

 9. CLOCK.TIME     VARIABLE,I      ((8+(X$T+29/60)*100
    +                                +(X$T+29)//60)//2400

10. REMAIN         BVARIABLE       (C$1 LT 1200) AND
    +                              ((C$1 LT 1000) OR (S$SERVER EQ 0))
```

The above statements are explained on the following pages.

1. This defines "BUILDING" as the name of a STORAGE with capacity 9.

2. This defines "ARRIVAL.TIME" as a dependent variable (y) which depends on the value of a random number generated by generator number 1 (RF$1). The relationship is as follows:

TABLE E.1

Value of random number	Value of "ARRIVAL.TIME"
0	0
.3	600
.65	1200
1.00	2100

Since the function is a continuous function if the random number generated falls between two of the table values the "ARRIVAL.TIME" will be got by interpolation between the corresponding "ARRIVAL.TIMES".

e.g. if random number = 0.4 which lies between 0.3 and 0.65 then "ARRIVAL.TIME" is got by interpolating between 600 and 1200 as follows:

$$\text{ARRIVAL.TIME} = 600 + \left[\frac{.4 - .3}{.65 - .3}\right](1200 - 600) = 771.43$$

If it had been a FUNCTION and not a FUNCTION,C statement the value of ARRIVAL.TIME corresponding to random number 0.4 would be 1200.

3. This defines INTERARRIVAL as a variable with a negative exponential distribution. Its value will depend on the value returned by random number generator number 4. Its mean value will be 2.5 $(n_2 \div n_1 = 5 \div 2 = 2.5)$.

Those with a mathematical inclination might like to verify that INTERARRIVAL = $-\frac{n_2}{n_1} \log_e r$ where r is the random number.

4. This defines two matrices "DAILY.RECORD" and "Q.ANALYSIS" DAILY.RECORD is an 8 x 2 matrix (i.e. has 8 rows and 2 columns) Q.ANALYSIS has 20 rows and 5 columns.

5. This sets up the following initial values

SAVEX called DAY has value 1

The SAVEX array X has values 3,3,3,3,4,5,6

The SAVEX matrix called "DAILY.RECORD" has 1 in all positions (i.e. 8 rows and 2 columns)

6. This defines "CH.IN.TABLE" as a statistical TABLE associated with the QUEUE called "CHECKIN.LINE". TABLE has the following format

CH.IN. TABLE
TABLE E.2

QUEUE TIME (t)	frequency
$20 < t \leqslant 30$	
$30 < t \leqslant 40$	
$40 < t \leqslant 50$	
$50 < t \leqslant 60$	

t = time spent by an an item on the QUEUE called CHECK.IN

Whenever an item or items leaves "CHECKIN.LINE' an entry is made in the frequency column of the above table. On completion of the simulaiton this will be analysed and information such as cumulative frequencies, mean, standard deviation will be provided for t.

7. This defines "DELAYS" as a TABLE for recording values of the SAVEX called N. It will have the following format:

DELAYS
TABLE E.3

Value of SAVEX N	frequency
$N \leqslant 5$	
$5 < N \leqslant 7$	
$7 < N \leqslant 9$	
$9 < N \leqslant 11$	
$11 < N \leqslant 13$	
$13 < N \leqslant 15$	
$15 < N$	

8. This defines "CLOCK.TIME" as a VARIABLE which will have the value of the arithmetic expression specified as DATA. This value will depend on the value of the SAVEX called T at the time reference is made to "CLOCK.TIME".

In fact if T is the time measured in 1-minute units with 0830 as 1, 0831 as 2 etc. then verify that the expression converts T to clock time,

e.g.	if T = 538	CLOCK.TIME = 1812	i.e. 6.12 p.m.
	if T = 1857	CLOCK.TIME = 1526	i.e. 3.26 p.m.

9. This defines "CLOCK.TIME" as an integer VARIABLE with value calculated from the arithmetic expression specified as DATA. Verify that for any value of the the SAVEX called T it will return the same value as the VARIABLE defined in 8.

10. The BOOLEAN VARIABLE called "REMAIN" will have a value 1 whenever either of the following conditions are true

(a) Current clock time (C$1) is less than 1000

(b) Current clock time (C$1) is between 1000 and 1200 but the STORAGE called "SERVER' has zero contents.

A table will make this clear

TABLE E.4

Value of C$1	Contents of 'SERVER"	Value of "REMAIN"
$C\$1 < 1000$	irrelevant	1
$1000 \leq C\$1 < 1200$	zero	1
$1000 \leqslant C\$1 < 1200$	non-zero	0
$1200 \geqslant C\$1$	irrelevant	0

E.4 CONTROL STATEMENT AND MODEL OPERATION

E.4.1 Description

Control statements are used to tell the simulator how one wants the simulation run e.g. how long to continue simulation, whether one wants re-runs and if so under what conditions. Seven different control statements will be required viz.,

CLEAR, END, JOB, ORDER, RESET, SEED, START

They are described in alphabetical order.
Control statements do not have label fields.

READING THE MODEL

When the simulator has been "called" via the @GPSS statement the first statement it should encounter is a JOB statement. This indicates the beginning of a job. The simulator then reads the subsequent statements and checks for errors. If there are no errors it is ready to start.

STARTING SIMULATION

If there are no errors the simulator will start when and if it encounters a valid START statement.

ENDING A SIMULATION

Simulation will end when the conditions for ending specified on the START statement are satisfied or when a STOP statement is encountered in the model or when an error causes it to stop. An output report is printed.

SUBSEQUENT PHASES OF SIMULATION

If simulation has stopped normally i.e. as a result of conditions specified on a START statement or on encountering a STOP block, having produced the output report the simulator will read and interpret any statements which follow the first START statement until it encounters another START statement. It will then start again. Between the two START statements it is possible to make some changes to the original model for the second simulation phase. If one wishes to change a model statement the new statement must appear after the first START statement and it must have a label which is the same as the label on the original statement which it is to replace. Thus any statement (including a GENERATE statement) which one wishes to alter subsequently must have a label. Control statements cannot have labels and therefore cannot be overlaid in this way. A data definition statement can only be replaced (overlaid) by another data definition statement of the same type and a model block by another model block. A replacement statement must be complete. An example follows.

```
         JOB
          .
          .
          .
        START 1
        CLEAR

L1   HOLD         TELEPHONE  TIME(4)

ARRIVAL.RATE    FUNCTION . . . . . .
          .
          .
          .
        START.1
```

The statement labelled L1 replaces the original statement with label L1 which might for instance have had a different TIME value. The FUNCTION replaces the original "ARRIVAL.RATE" FUNCTION e.g. one might wish to vary the arrival rate between runs.

END A SIMULATION

Simulation will end when an END control statement is encountered. That should be the last statement.

E.4.2 List of Control Statements

STATEMENT TYPE	FUNCTION	DATA
CLEAR	To remove all TRANSACTIONS from the model and to reset all values prior to a new START. All SAVEX and matrix SAVEX values will be reset to zero unless some are specified in which case only the specified ones are reset.	Format: [item 1/item 2/ .....] where "item" can be (a) the name of a SAVEX e.g. DAY (b) an indexed SAVEX e.g. X(4) (c) an array range e.g. X(1-5) (d) an indexed matrix e.g. TAB(2,4) (e) a matrix range e.g.TAB(1-3,1-6) Note: data is optional
CLEAR,EXCEPT	As for CLEAR but here the SAVEX or matrix SAVEX values specified are the only ones <u>not</u> reset to zero,	Format: as for CLEAR
END	To indicate that the job is finished i.e. this should be the final statement	None
JOB	To indicate that what follows is a job i.e. this should be the first statement (Note: some compilers at least, apparently require that JOB should not commence in column one - this is not true of control statements in general which may start in column 1 as there is no LABEL field)	None

STATEMENT TYPE	FUNCTION	DATA
ORDER,F ORDER,S ORDER,Q ORDER,P ORDER,T ORDER,X ORDER,FN ORDER,V	These have four functions (i) To define the order in which entities are allocated (ii) To specify that an entity will be referenced using indexing (iii) To specify an entity type (iv) To allocate more than one name to the same entity. This is done by using more than one ORDER statement to effectively allocate the same entities different names (see examples on page 441) Note that the statement TYPE includes a mnemonic (F, S, Q etc.) identifying the type of entity being described	Format 1: n[,A,B,C .....] this format is used when one wishes to indicate that an entity will be indexed and optionally specify other entities of the same type n is the maximum index A,B,C ... are the names of other entities of this type (see example) Format 2: A[,B,C ......] this format is used to define and classify entities

STATEMENT TYPE	FUNCTION	DATA
RESET	To erase certain statistical data but to leave all current TRANSACTIONS as they are. (A CLEAR statement would remove all TRANSACTIONS). One must specify which entities are to be reset, however the relative clock time is always reset to zero and the total block counts are always reset to current block counts. The entities which can be reset and the effects of resetting are as follows: FACILITY "Entry count" is set equal to the number of TRANSACTIONS currently pre-empting the FACILITY and the number of TRANSACTIONS (either zero or one) currently seizing the FACILITY. STORAGE (a) The "time of last status change" is set equal to the current absolute clock time (b) "Entry count" and "maximum entry count" are set equal to the current contents	Format: [item/item/item/...]. "item" indicates the entity to be reset "item" may be the name of an entity to be reset, it may be an indexed entity, either a single entity such as F(1) or a range such as V(1 - 5) or in the case of a SAVEX matrix it may be a single element such as RECORD(4, 3) or a range such as RECORD(1 - 4, 2 - 8) If no data is specified then all entities are reset except SAVEX. If any are specified then only those specified are reset.

STATEMENT TYPE	FUNCTION	DATA
	QUEUE (a) The "time of last status change" is set equal to the current absolute clock time (b) The "total content" and "maximum content" are set equal to the current contents (c) The number of "zero-delay entries" is set equal to zero. TABLE (a) All numeric attributes are set equal to zero (b) All intermediate and final statistical data are erased SAVEX (including SAVEX MATRIX) Each location is reset to zero	

STATEMENT TYPE	FUNCTION	DATA
RESET,EXCEPT	As RESET but the items specified are those <u>not</u> to be reset	as for RESET
START	To instruct the simulator to start simulating and to give instructions on when to stop the simulation. Stopping can be on the basis of the number of TRANSACTION terminations or the time. If both are specified (n_1 and n_2) then when the first is satisfied simulation stops. By specifying either or both n_3, n_4 intermediate printouts may be obtained	Format $n_1[,n_2][,n_3]\ [,n_4]$ n_1 the number of TERMINATIONS to be recorded before stopping n_2 the simulator time at which to stop n_3 the number of terminations between intermediate printouts n_4 the simulator time between printouts
SEED	To change the series of random numbers produced by a generator by redefining any or all of the following; multiplier increment, seed. Or to define a new generator n ($n > 10$)	Format: n_1, n_2, n_3, n_4, n_1 the number ($1 \leqslant n \leqslant 10$) of the generator being altered or the number n_1 ($n_1 > 10$) of the new generator being defined

STATEMENT TYPE	FUNCTION	DATA
		n_2 the multiplier n_3 the increment n_4 the starting number or seed The word "SAME" should be put in place of n_2 or n_3 if they are not to be changed. If the word "MULT" is put in place of n_4 then the multiplier is used as the seed (as is done for the standard generators unless changed on a SEED instruction). If n_2 is specified as zero then the simulator itself produces a suitable multiplier to be used in place of the original one which will depend on the time at which the program is run.

E.4.3 Examples of Control Statements

```
CLEAR           DAY/X(1-3)/TAB(1-3, 1-6)/TAB(10, 10)
```

The above statement clears all TRANSACTIONS from the model and resets all values to what they were before the last START except SAVEX and matrix SAVEX values not mentioned in the list.

```
CLEAR,EXCEPT    HOUR/X(4-6)/TAB(4-10, 1-6)/TAB(1-10, 7-9)/TAB(1-9, 10)
```

The above statement clears all TRANSACTIONS from the model and resets all values to what they were before the last START except the SAVEX and matrix SAVEX values mentioned in the list. Note that if DAY, HOUR X(1-6), TAB(1-10, 1-10) are the only SAVEXES the above two CLEARS are equivalent.

```
ORDER,F         6, FAC.1, FAC.2, FAC.3
```

The above statement defines nine FACILITIES viz., F(1), F(2), F(3), F(4), F(5), F(6), FAC.1, FAC.2, FAC.3 and allocates the first nine FACILITIES to these.

```
ORDER,F         CHECK, CASH
```

The above statement allocates the first two FACILITIES to entities called "CHECK" and "CASH".

If the above two ORDER statements appeared in the same program then F(1) would be synonymous with "CHECK" and F(2) with "CASH".

```
ORDER,X         6, HOUR, DAY
ORDER,X         TOT.1, TOT.2
```

The above pair of ORDER statements define 8 SAVEXES viz.

X(1-6), HOUR, DAY

X(1) will also be known as TOT.1

X(2) will also be known as TOT.2

RESET	The above statement with no data resets all entities except SAVEXES (for effects of resetting see page 219)
RESET	RECORD(1-4, 2-8)
	The above statement resets rows 1-4, columns 2-8 of the matrix SAVEX "RECORD" to zero. Nothing else is reset.
RESET	F(1-3)/S(2)/HOUR
	The above statement resets the first three FACILITIES, the second STORAGE and the entity called "HOUR" only.
RESET,EXCEPT	F(1-6)
	The above statement resets all entities except the first six FACILITIES.
SEED	1, 0, SAME, MULT
	This instructs the simulator to select a new seed (n_2=0) and to use it also as the multiplier (n_4=MULT) while leaving the increment unchanged (n_3=SAME) for generator number 1 (n_1=1). This seed will be different each time the program is run since it depends on the actual time of day when the choice is made.
START	100,500,,10
	The above statement causes simulation to stop when the first of these two conditions is true (a) number of recorded TERMINATIONS equals 100 (b) simulation time equals 500. Printouts will be provided after every 10 time units.
START	1000,,50
	The above statement causes simulation to stop after 1000 recorded TERMINATIONS. Printouts will be provided after every 50 recorded TERMINATIONS.

APPENDIX F

SIMULATION OF TRAFFIC FLOW

IN A MULTI - STOREY GODOWN

INTRODUCTION TO APPENDIX F

The appendix is a report on a study which was carried out in Hong Kong. It was carried out by the author on behalf of the engineers responsible for the project. The author worked with the engineers who provided the basic data such as arrival rates and durations of various activities.

The building itself was to be a multi-storey warehouse with access to floors via spiral ramps. It was anticipated that this might cause problems for containers. In most countries such warehouses would be built all on ground level or at least with parking and loading/unloading taking place at ground level. However in Hong Kong land in urban areas is extremely expensive and it was a requirement that the warehouse or godown (as it is known in Hong Kong) be built close to the airport. A ground area to provide parking and loading/unloading facilities for several hundred large vehicles per day many of which would be forty-foot containers would be too big to be economical, hence the proposal to build it on several floors. Almost all buildings in Hong Kong residential or commercial are high-rise and it is quite usual to have multi-storey buildings with different factories on each floor.

The model referred to in this appendix as Model I is the model which was discussed and developed in Unit 9 of the teaching package.

This report has not been published outside of the company concerned with the development.

CONTENTS Page

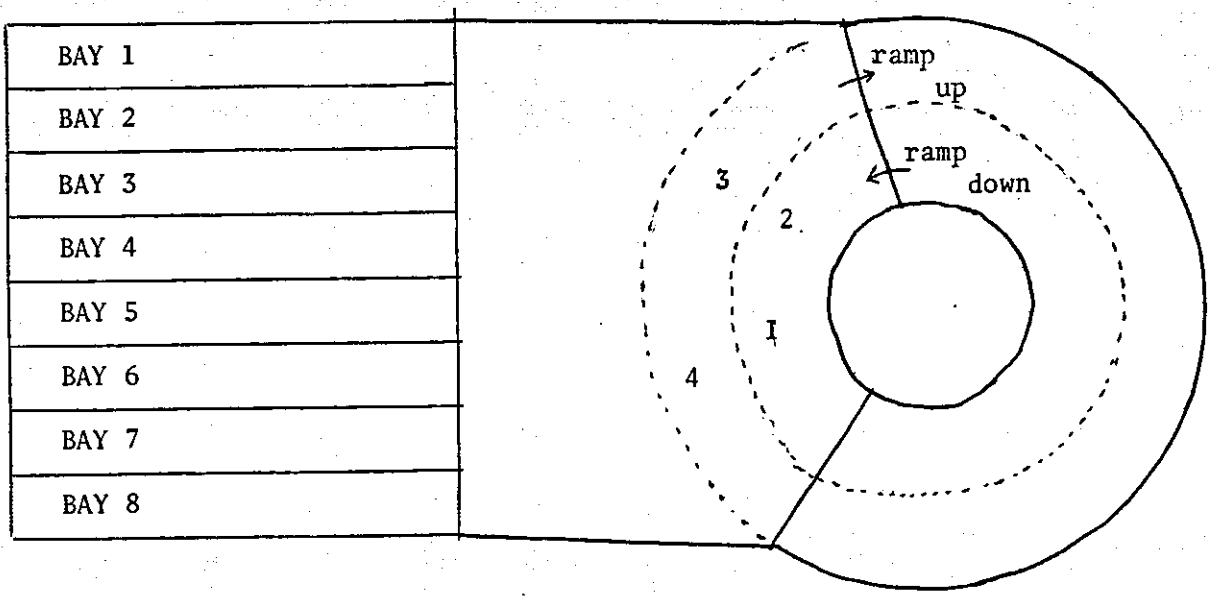
1. DESCRIPTION OF THE SYSTEM

1.1 Physical Layout of Building

As far as this study is concerned the building consists of a ground floor with no loading/unloading facilities, eight floors above that each with eight parking bays and an open roof area (ninth floor) with parking space but no loading/unloading facilities. Entry and exit are by means of a two-way ramp between floors.

1.2 Layout of Floors 1 - 8

Schematic Diagram of Floors 1 - 8



In the above diagram the apron or manoeuvering area has for convenience been divided into 4 areas or boxes numbered 1-4. This area is for vehicles manoevering in or out of the bays on this floor and also for through traffic proceeding to higher or lower floors. Interference on the apron between through traffic in both directions and vehicles trying to park on or leave from this floor causes one of the principal bottle-necks in the system and much will be said about this later.

1.3 Roof

The roof area is an open space which can accommodate a large number of waiting vehicles but they cannot be loaded or unloaded there. Consideration was given to having a check-in procedure here rather than at ground level. Vehicles which cannot park on their destination floor be cause all bays are occupied go up to the roof and wait there until space on their floor becomes available whereupon they descend.

1.4 Vehicles

Two types of vehicles are distinguished, lorries and container vehicles referred to subsequently simply as containers. These arrive at bays and are either loaded or unloaded. The material from several lorries may be consolidated here and put onto one container or vice versa. Each floor may be let to a different customer so when a vehicle arrives at the ground floor it is destined for a particular floor in the building.

1.5 Check-in Procedure

When a vehicle arrives at the entrance there are some formalities which take about 2 minutes. Consideration was given at one stage to having this take place on the roof i.e. all vehicles would proceed directly to the roof to await processing and direction to appropriate bay when available. There could be several supervisors checking in vehicles so that if say three vehicles arrived simultaneously they could be checked in simultaneously by supervisors moving along the line.

2 TRAFFIC RULES

2.1 RAMPS

Two containers cannot pass each other on a ramp even if travelling in opposite directions. Thus if a container is ascending from floor n to floor n+1 no container can descend from floor n + 1 to floor n. Parking on ramps is not allowed. There is no overtaking. This implies that a vehicle cannot be allowed to commence ascent or descent on a ramp until its exit from that ramp is clear. Ramp traffic is therefore restricted by parking space on the apron above or below. Thus if only 2 vehicles can park on an apron no more than 2 vehicles can be ascending from floor n to floor n + 1 at one time. Similarly for descent.

2.2 APRONS

Several possibilities here were tried. One was that one container or two lorries could occupy boxes one and two and one container or two lorries could occupy boxes three and four (see schematic diagram on page 446. Subsequently this was restricted to one vehicle only in boxes 1, 2 (then effectively one box) and one vehicle only in boxes (3, 4) (then effectively one box) as trials had shown that this might be the physical limit.

2.3 PARKING

A vehicle travelling from floor n - 1 or from floor n + 1 to park on floor n could require more space on the apron of floor n than a "through" vehicle for manoeuvering purposes. Different procedural rules were tried some of which had to be abandoned because they could lead to a freezing of all movement. The first set of rules tried was as follows:

A container ascending to park on floor n requires boxes 3,4 and 1 on floor n for manoeuvering so these must be free before it leaves floor n - 1. A container descending to park on floor n requires boxes 1,2 and 4 to be free on floor n before leaving floor n + 1. An ascending lorry requires only box 3 on the floor on which it is to park but a descending lorry requires all 4 boxes 1 - 4.

2.4 ROOF

The roof was considered as a flat area with no apron as such and 200 spaces for vehicles. This capacity was never fully used so there is virtually no restriction on vehicles ascending from floor 8 to park on the roof except the ramp restriction.

3. TIMES

3.1 Interarrival times of vehicles.

Originally a random pattern with peaks and troughs was used but subsequent observation of cargo consolidation operations elsewhere led to the conclusion that for a large part of the day the arrival rate was constant. It was estimated that this could be of the order of 50 vehicles per hour. However one purpose of the simulation was to determine what vehicle throughput the system could handle. The ratio of lorries to containers was estimated to be 4:1. In later simulation runs consideration was given to limiting container traffic to certain periods of the day e.g. for the first two hours of the day.

3.2 Length of day

No definite decision has been made on whether operations will be on a 24 hour basis or, say 14 hours. However since it is mainly the peak which is of concern most simulation runs simulated a constant arrival rate maintained for about 5 hours.

3.3 Check-in Time

This is estimated to require two minutes per vehicle but several may be checked in simultaneously.

3.4 Transit times

The time to travel between floors is estimated to be thirty seconds (based on a speed of 8 mph). Since there is no overtaking or parking on ramps and ascent or descent cannot start until exit is free this should be relatively constant except in the case of a breakdown or accident.

3.5 Manoeuvering prior to parking

When a lorry arrives on floor n to park it will cocupy some space on the apron for 30 seconds. When a container arrives on floor n to park it will occupy some space on the apron for 3 minutes.

3.6 Manoeuvering on leaving bay prior to descent.

This is estimated to take 30 seconds for all vehicles.

3.7 Unloading/loading time

From observation of similar operations elsewhere it is estimated that the dock time for a container will be 90 minutes and for a lorry 30 minutes. In practise a customer may keep a vehicle in a bay all day if he wishes but presumably he would only do this if he had no arriving vehicles requiring the space in which case it does not matter whether the space is regarded as occupied or not.

3.8 PRIORITY

It is possible to give certain transactions (vehicles) higher priorities than others. This means that if there is a queue for a facility such as a parking space on an apron the higher priority transactions will go to the head of a queue. Since there is no physical overtaking within the building priority can only apply to vehicles approaching a facility which they both require from opposite directions. For example if a vehicle on floor n + 1 wants to descend to floor n to park and requires box no. 3 on floor n for this purpose but finds box no. 3 occupied it must wait. Suppose in addition there is also a vehicle on floor n - 1 waiting for box 3 on floor n to become free so that it can ascend. Then it is a matter of priority as to which vehicle (the one on floor n - 1 or the one on n + 1) gets box 3 when it becomes available. If both had equal priority then of course the first to request box 3 would get it. A similar situation would arise if a vehicle was trying to leave a bay on floor n but found the apron blocked and vehicles waiting either on n - 1 or n + 1 to move to floor n. These situations only arise when both vehicles require the same box or parking position and whether they do will depend on the rules for such movements, specifically, which boxes are required by parking or departing vehicles.

3.9 . CAPACITY OF SYSTEM

Since the ratio of lorries to containers is 4:1 and the respective dock times are 30 and 90 minutes the average dock time is 42 minutes. There are altogether 64 bays (8 on each of 8 floors) so the theoretical maximum capacity is $\frac{60}{42} \times 64 = 91$ vehicles / hour. Of course this cannot be achieved because of the random nature of arrivals and because of traffic bottle-necks in the system.

4. THE SIMULATION

The programming language used was GPSS and processing was on a UNIVAC 1100 computer on a batch processing basis using punched cards.

4.1 SIMULATION CLOCK

The simulator clock advances in 5-second increments i.e. 5 units of simulator clock time = 25 seconds of real time, 720 units of simulator clock time = 1 hour. Units smaller than 5 seconds are never required. In fact all current time values are multiples of 30 seconds so an increment of 30 seconds could be used.

4.2 FUNCTIONS

The "functions" used in the program are as follows.

(i) Floor assign Function: This function gives the probability distribution of floor required. In practise a uniform distribution was used since no details are available at this stage on particular customers. Obviously some customers might use their bays more than others leading to relatively greater utilisation of some floors.

Cumulative Probability	.125	.25	.375	.5	.625	.75	.875	1
Floor	1	2	3	4	5	6	7	8

Thus each arrival is allocated a floor destination by means of a random number generator such that all floors (1-8) have equal probability of being selected. (A discrete function is used)

(ii) Type Assign Function (discrete function)

This assigns each vehicle a type number which is either 1 or 2. 1 is a lorry and 2 is a container. The ratio can be varied but the ratio used was 4:1. Actual assignment of type is on a random basis using the following cumulative probability distribution

Cumulative probability	0.8	1
Type	1	2

(iii) Rate Function (continuous function)

This function specifies the arrival rate or rather the average inter-arrival time at different times of the day.

x = time   y = average inter-arrival time.

A set of values for x and y can be specified. Values of y corresponding to intermediate values of x are got by linear interpolation.
In fact, after some initial trials this function was not used as it was decided to use a constant arrival rate. The facility still exists in the program however.

(iv) Arrival function (exponential distribution - simulator defined)
This function is used to generate random arrivals according to a specified rate.

4.3 STORAGES

GPSS uses the notion of a "Storage" to represent a physical entity with a limited capacity such as a car-park or a floor with 8 bays. In all forty storage entities are used. These are indexed S(1), S(2) ... S(40).

S(1) - S(8)	represent the floors each with capacity 8
S(9)	represents the roof with capacity 200
S(10) - S(17)	represent the apron spaces on floors 1 - 8 for ascending vehicles with capacity depending on the parking rule (see below)
S(19) - S(26)	similar to S(10) - S(17) but for descending vehicles.
S(28) - S(36)	represent the ramps between floors
S(40)	represents the check-in. The capacity is the number which can be checked in simultaneously
S(18), S(27)	unused
S(37), S(38), S(39)	technical use in program. no physical analogy (see version of simulator which does not allow parking on ascent)

Capacity of Ramps and Apron Storage Entities.

Initially the capacity of all upward and downward aprons was defined as 2 corresponding to the boxes referred to on P.1 A lorry used 1 unit of capacity and a container used 2 units. Subsequently it was decided that only one vehicle could park on either the upward or downward apron. The capacity used here was 3 of which a lorry required 2 units and a container 3. Thus only one vehicle could park and it was possible to determine the type of vehicle occupying an apron by looking at the available capacity of that storage (zero if occupied by a container, one if occupied by a lorry). The capacity of the ramps was defined as 6 of which a container required 4 and a lorry 1. This meant that 2 containers could not occupy the same ramp nor could one container and three lorries. More than 4 vehicles could not occupy a ramp simultaniously because of parking restrictions on the aprons. i.e. a vehicle cannot ascend a ramp unless parking is available on the apron above and this is limited to 2 vehicles at most, similarly for a descending vehicle.

4.4 Running time for a simulation run

The running time varies between 2½ mins. and 15 mins. per run. Debugging runs or runs which fail generally take about 2 minutes. Effective runs take a time which depends on (a) the simulated time interval and (b) the arrival rate. In fact the run time increases dramatically as the arrival rate approaches the capacity of the building. To double the arrival rate more than doubles the run-time because of traffic jams which involve many instructions to sort out especially if a vehicle is diverted to the roof. The charge for run time is $65 per min. plus some small charges for stationery. Thus effective runs cost between about $175 and $1,000 each. (All in HK Dollars)

5 OUTPUT FROM SIMULATION

5.1 QUEUES

An entity referred to in GPSS as a queue may or may not correspond to a physical queue. At any stage in its progress through a system an item may be "entered" into a "queue". At a subsequent stage it may be removed. The purpose is to gather statistical information on the "queue" or on the time the item spends in the "queue". This information is automatically provided by GPSS for all queues defined in the program. The information provided is as follows:

(a) The maximum contents of the queue
(b) The average contents of the queue
(c) The total number of items which have entered the queue
(d) The number of items which entered and left with zero time lapse
(e) The percentage of items which entered and left with zero time lapse
(f) The average time per entry for all entries
(g) The average time per entry for non-zero entries (i.e. excluding (d))
(h) The current contents

Obviously the above information is useful for physical queues. For example when a vehicle first arrives (is generated by the random procedure) it is entered into a "check-in" queue. When a supervisor becomes free it leaves that queue. The above information (a) - (h) is then available for the "check-in" queue.

The queue entity may be used to monitor other activity. For instance we may be interested in how long it takes a vehicle to travel from the ground to its destination bay. As it leaves the ground it can be entered into a specially defined "queue". When it reaches its destination it leaves the "queue". The information conveyed by (a) - (h) would be as follows:

(a) The maximum no. of vehicles progressing towards their destination at any one time.

(b) The average no. of vehicles progressing towards their destination.

(c) The total no. of vehicles which have entered the building

(d) Not applicable (always zero)

(e) Not applicable (always zero)

(f) The average time required to reach the destination

(g) Same as (f) since all are non-zero.

(h) The no. of vehicles progressing to destination when the simulation stopped.

A total of 102 queue entities have been used in the program. Many of these are of very little practical significance and their inclusion was mainly as an aid in tracing errors.

Definition of queues used

Q(1 - 8) Vehicles waiting on floor n for permission to proceed up to floor n+1. Not more than two vehicles can occupy the apron on floor n+1 at one time and this includes vehicles on the ramp ascending to floor n+1. (In later runs this was reduced to one). Similarly there cannot be more than 2 vehicles on floor n so the maximum contents of these queues should be 2. An exception is the ground floor where it was assumed that 5 vehicles could wait for permission to ascend. These 5 would already have been checked in. If 5 vehicles were waiting no further vehicles would be admitted i.e. they would remain in the street. Q(8) would consist of vehicles on floor 8 waiting to proceed to the roof. They would not be constrained by apron space on the roof but only by occupancy of the ramp. Thus the time on Q(8) should be much less than on the others.

Q(9) From arrival to reaching bay on ascent (i.e. not including vehicles which go to the roof). This provides such information as the average time required to travel up through the building including delays on intermediate aprons until the vehicle reaches its destination.

Q(10) The interval between completion of loading/unloading and exit from the building i.e. the time required to get out of the building.

Q(11 - 18) Vehicles waiting on aprons on floors 2 - 9 for permission to descend to the apron on the floor below. These are similar to Q(1-8) but for descent. Note that Q(18) is on the roof and thus can hold a number of vehicles equal to the roof capacity but Q(11 - 17) are all limited to 2.

Q(19 - 26) Vehicles waiting for permission to ascend or descend to park on floor n. These vehicles may require more space on the apron of floor n than through traffic.

Q(27) The interval between check-in and reaching the roof for vehicles which do go to the roof. In the earlier runs the decision on whether a vehicle should go to the roof was made at check-in. In later runs, traffic conditions encountered on its progress upwards could result in a vehicle's being directed to the roof at that stage of its progress. Q(27) for such vehicles would include only the time from when the decision to direct it to the roof was made to arrival on the roof.

Q(28) The interval between allocation of bay space to a vehicle on the roof and the arrival of that vehicle at the bay i.e. the time taken to descend to the required bay.

Q(29) Vehicles waiting for check-in. Given that only a certain number of vehicles can be checked in at one time a queue may form. Another possible reason for this queue would be that the area between the check-in and the ramp to the first floor is fully occupied (5 vehicles) so no more can be admitted.

Q(30) Vehicles waiting on the roof for their floor to become available (i.e. until a vehicle leaves a bay on their destination floor).

Q(31) Containers awaiting clearance to ascend. This was only used in one particular version of the program which imposed special restrictions on the entry of containers e.g. no container can enter the building while another is descending.

Q(32 - 102) All of these queues consist of the interval of time between arrival of a vehicle and final departure (to ground) of that vehicle. The various queues are composed of different sub-groups of vehicles as follows:

Q(32 - 43)	All vehicles by hour of arrival
	Q(32) = all vehicles arriving in hour 1
	Q(33) = all vehicles arriving in hour 2 etc.
Q(45 - 56)	All lorries by hour of arrival
Q(58 - 69)	All containers by hour of arrival
Q(71 - 79)	All vehicles by destination floor
	Q(71) = all vehicles destined for floor 1
	Q(72) = all vehicles destined for floor 2 etc.
Q(81 - 89)	All lorries by destination floor
Q(91 - 99)	All containers by destination floor
Q(100)	All lorries
Q(101)	All containers
Q(102)	All vehicles

Q(44), Q(57), Q(70), Q(80), Q(90) are unused.

Above definitions apply to the final model. Earlier versions did not have so many.

5.2 TABLES

G.P.S.S. has a facility for providing extra statistical analysis for specified queues when requested. It is not provided automatically for all queues as this could be wasteful. This is done by defining a "Table" associated with a queue (known in G.P.S.S. as a QTABLE). This was done in the program for Queues Q(27), Q(28), Q(29), Q(100), Q(101). The extra information provided for these queues is mainly the probability distribution of the time spent on the queue. For example suppose that the maximum possible time (estimated by the programmer) to be spent on a queue is 2 hours, the programmer might wish to know how many spent 0 - 15 mins., 15 - 30 mins. etc. The table provides this. In fact for each interval such as 0 - 15 it gives (a) the observed frequency (b) the relative frequency (c) the cumulative percentage (d) the cumulative remainder (e) the multiple of the mean (f) the no. of standard deviations away from the mean. Other information provided in a summary table is as follows for all tables:

(a) the number of entries
(b) the sum of arguments
(c) the mean argument
(d) the standard deviation

both weighted and non-weighted versions are provided but only non-weighted versions are relevant to this program.

A table may also be defined for any process requiring time and not defined as a queue. Such a table was defined to record the total waiting time at all stages of a vehicle's progress. This time is defined as follows:

minimum time	=	check-in time + time to drive unimpeded to and from the required floor + time to manoeuvre into bay + unloading/loading time + time to manoeuvre out of bay onto the apron.

for a container going to floor n this would be (in minutes)

$$2 + 2n(.5) + 3 + 90 + .5 = (95.5 + n) \text{ minutes}$$

for a lorry going to floor n this would be

$$2 + 2n(.5) + .5 + 30 + .5 = (33 + n) \text{ minutes}$$

The wait-time is then the actual time taken minus the minimum time as just defined.

This table is called WAIT.TIME

The table associated with Q(27) is called GROUNDROOF
The table associated with Q(28) is called ROOFQ
The table associated with Q(29) is called GROUNDQ

An important difference between the WAIT.TIME table and the others is that as the WAIT.TIME table is user-defined (not associated with a GPSS queue) the time units may be altered. This has been done and the WAIT.TIME table gives time in minutes whereas all other queues and tables give time in 5-second units, the basic time interval of the simulation.

5.3 Intermediate Results

In addition to the tables and queue details referred to above which are printed out when the simulation terminates details can be printed during the progress of the simulation. There is one such instruction in the program currently which prints details of a particular vehicle as it leaves the building. The details printed are as follows:

(a) The line number; this is irrelevant and simply refers to the position in the program of the print instruction.

(b) The arrival time of the vehicle in minutes from the beginning of the simulation.

(c) A 5-digit identification number. The first 3 digits are a vehicle identification number. The first vehicle is 1, the second is 2 etc. The fourth digit identifies the type, 1 for lorry, 2 for container and the fifth (unit) digit is the floor to which the vehicle went.

(d) A single digit specifying the route within the building. There are 3 possibilities

0 = did not go to the roof but directly to required floor
1 = was directed to roof on arrival at checkin
2 = was directed to floor but subsequently because of traffic conditions was redirected to the roof.

(e) The total time spent from arrival onto the checkin queue to arrival on ground floor at departure in minutes.

(f) The total waiting time i.e. the difference between the actual time as given in (e) and the minimum possible time (assuming no delays at any stage) in minutes.

5.4 Technical Details

GPSS also prints some technical details of the simulation which are mainly of interest only to the programmer. These are as follows

(a) Clock time, this is the simulated time at which the simulation stopped (measured in 5 sec. units from the beginning of the run).

(b) Termination count, this is the total number of transactions (vehicles) which have been recorded as they are terminated. This may not agree with the total of vehicles which passed through the building as at a certain (predetermined) time no further vehicles are allowed in, these extra simulated vehicles being immediately terminated but they are included in this count.

(c) A list of blocks giving for each the total number of transactions (vehicles) which passed through the block and also the current (end of simulation) contents of the block. A "block" is a program instruction which might mean something like "ascend to next floor" or "check space availability on floor n". Thus the total number of transactions gives the total number of times the system was asked to execute that instruction. Some blocks may contain several transactions awaiting processing but cannot be processed until some condition is satisfied e.g. (i) Advance time (3 minutes) - a transaction will remain in this block for 3 simulated minutes
(ii) Enter floor (n) - a transaction cannot enter this block unless there is space on floor n and hence remains in the previous block.

The current contents of blocks can be useful in tracing where vehicles are which have not left the building when the simulation ends.

5.5 SAVEX VALUES

In G.P.S.S. the term SAVEX denotes a variable e.g. "checkin-time" could be defined as a SAVEX and could hold a value such as 2 minutes. Other SAVEX locations could be used for doing arithmetical calculations. On termination of the simulation the output includes a list of all savex names and associated values at the time the simulation stopped.

5.6 Stopping a Simulation Run

A simulation may be terminated in several ways. Following are some possibilities

(a) Generate only 300 vehicles and when the 300th vehicle has left the building terminate.

(b) Continue simulation for 5 hours of simulated time and then terminate.

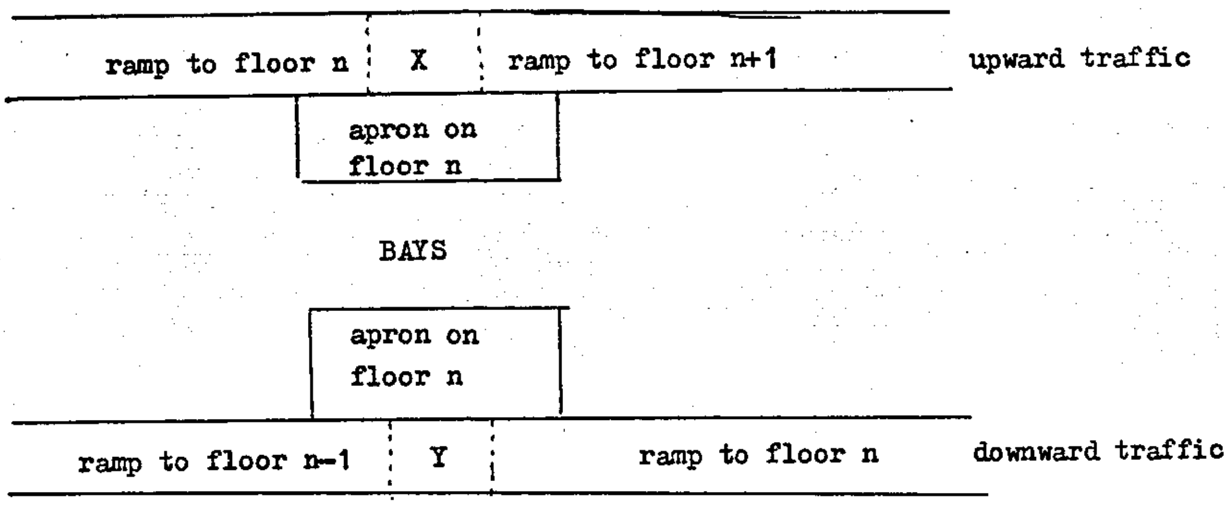
(c) Simulate arrivals for 5 hours. Allow a further period of 2 hours during which no vehicles arrive but during which all vehicles already in the building can be cleared.

The method used was mainly (c)

PRELIMINARY MODEL (MODEL I)

Relative to subsequent models, the preliminary model had simple rules. The original program had 179 instructions. Subsequent models had up to 400 instructions.

In this model only one container at a time could move anywhere within the building. The apron areas were used only for vehicles parking on or departing from that floor and not for through traffic. Schematically this can be represented as follows:



If the apron was occupied a vehicle would remain at X or Y blocking through traffic. There was no interference between ascending and descending traffic.

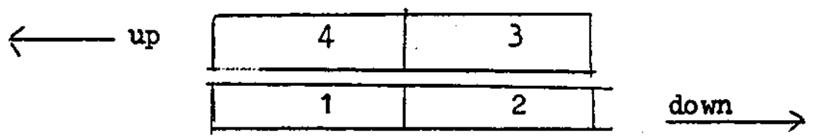
The times used for manoeuvering etc. in this version were arbitrarily chosen and much too small as become apparent later. The check-in time was 15 secs. (2 mins. in subsequent models) manoeuvering time was 25 secs. (subsequently 3 mins. for a container, 30 secs. for a lorry), dock time was 1 hour for a container and 30 mins. for a lorry (subsequently 90 and 30 mins. respectively), the drive time between floors was 5 secs. (subsequently 30 secs.). The arrival rate varied between 20/hour off peak to 50/hour at peak and averaged 40/hour.

REVISED MODEL WITH REALISTIC RULES (MODEL II)

In order to make the model more realistic complex rules for the use of the aprons were introduced. These rules were as follows.

Restrictions on vehicles movements on Ramp/Apron

1. Container vehicles cannot move between adjoining apron levels over same time interval in opposing directions.
2. Other vehicles can move between adjoining apron levels over same time interval in opposing directions.
3. For simulation purposes the apron area should be divided into four segments - two to each traffic "lane" in each direction.



For vehicles on the apron level in question, the following "rules" apply:

a. Container vehicle can leave box 4 if box 3 and box 4 on the next level above are vacant.
b. Other vehicle can leave box 4 if box 3 on the next level above is vacant.
c. Container vehicle can leave box 2 if box 1 and box 2 on the next level below are vacant.
d. Other vehicle can leave box 2 if box 1 on the next level below is vacant.

The following additional rules apply to vehicles on the apron level in question in regard to the destination/operation required at the adjacent level:

If the vehicle whose departure from the apron level is being studied requires to park at the adjoinint level, then different conditions on adjoining apron status are required. On the basis of container vehicle parking being assumed to take place adjacent box 4 only then:

e. Container vehicle can leave box if boxes 3, 4 and 1 on the next level above are vacant.
f. as (b) above.
g. Container vehicle can leave box if boxes 1, 2 and 4 on the next level below are vacant.
h. Other vehicle can leave box 2 if boxes 1, 2, 3 and 4 on the next level below are vacant.

A departing vehicle requires all 4 boxes on its floor.

Simulation Parameters

Speed of movement between aprons: time of movement from leaving sector 4 to arrive at sector 3 or 4 at next level or from sector 2 to sector 1 or 2 at next level = 30 seconds (indicates average speed just over 3 mph for full circuit if continuing to other levels).

Time of clearing apron for parking:

for other than container : 30 seconds

for containers : 3 minutes: note that after the first two minutes then use could be made of boxes 1 and 2.

Lorry: Container ratio: Likely ratio on basis of prevalent container usage indicates ratio of 4 lorries to 1 container.

Container Dock Time: allow 90 minutes.

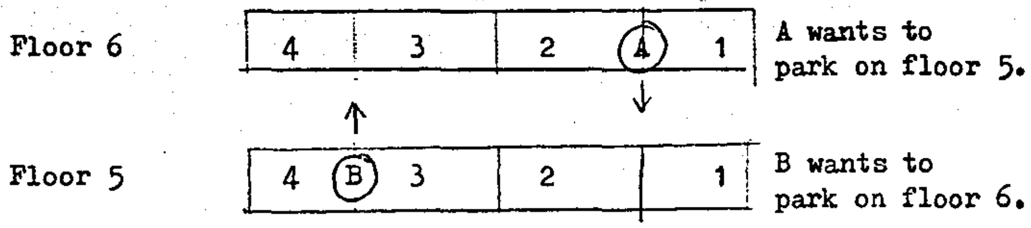
Other Vehicle Dock Time: allow 30 minutes.

Time to clear apron on departure: allow 30 seconds.

The division of the apron into 4 boxes was achieved by representing each apron by two "STORAGES" (one for ascending, one for descending) each with a capacity of two. A container would require 2 units of storage (2 boxes) and a lorry one, for through traffic.

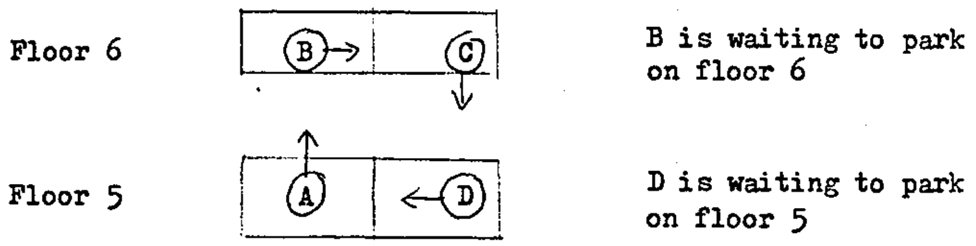
Result of Simulation

When the simulation was run it was found that the traffic rules could lead to a situation where all traffic stopped and this in fact happened. The problem can be illustrated as follows:



B occupies boxes 3 and 4 on floor 5 and is waiting to ascend to floor 6. It must wait till A moves as it requires box 1, but A cannot descend as A requires box 4 occupied by B so neither can move and all traffic would eventually stop.

At first it appeared that this could be prevented by allowing A to descend to floor 5 and wait in boxes 1 and 2 until B left 3 and 4. However this leads to an even worse situation as illustrated below:


Floor 6
B
C
B is waiting to park on floor 6
Floor 5
A
D
D is waiting to park on floor 5

Since this problem is caused by vehicles approaching to park from opposite directions a simple solution was to make all vehicles go to the roof and park on descent only. This would have the added advantage of transferring the check-in queue (along with the check-in procedure) from the street to the roof where there was plenty of space. The program was therefore modified to allow parking on descent only (Model III)

NO PARKING ON ASCENT (MODEL III)

This model is only slightly different from the previous one. In the previous model a vehicle is directed to the roof if all bays on its floor are full when it checks in at the ground floor entrance. In Model III all vehicles are directed to the roof and the check-in procedure takes place there. This avoids the potential traffic freeze-up situation because in this model ascending traffic require only boxes 3 and 4 on the ascent apron and do not cross over. The freeze-up in the previous model was the result of a descending vehicle requiring the ascent boxes 3 or 4 for parking and the ascending vehicle requiring the descent boxes (1 or 2).

The model was run with several arrival rates to see what was the maximum traffic flow the building could handle without having very long delays. The rates used were 48, 57 and 74 per hour. Some of the results are shown in the table on the following page. The average waiting time was 19, 33 and 105 minutes. Since an average waiting time of about 33 minutes would be just about tolerable it seems that the capacity is about 60 vehicles per hour with a container/lorry ratio of 1:4. The street queues are small because check-in is on the roof. The only reason for a street queue is the fact that vehciles may have to wait before ascending the ramp to floor 1.

There were objections in principle to making all vehicles go to the roof. It would seem silly to a driver to have to pass his floor which had empty bays to go up to the roof and back down again. It would not be feasible to give drivers discretion as to whether to park or go to the roof as this would lead to uncontrollable situations e.g. drivers waiting on intermediate aprons to avoid going to the roof.

RESULTS FROM MODEL III

Arrival Rate per Hour	49	57	74
Maximum Time in System for 90% of Containers	-	180	300
Maximum Time in System for 90% of Lorries	-	100	233
Average Time in System for Containers	123	144	226
Average Time in System for Lorries	56	67	136
Average Time in System for All Vehicles	68	83	154
Average Waiting Time in the System (all vehicles)	19	33	105
Maximum Waiting Time for 90% of vehicles	32	65	200 *
Average Size of Street Queue	.01	.08	.19
Maximum Size of Street Queue	2	4	7
Utilisation of Bays	.45	.66	.75
Average Time in System for Vehicles going to Floor 1	72	88	158
2	65	82	127
3	65	84	149
4	64	77	122
5	73	80	134
6	64	73	122
7	80	99	210
8	65	83	213
Average Time in System for Vehicles arriving in Hour 1	65	65	88
2	63	70	142
3	67	83	159
4	78	105	191
5	74	94	203
6	63	-	-
7	69	-	-

* With so many figures to look at it is difficult to know which is important but it is suggested that this is the most relevant figure i.e. the waiting time (time wasted) which will be exceeded by at most 10% of vehicles. This should be considered in conjunction with the one immediately above it. i.e. the average waiting time.

NO OVERTAKING OF CONTAINERS (MODEL IV)

At this time it was decided that it might not be possible in fact for any vehicle to pass a container (subsequent trials reversed this), so it was decided to modify the program to prevent any passing of containers. This would solve the traffic freeze-up problem without having to send all traffic to the roof but it could be expected to cause large street queues while containers waited for all departing vehicles to leave before they could enter.

This change was accomplished as follows. Suppose a container arrives at the ground floor and wants to park on floor 3. It waits until apron positions 1, 2 on floor 3 are free. These are then reserved. It repeats this successively for floor 2 and floor 1 until all descending aprons are free. It then commences ascent. As it passes a floor the descending boxes (1,2) on that floor are released for departing vehicles. For a descending container the situation is reversed. It was assumed that a container on the ground awaiting clearance to ascend could be overtaken by an ascending lorry and similarly for one descending.

As can be seen from the results on the following page this restriction would cause very long delays at the ground check-in. Also the maximum waiting time of 132 for 55 vehicles/ hour is excessive.

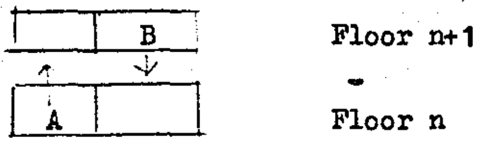
RESULTS FROM MODEL IV

Arrival Rate per Hour	45	55
Maximum Time in System for 90% of Containers	288	353
Maximum Time in System for 90% of Lorries	117	139
Average Time in System for Containers	192	252
Average Time in System for Lorries	62	69
Average Time in System for All Vehicles	89	108
Average Waiting Time in the System (all vehicles)	39	51
Maximum Waiting Time for 90% of vehicles	100	132
Average Size of Street Queue	2.88	4.5
Maximum Size of Street Queue	21	28
Utilisation of Bays	.52	.56
Average Time in System for Vehicles going to Floor 1	81	103
2	85	104
3	81	100
4	74	101
5	95	101
6	83	86
7	117	151
8	93	110
Average Time in System for Vehicles arriving in Hour 1	58	60
2	52	79
3	101	115
4	89	160
5	123	127
6	98	-
7	122	-
Max. no. of containers awaiting clearance at one time	19	21

DIVERSION TO ROOF AND REDUCED EFFECTIVE APRON SIZE (MODEL V)

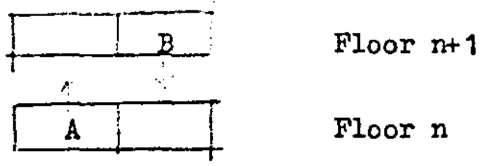
It was decided after further trials that in fact any vehicle could pass a container on an apron and a lorry could pass a container on a ramp. This was the original rule in MODELS III and IV. This rule was restored in MODEL V. In addition as a result of the trials it was decided that the pairs of boxes hitherto referred to as 1, 2 (descent) and 3, 4(ascent) should in fact be reduced to one only for descent and one only for ascent. This was because there did not appear to be sufficient room for 2 lorries to park while waiting to ascend or descend. The capacity of the storages corresponding to the aprons was consequently changed to 3 units of which a lorry required 2 and a container 3. Thus only one could occupy the apron and it would be possible by testing the capacity remaining to tell whether the vehicle was a container or a lorry.

Although it was decided that 2 containers could pass on an apron this would require extra manoeuvering by the descending container. The following rule was incorporated.



If a container A is on floor n waiting to ascend to floor n+1 while a container B on floor n+1 is waiting to descend to floor n then A waits one minute for B to move aside. A then ascends. When A, reaches floor n+1 B can descend assuming no other container has arrived on floor n. It is assumed here that A does not want to park on floor n + 1

In order to avoid the traffic freezing situation described for MODEL II the following rule was incorporated.



If a container A is on floor n waiting to ascend to floor n+1 to park but there is a vehicle on the apron of floor n+1 waiting to descend then A is directed to the roof i.e. it does not wait on floor n for the apron on n+1 to become clear.

At this stage it was also decided to try varying the arrival pattern e.g. lorries only or containers only or containers only for 2 hours followed by lorries only for 4 hours. This latter combination has the advantage that descending containers would rarely encounter ascending ones as by the time the first container was leaving the last would have arrived (there might be a small overlap if the arrival period was 2 hours).

Comments on Results of Model V

Results are shown on the following page for 3 arrival rates viz. 41, 55, 64 per hour respectively. For 55 and 64 per hour the street queues are excessive but this was to be expected since the check-in time was 2 mins. and only 2 at a time could be dealt with. The waiting times were not too bad but of course they would get longer as time progressed and the street queue built up. The simulation was only for 5 hours and for the first 2 hours waiting times of points inside the building would be low because the building was empty and is just filling up.

On page 469 some results are given for different traffic conditions. A shows the results for a system using all lorries (70 per hour). As can be seen the street queue is large because of the rate and will be increasing. B gives results for a system which allows only containers for the first 2 hours and then only lorries for the following 5 hours. Waiting times were not unreasonable. But street queues are very big.

To reduce the problem of street queues it was decided to assume that 4 vehicles could be checked in simultaneously. This could be done by 4 supervisors moving along the line whenever a queue materialises. As can be seen from result C the maximum street queue is reduced from 19 (inB) to 4 and the average time in the system for lorries is reduced from 43 to 40 minutes.

It was decided at this stage to test the effect of not allowing lorries to pass containers on the ramps and at the same time to increase the arrival rate. The previous example (C on page 469) showed that the system could handle more than 25 containers per hour and 55 lorries per hour so these figures were increased to 40 and 70 respectively. At the same time it was to try allowing a second batch of containers later in the day.

RESULTS FROM MODEL V

Arrival Rate per Hour	41	55	64
Maximum Time in System for 90% of Containers	120	158	190
Maximum Time in System for 90% of Lorries	47	90	110
Average Time in System for Containers	106	130	135
Average Time in System for Lorries	48	64	73
Average Time in System for All Vehicles	53	77	85
Average Waiting Time in the System (all vehicles)	3	27	36
Maximum Waiting Time for 90% of vehicles	9	65	75
Average Size of Street Queue	•26	7•9	17
Maximum Size of Street Queue	5	24	49
Utilisation of Bays	•3	•5	•55
Average Time in System for Vehicles going to Floor 1	54	72	81
2	43	78	76
3	50	84	77
4	44	67	87
5	54	86	82
6	48	75	86
7	69	80	98
8	55	73	93
Average Time in System for Vehicles arriving in Hour 1	56	66	54
2	44	73	62
3	58	91	122
4	55	80	117
5	46	76	83
Number which was directed to the roof from the ground	8	48	26
Number diverted to the roof because of traffic conditions	5	18	24
Number of containers moved aside to allow another to pass	15	32	48

RESULTS FROM MODEL V

A: all lorries at 70/hour arrival rate

B: 25 containers/hour for 2 hours followed by 55 lorries per hour for 5 hours

C: as B but 4 can be checked in simultaneously (2 only in all previous models)

		A	B	C
Arrival Rate per Hour		70	25/55	25/55
Maximum Time in System for 90% of Containers		-	115	110
Maximum Time in System for 90% of Lorries		67	50	50
Average Time in System for Containers		-	106	106
Average Time in System for Lorries		54	43	40
Average Time in System for All Vehicles		54	53	50
Average Waiting Time in the System (all vehicles)		20	5	3
Maximum Waiting Time for 90% of vehicles		35	15	8
Average Size of Street Queue		12.18	1.6	.06
Maximum Size of Street Queue		36	19	4
Utilisation of Bays		.45	.30	.32
Average Time in System for Vehicles going to Floor	1	49	46	44
	2	51	54	51
	3	50	46	43
	4	51	51	49
	5	52	55	53
	6	58	55	53
	7	58	56	55
	8	60	58	55
Average Time in System for Vehicles arriving in Hour	1	40	106	105
	2	46	107	107
	3	59	44	43
	4	65	39	38
	5	64	48	42
	6	-	40	38
	7			39
Number which was directed to the roof from the ground		6	20	20
Number diverted to the roof because of traffic conditions		0	2	2
Number of containers moved aside to allow another to pass		0	15	15

MODEL VI

Since it appeared from trials that it would be difficult for a lorry and container to pass on a ramp the program was modified to prevent any vehicle travelling between 2 floors at the same time as a container. As the waiting times for the previous run (25 containers per hour followed by 55 lorries per hour) were very short it was decided to increase these to test the capacity. The figures used were 40 containers per hour for 80 minutes followed by 70 lorries per hour for 3 hours followed by a second batch of containers at 40/hr. for 80 mins. followed by a second batch of lorries at 70/hr. for 3 hours. Thus in a total time of 8 hr. 40 mins. the number of vehicles would be 526 or an average of 61 per hour. In fact the simulation did not continue for 8 hr. 40 mins. but was stopped after 7 hours i.e. about half-way through the second batch of lorries. The capacity of the check-in was increased to 6 i.e. 6 could be checked-in simultaneously. This was done so that if large queues appeared on the street it would be apparent that it was not caused by the check-in but by the ramp.

The results of this run are shown on the following page.

RESULTS FROM MODEL VI

Arrival Rate per Hour	40/70
Maximum Time in System for 90% of Containers	225
Maximum Time in System for 90% of Lorries	190
Average Time in System for Containers	170
Average Time in System for Lorries	124
Average Time in System for All Vehicles	139
Average Waiting Time in the System (all vehicles)	73
Maximum Waiting Time for 90% of vehicles	140
Average Size of Street Queue	32
Maximum Size of Street Queue	66
Utilisation of Bays	.82
Average Time in System for Vehicles going to Floor 1	102
2	130
3	138
4	142
5	131
6	127
7	161
8	153
Average Time in System for Vehicles arriving in Hour 1	123
2	104
3	114
4	142
5	193
6	173
7	125
Number which was directed to the roof from the ground	194
Number diverted to the roof because of traffic conditions	16
Number of containers moved aside to allow another to pass	22

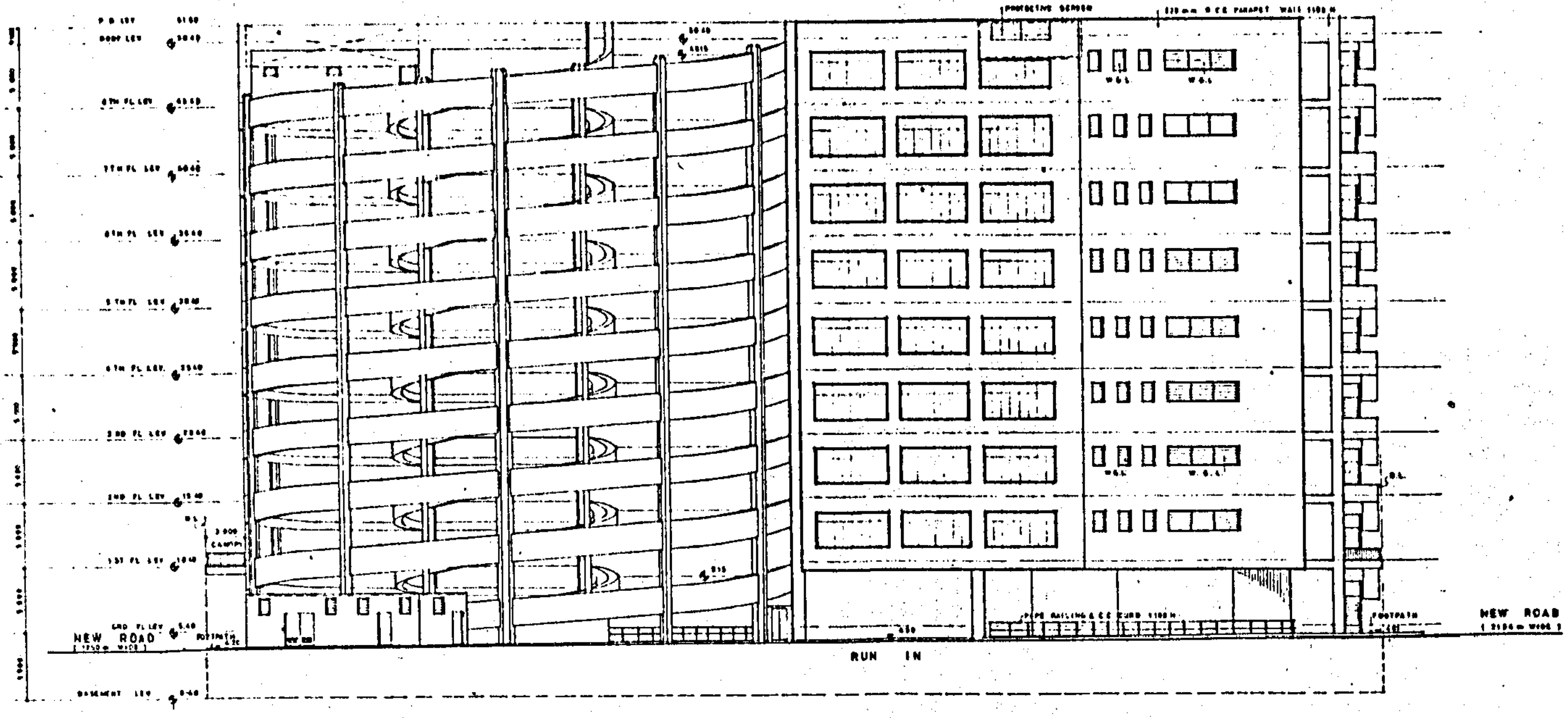
Comments on Results of Model VI

The waiting times and the street queues have increased enormously. The street queue is caused by over utilisation of the ramp to floor 1. How this arises can be seen from the following figures:

Time	No. of Containers Arriving	No. of lorries Arriving	No. of Containers Departing	No. of lorries Departing	Total Requiring Ramp 1
8.00- 8.20	13		0	0	
8.20- 8.40	13		0	0	
8.40- 9.00	13		0	0	
9.00- 9.20	13				
9.20- 9.40	0				
9.40-10.00	0	23	13		23L + 13C
10.00-10.20	0	23	13		23L + 13C
10.20-10.40	0	23	13	23	46L + 13C
10.40-11.00	0	23	13	23	46L + 13C
11.00-11.20	0	23	0	23	46L
etc.					

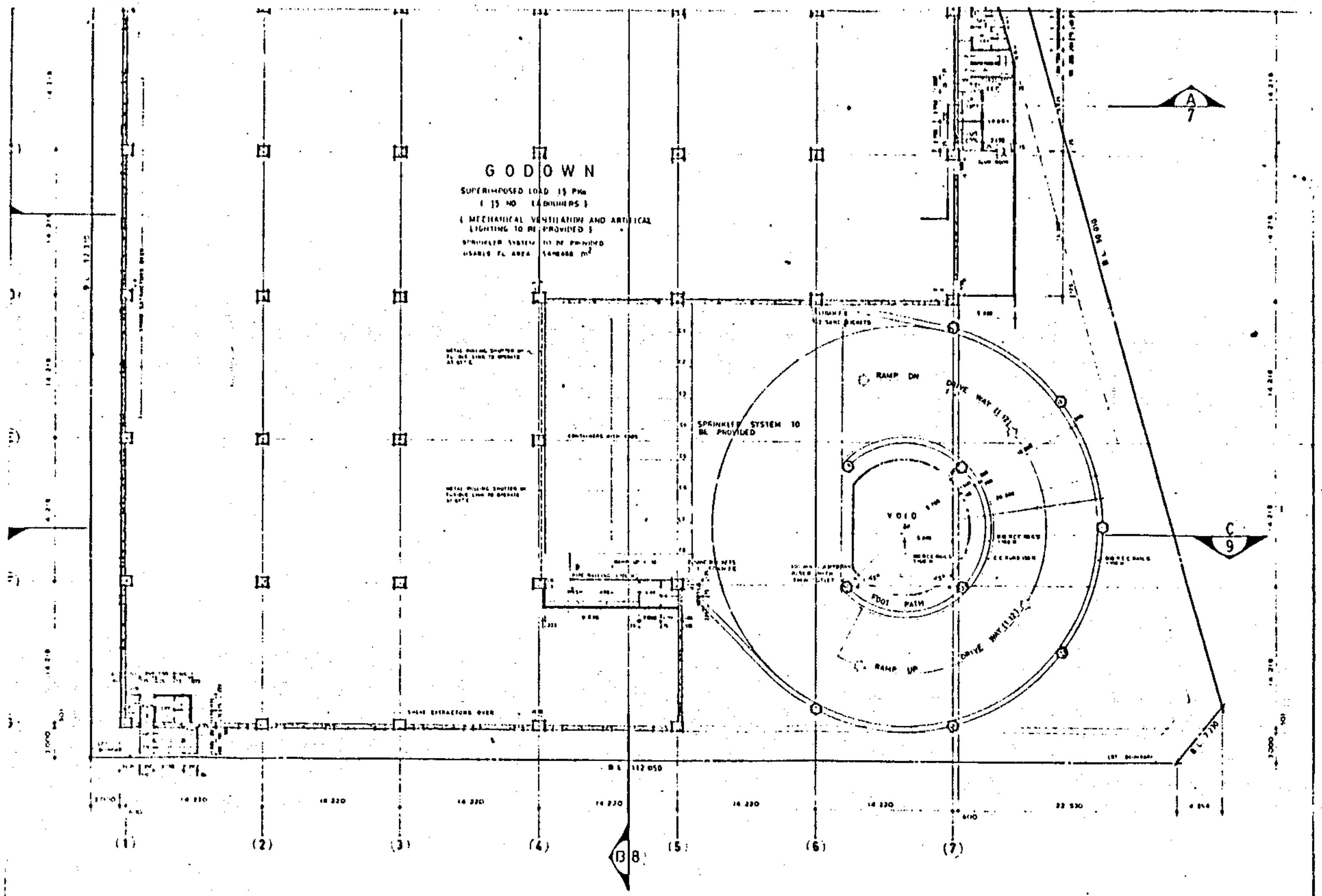
The numbers shown above as "departing" in a period are actually trying to depart. It is assumed that containers which arrived in the interval 8.00-8.20 will be trying to leave in the interval 9.40-10.00. Lorries which arrive between 9.40 and 10.00 will try to leave between 10.20 and 10.40. If each vehicle occupies a ramp for 30 seconds, in a 20 min. period if 40 lorries tried to use the ramp there would always be a lorry either ascending or descending. In fact with 46 there would in general be more than 1 lorry occupying the ramp to floor 1, so the 13 containers could not get out. The consequence would be that a large street queue would build up and vehicles could not leave bays because ramps were blocked. This would cause entering vehicles to be sent to the roof which would make traffic conditions even worse.

It may be that the 30 seconds of ramp occupancy is too high. This 30 seconds in fact includes an allowance for stopping and starting on an apron. If this was divided into say 20 seconds for travelling up the ramp and 10 seconds for crossing the apron then the ramp could be released aften 20 seconds instead of 30 seconds.



EAST ELEVATION

(FACING TAI YIP STREET 15.80m WIDE)


GODOWN
SUPERIMPOSED LOAD 15 PKa
(15 NO LABOURERS)
(MECHANICAL VENTILATION AND ARTIFICAL LIGHTING TO BE PROVIDED)
SPRINKLER SYSTEM TO BE PROVIDED
SPRINKLER SYSTEM TO BE PROVIDED
VOID
FOOT PATH
RAMP DN
RAMP UP
DRIVE WAY
(1)
(2)
(3)
(4)
(5)
(6)
(7)
A 7
C 9
B 8

TYPICAL FLOOR PLAN