SIMULATION LANGUAGE

Unit 8

Simulation Language

- A simulation language is a special purpose language structured to meet the programming requirements of the simulation models of a specific class of situations.
- The analyst develops the simulation model, employing this special purpose language, which is applicable over a class of applications.
- For example, the simulation language **GPSS** is applicable to queuing like situations while SIMAN (**SIM**ulation **AN**alysis) and SLAM II are more appropriate for simulating the manufacturing and material handling systems.

Simulation Language..

The important features of simulation language can be identified as follows:

- Generation of large streams of random numbers
- Generation of random variates from a large number of probability distributions
- Determining the length of simulation run and length of wrapping (covering) up period
- Advancing the simulation clock
- Scanning the event list to determine the next earliest event to occur
- Collecting data
- Analyzing data and setting confidence intervals

Simulation Language..

These common features, which have to be invariably modeled in all simulations, are perhaps the cause of the development of special simulation software. Simulation software packages are designed to meet the following objectives:

- **To conveniently describe the elements**, which commonly appear in simulation, such as the generation of random deviates.
- **Flexibility of changing the design** configuration of the system so as to consider alternate configuration.
- Internal timing and control mechanism, for book **keeping of the vital information during the simulation run.**
- **To obtain conveniently**, the data and statistics on the behavior of the system.

Merits of Simulation Language

- Since most of the features to be programmed are in-built, simulation languages take comparatively ***less programming time and effort.***
- Since simulation language consists of blocks, specially constructed to simulate the common features, they ***provide a natural framework for simulation modelling***.
- The simulation models coded in simulation language can ***easily be changed and modified***.
- The ***error detection and analysis*** is done automatically in simulation languages.
- The simulation models developed in simulation languages, especially the specific application packages, called simulators, are very ***easy to use***.

Desirable Features of Simulation Software

1. Modelling Flexibility
2. Ease of Modelling
3. Fast Execution Speed
4. Compatibility of various Computer Systems
5. Statistical Capabilities
6. Capability of Animation
7. Report Presentation Capabilities

GPSS (General Purpose Simulation System)

- **GPSS**, which stands for ***General Purpose Simulation System***. This language was developed primarily by Geoffrey Gordon at IBM around 1960, and has contributed important concepts to every commercial discrete event Computer Simulation Language developed ever since.
- **GPSS** is a discrete time simulation general-purpose programming language, where a simulation clock advances in discrete steps. A system is modeled as transactions enter the system and are passed from one service (represented by blocs) to another.
- This is particularly well-suited for problems such as a factory. **GPSS** is less flexible than simulation languages such as Simulate and SIMSCRIPT but it is easier to use and more popular.

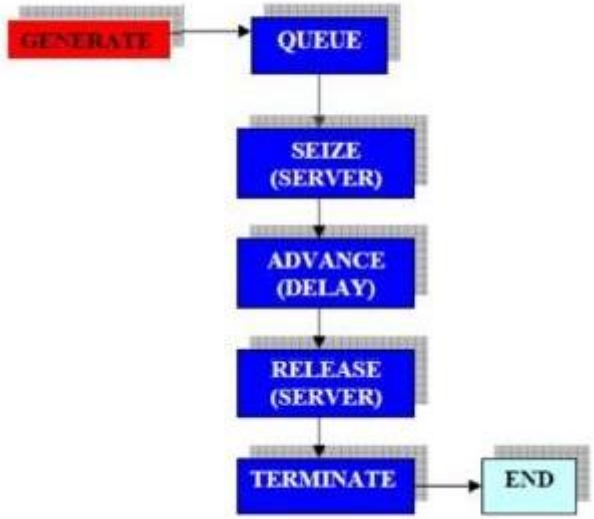
GPSS (General Purpose Simulation System)..

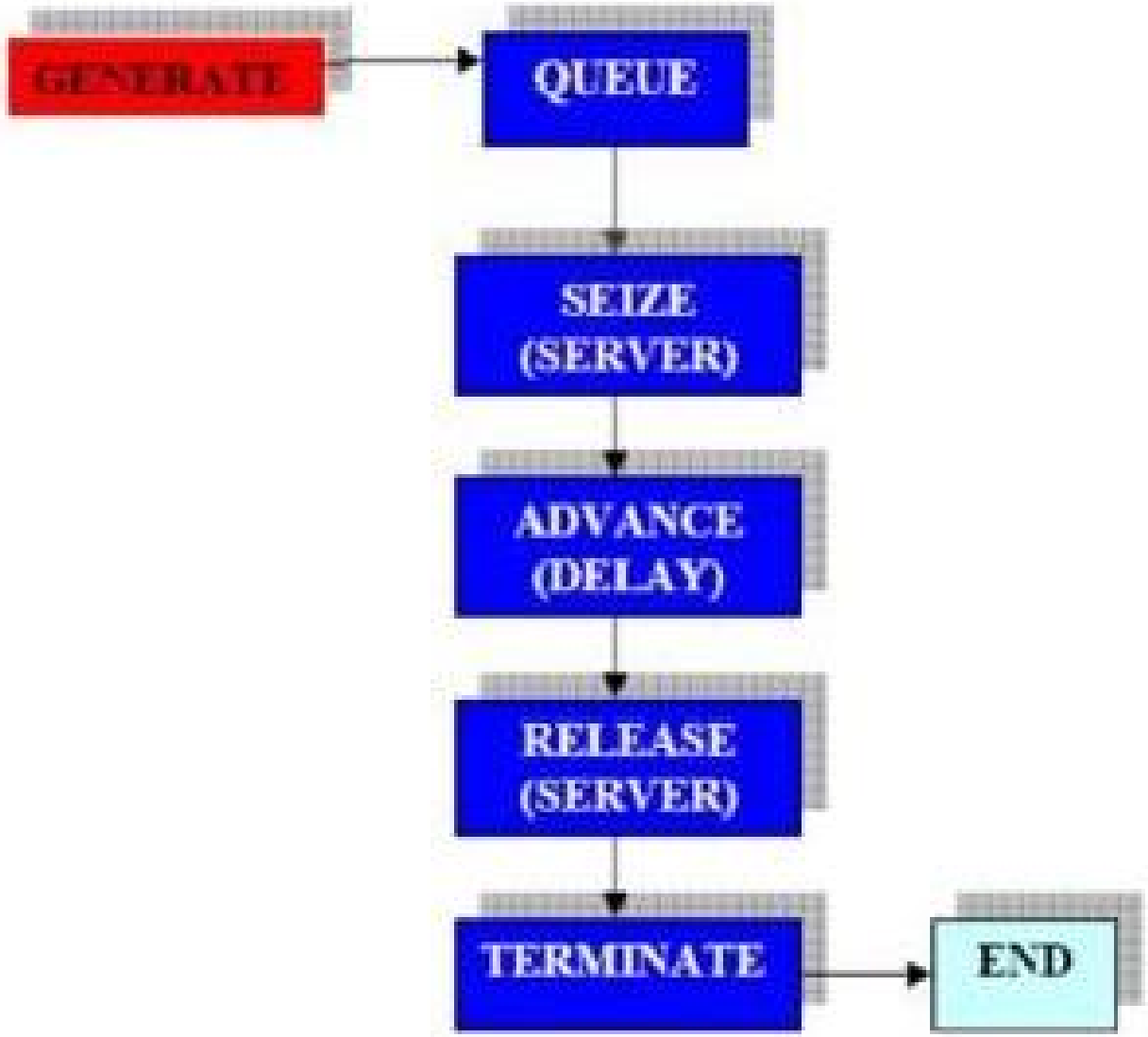
- **GPSS** was designed especially for analysis who were not necessarily computer programmer. It is particularly suited for modeling traffic and queuing systems.
- A **GPSS** programmer does not write a program in the same sense as SIMSCRIPT programmer does instead, he construct a block diagram – a network of interconnected blocks, each performing a special simulation oriented function.
- **GPSS** provides a set of 48 different blocks to choose from each of which can be used repeatedly.
- Each block has a name and specific task to perform. Moving through the system of block are entities called transaction are customer, messenger, machine parts, vehicle, etc.

GPSS (General Purpose Simulation System)..

Basic Structure

- A transaction is a GPSS object with a number of attributes. A transaction is like a customer entering into the process for service.
- A single transaction may represent several individual entities. Each transaction has to be generated either one at a time or in batches. Once they appear into the system, they must be contained exactly in one action Block. However, a Block may contain many transactions.




GENERATE
QUEUE
SEIZE
(SERVER)
ADVANCE
(DELAY)
RELEASE
(SERVER)
TERMINATE
END

Basic Structure

- A **GENERATE** Block ***generates a stream of transactions with a specific set of behavior***. No transaction may again enter this block. Behavior could be deterministic, stochastic, functional, etc. A Transaction leaving a GENERATE Block descends into the next available Block it finds. The entering Block shouldn't deny entries to transactions. Otherwise, system backups may result.
- A **QUEUE** Block never ***refuses any transaction***. If a transaction cannot enter into the next Block, it stays at the current Block. Therefore, a QUEUE simulates an infinitely long buffer.

Basic Structure

- A transaction attempts to **SEIZE *a facility (server, router, CPU) for service.*** If it succeeds, it would leave the current Block and start using the facility. If not, it stays where it is until the next time. As long as a facility is occupied, it cannot allow another transaction to SEIZE it.
- An **ADVANCE** Block captures the transaction and imposes a ***delay on it wherever it is***. The delay could be deterministic, probabilistic, etc.
- A **RELEASE** Block forces ***a transaction to release its facility***. For every successful SEIZE, there must be a RELEASE.
- A **TERMINATE** Block ***kills the entering transaction*** here.

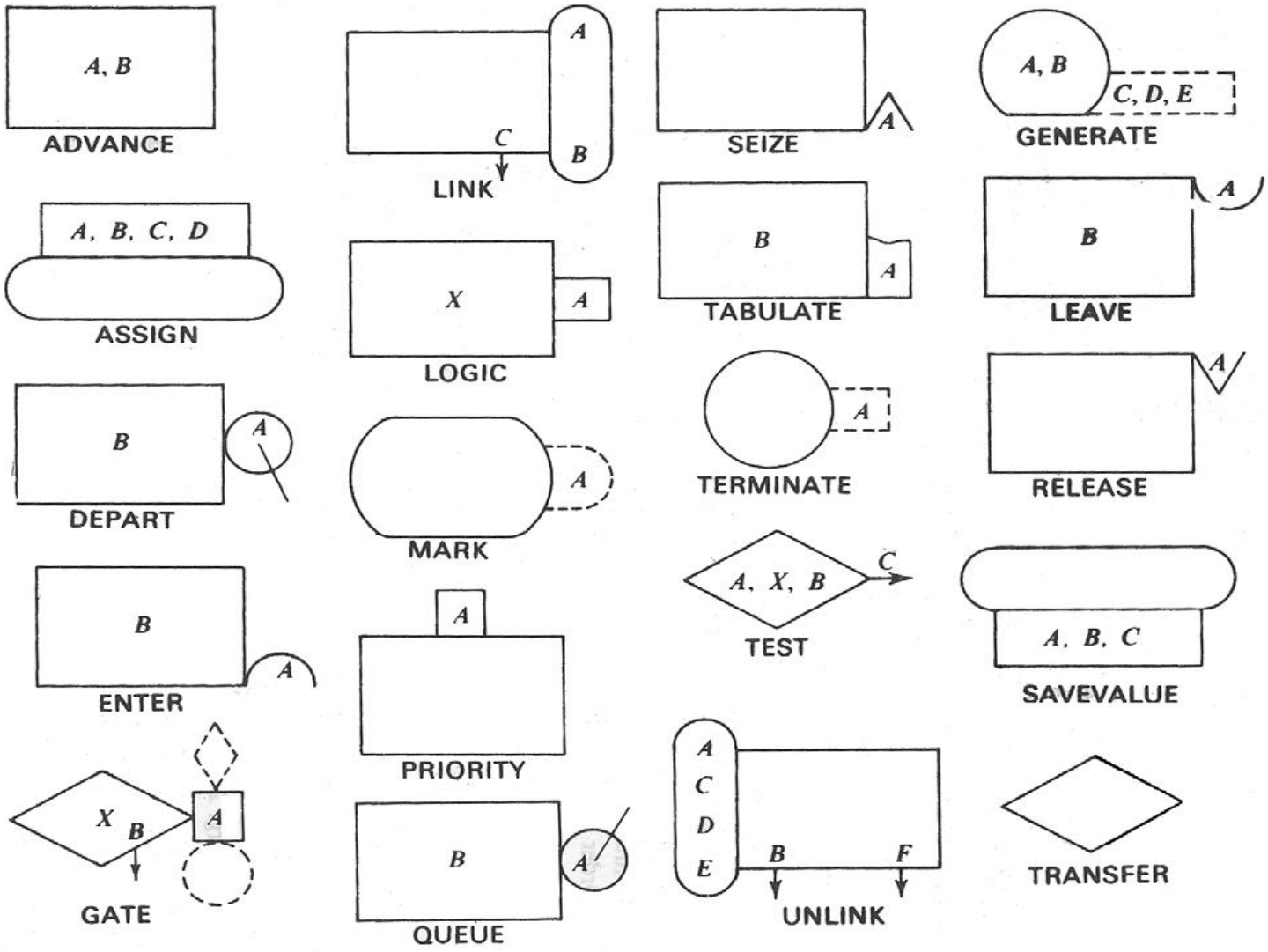
Basic Structure

- **TABULATE**, tabulates the time in it took the transaction to reach that point from the time it entered the simulation system. GPSS handles the advancement of time by a block called ADVANCE. When a transaction enters the block an action time is computed and added to the current time to produce a block departure time. When the time reaches the departure time the transaction will be moved, if possible, to the next block in the chart. Transaction might process certain attributes which are used to make logical decision within block.

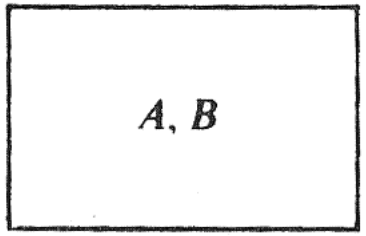
GPSS - Basic Commands

- CLEAR : reset statistics and remove transactions
- CONTINUE : resume the simulation
- EXIT : end the GPSS world session
- HALT : stop the simulation and delete all queued commands
- INCLUDE : read and translate a secondary model file
- INTEGRATE : automatically integrate a time differential in a use variable
- REPORT : set the name of the report file or request an immediate report
- RESET : reset the statistics of the simulation
- SHOW : evaluate and display expression
- START : set the termination count and begin a simulation
- STEP : attempt a limited number of block entities
- STOP : set a stop condition based on block entry attempts
- STORAGE : define a storage entity
- TABLE : define a table entity
- VARIABLE : define a variable entity

GPSS – Blocks


A, B
ADVANCE
A
C
B
LINK
SEIZE
A
A, B
C, D, E
GENERATE
A, B, C, D
ASSIGN
X
A
LOGIC
B
A
TABULATE
B
A
LEAVE
B
A
DEPART
A
MARK
A
TERMINATE
A
RELEASE
A
B
ENTER
A
A
PRIORITY
A, X, B
C
TEST
A, B, C
SAVEVALUE
X
B
A
GATE
B
A
QUEUE
A
C
D
E
B
F
UNLINK
TRANSFER

GPSS



ADVANCE

An ADVANCE Block delays the progress of a Transaction for a specified amount of simulated time.

ADVANCE A,B

Operands

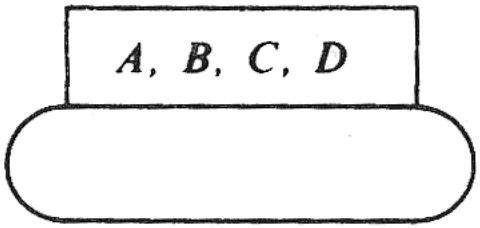
A - The mean time increment. Required. The operand must be Name, Number, String, ParenthesizedExpression, SNA or SNA*Parameter.

B - The time half-range or, if a function, the function modifier. Optional. The operand must be Null, Name, Number, String, ParenthesizedExpression, SNA or SNA*Parameter.

Example

ADVANCE 101.6, 50.3

This example creates a Block which chooses a random number between 51.3 and 151.9, inclusively (i.e. 101.6 plus or minus 50.3), and



ASSIGN

ASSIGN Blocks are used to place or modify a value in a Transaction Parameter.

ASSIGN A,B,C

Operands

A - Parameter number of the Active Transaction. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter, followed by +, -, or Null.

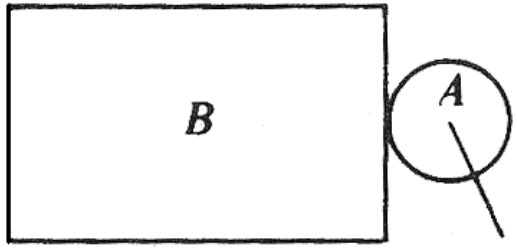
B - Value. Required. the operand must be Name, Number, String, ParenthesizedExpression, SNA, or SNA*Parameter.

C - Function number. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter

Examples

ASSIGN 2000, 150.6

This is the simplest way to use the ASSIGN Block. The value 150.6 is assigned to Parameter number 2000 of the entering Transaction. If no such Parameter exists, it is created.



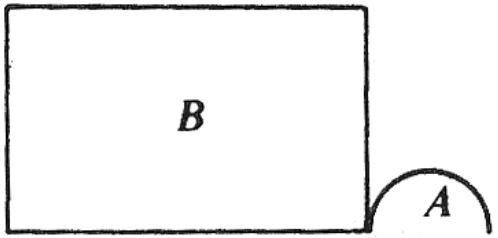
- **DEPART**
  - A DEPART Block registers statistics which indicate a reduction in the content of a Queue Entity.
    - DEPART A,B
- Operands
  - A - Queue Entity name or number. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA or SNA*Parameter.

    B - Number of units by which to decrease content of the Queue Entity. Default value is 1. Optional. The operand must be Null, Name, PosInteger, String, ParenthesizedExpression, SNA or SNA*Parameter.

Example

DEPART WaitingLine

In this example the content of the Queue Entity named WaitingLine is reduced by one and the associated statistics accumulators are updated.



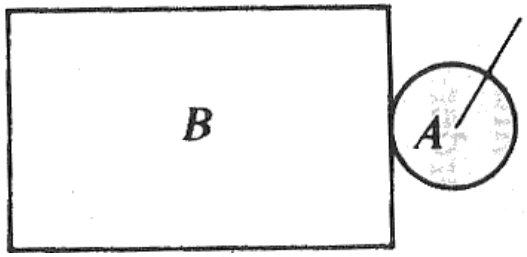
- **ENTER**
  - When a Transaction attempts to enter an ENTER Block, it either takes or waits for a specified number of storage units.
    - ENTER A,B
- Operands
  - A - Storage Entity name or number. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA or SNA*Parameter

B - Number of units by which to decrease the available storage capacity. Default value is 1. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA or SNA*Parameter

Example

ENTER Toolkit, 2

In this example the Active Transaction demands 2 storage units from the storage units available at the Storage Entity named Toolkit. If there are not enough storage units remaining in the Storage Entity, the Transaction comes to rest on the Delay Chain of the Storage Entity.



QUEUE

A QUEUE Block updates Queue Entity statistics to reflect an increase in content.

QUEUE A,B

Operands

A - Queue Entity name or number. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

B - Number of units by which to increase the content of the Queue Entity. Default value is 1. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

Example

QUEUE WaitingLine

In this example the content of the Queue Entity named WaitingLine is increased by one and the associated statistics accumulators are updated.

RELEASE

A RELEASE Block releases ownership of a Facility, or removes a preempted Transaction from contention for a Facility.

RELEASE A

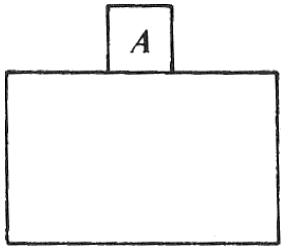
Operand

A - Facility number. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

Example

RELEASE Teller1

In this example, when a Transaction which owns the Facility Entity named Teller1 enters the RELEASE Block, it gives up ownership to the Facility.



PRIORITY

A PRIORITY Block sets the priority of the Active Transaction.

PRIORITY A,B

Operands

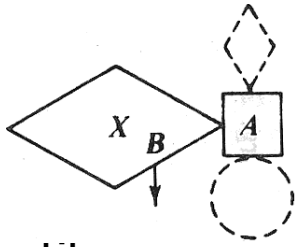
A - New priority value. Required. The operand must be integer, String, ParenthesizedExpression, SNA, or SNA*Parameter.

B - Buffer option. Places Active Transaction behind priority peers on CEC. Optional. The operand must be BU or Null

Example

PRIORITY 10

In this example any entering Transaction is made to have a priority of 10.



GATE

A GATE Block alters Transaction flow based on the state of an entity.

GATE O A,B

Operands

O - Conditional operator. Condition required of entity to be tested for successful test. Required. The operator must be FNV, FV, I, LS, LR, M, NI, NM, NU, SE, SF, SNE, SNF, SNV, SV, or U.

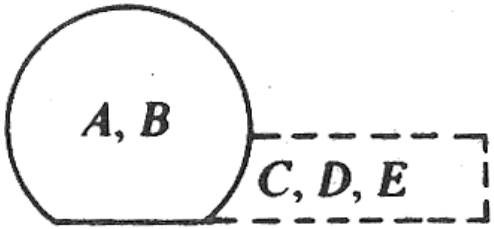
A - Entity name or number to be tested. The entity type is implied by the conditional operator. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter

B - Destination Block number when test is unsuccessful. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter

Examples

GATE SNF MotorPool

In this example of a "Refuse Mode" GATE Block, the Active Transaction enters the GATE Block if the Storage Entity named MotorPool is not full (i. e. if at least 1 unit of storage is available). If the Storage is full, the Active Transaction is blocked until 1 or more storage units become available.



GENERATE

A GENERATE Block creates Transactions for future entry into the simulation.

GENERATE A,B,C,D,E

Operands

A - Mean inter generation time. Optional. The operand must be Null, Name, Number, String, ParenthesizedExpression, or DirectSNA. You may not use Transaction Parameters.

B - Inter generation time half-range or Function Modifier. Optional. The operand must be Null, Name, Number, String, ParenthesizedExpression, or DirectSNA. You may not use Transaction Parameters.

C - Start delay time. Time increment for the first Transaction. Optional. The operand must be Null, Name, Number, String, ParenthesizedExpression, or DirectSNA. You may not use Transaction Parameters.

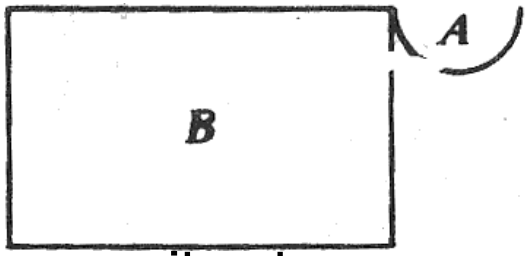
D - Creation limit. The default is no limit. Optional. The operand must be Null, Name, PosInteger, String, ParenthesizedExpression, or DirectSNA. You may not use Transaction Parameters.

E - Priority level. Optional. Zero is the default. The operand must be Null, Name, integer, String, ParenthesizedExpression, or DirectSNA. You may not use Transaction Parameters.

Example

GENERATE 0.1

This is the simplest way to use the GENERATE Block. This Block causes a priority zero Transaction to enter the simulation every tenth of a time unit.



LEAVE

A LEAVE Block increases the accessible storage units at a Storage Entity.

LEAVE A,B

Operands

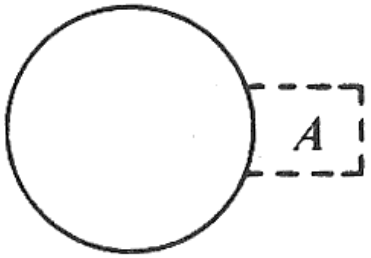
A - Storage Entity name or number. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter

B - Number of storage units. The default is 1. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

Example

LEAVE RepairMen,10

In this example, when a Transaction enters the LEAVE Block, the available storage units at the Storage Entity named RepairMen is increased by 10.



TERMINATE

A TERMINATE Block removes the Active Transaction from the simulation and optionally reduces the Termination Count.

TERMINATE A

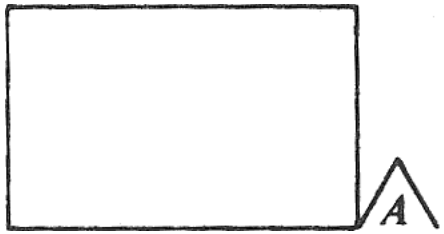
Operand

A - Termination Count decrement. Default is 0. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

Example

TERMINATE 1

In this example, when a Transaction enters the TERMINATE Block it is removed from the simulation. Also, the Termination Count of the simulation, which is set by a START Command is decremented by 1.



SEIZE

When the Active Transaction attempts to enter a SEIZE Block, it waits for or acquires ownership of a Facility Entity.

SEIZE A

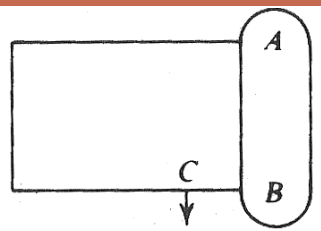
Operand

A - Facility name or number. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter

Example

SEIZE Teller1

In this example, when a Transaction attempts to enter the SEIZE Block, the state of the Facility named Teller1 is tested. If it is idle, ownership is given to the Active Transaction, which is allowed to enter the SEIZE Block and proceed to the Next Sequential Block (NSB). If the Facility is busy (owned), the Active Transaction comes to rest on the Delay Chain of the Facility.



LINK

A LINK Block controls the placement of the Active Transaction on the User Chain of a Userchain Entity.

LINK A,B,C

Operands

A - Userchain number. The Userchain Entity which may receive the entering Transaction. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

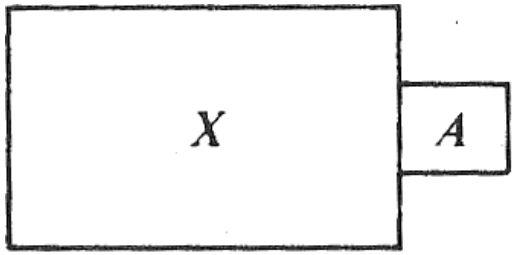
B - Chain ordering. The placement of new Transactions on the Userchain. Required. The operand must be LIFO, FIFO, ParenthesizedExpression, SNA, or SNA*Parameter.

C - Next Block location. The destination Block for Transactions which find the Link Indicator of the Userchain in the off state (reset). Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

Example

LINK OnHold,FIFO

In this example, the Active Transaction is placed at the end of the User Chain Entity named OnHold. It is removed from all chains except Transaction Groups and Interrupt Chains. In other words, preemptions are not cleared. The Transaction remains on the User Chain until some other Transaction enters an UNLINK Block and specifically removes it. In the present example, the Transaction is placed at the end of the User Chain named OnHold.



LOGIC

A LOGIC Block changes the state of a Logicswitch entity.

LOGIC O A

Operands

O - Logic operator. Required. The operator must be S, R, or I.

A - Logicswitch Entity number. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

Example

LOGIC S PowerSwitch

In this example, the Logicswitch Entity named PowerSwitch is left in the true or "set" state.

MARK

A MARK Block places an absolute clock time stamp into the Active Transaction or into its Parameter.

MARK A

Operand

A - Parameter number. Parameter to receive value of system clock. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, SNA*Parameter

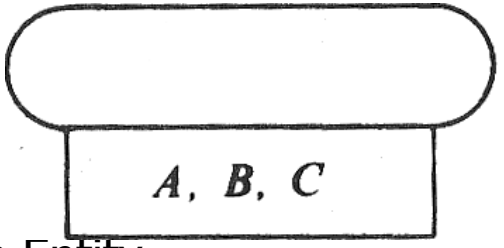
Examples

MARK Beginning

In this example, when a Transaction enters the MARK Block, its Transaction Parameter named Beginning is given a value equal to the value of the absolute system clock, AC1.

MARK

In this example, when a Transaction enters the MARK Block, its Mark Time is set equal to the value of the absolute system clock.



SAVEVALUE

A SAVEVALUE Block changes the value of a Savevalue Entity.

SAVEVALUE A,B

Operands

A - Savevalue Entity number. Required. May be followed by + or - to indicate addition or subtraction to existing value. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, SNA*Parameter, followed by +, -, or Null

B - The value to be stored, added, or subtracted. Required. The operand must be Name, Number, String, ParenthesizedExpression, SNA, SNA*Parameter

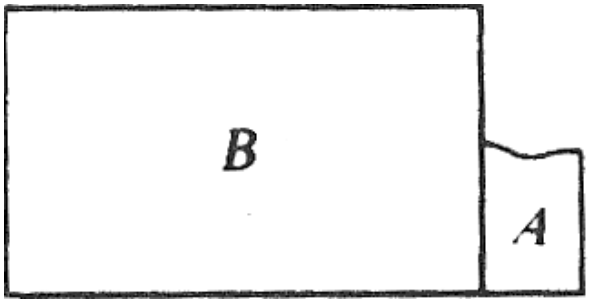
Examples

SAVEVALUE Account,99.95

In this example, the Savevalue Entity named Account takes on the value 99.95.

SAVEVALUE The_Bard,"A rose by any other name ..."

In this example, the Savevalue Entity named The_Bard is assigned a string. If the Savevalue Entity does not exist, it is created.



TABULATE

A TABULATE Block triggers the collection of a data item in a Table Entity.

TABULATE A,B

Operands

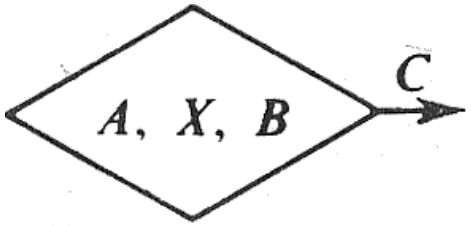
A - Table Entity name or number. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter

B - Weighting factor. Optional. The operand must be Null, Name, Number, ParenthesizedExpression, SNA, or SNA*Parameter

Example

TABULATE Sales

When a Transaction enters this TABULATE Block, the Table Entity named Sales is found. Sales must have been defined in a TABLE Command. Then the statistics associated with the table are updated with no weighting.



TEST

A TEST Block compares values, normally SNAs, and controls the destination of the Active Transaction based on the result of the comparison.

TEST O A,B,C

Operands

O - Relational operator. Relationship of Operand A to Operand B for a successful test. Required. The operator must be E, G, GE, L, LE, or NE.

A - Test value. Required. The operand must be Name, Number, String, ParenthesizedExpression, SNA, or SNA*Parameter.

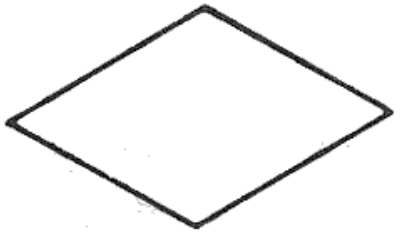
B - Reference value. Required. The operand must be Name, Number, String, ParenthesizedExpression, SNA, or SNA*Parameter.

C - Destination Block number. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

Example

TEST G C1, 70000

In this example of a "Refuse Mode" TEST Block, the Active Transaction enters the TEST Block if the relative system clock value is greater than 70000. Otherwise, the Transaction is blocked until the test is true.



TRANSFER

A TRANSFER Block causes the Active Transaction to jump to a new Block location.

TRANSFER A,B,C,D

Operands

A - Transfer Block mode. Described below. Optional. The operand must be BOTH, ALL, PICK, FN, P, SBR, SIM, fraction, Name, PosInteger, ParenthesizedExpression, SNA, SNA*Parameter, or Null

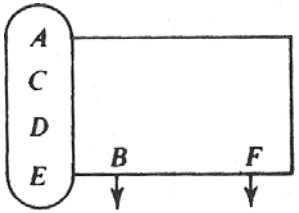
B - Block number or location. Parameter name or number when in P Mode. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, SNA*Parameter

C - Block number or location. Increment value in FN or P Mode. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, SNA*Parameter

D - Block number increment for ALL Mode. Default is 1. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, SNA*Parameter

Example:

Look into the modes.



UNLINK

An UNLINK Block removes Transactions from the User Chain of a Userchain Entity.

UNLINK O A,B,C,D,E,F

Operands

O - Relational operator. Relationship of D to E for removal to occur. These choices are explained below. Optional. The operator must be Null, E, G, GE, L, LE or NE.

A - User Chain number. User Chain from which one or more Transactions will be removed. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

B - Block number. The destination Block for removed Transactions. Required. The operand must be Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

C - Removal limit. The maximum number of Transactions to be removed. If not specified, ALL is used. Optional. The operand must be ALL, Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

D - Test value. The member Transaction Parameter name or number to be tested, a Boolean variable to be tested, or BACK to remove from the tail of the chain. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter, or BACK.

E - Reference value. The value against which the D Operand is compared. Optional. The operand must be Null, Name, Number, String, ParenthesizedExpression, SNA, or SNA*Parameter. Operand E is not used if Operand D is a Boolean Variable.

F - Block number. The alternate destination for the entering Transaction. Optional. The operand must be Null, Name, PosInteger, ParenthesizedExpression, SNA, or SNA*Parameter.

Example

UNLINK OnHold,Reentry,1

This is the simplest way to use the UNLINK Block. The first Transaction at the head of the Userchain Entity named OnHold, if any, is taken off the chain and is directed to the Block labeled Reentry. It is put on the CEC behind Transactions of the same priority. The Transaction entering the UNLINK Block proceeds to the Next Sequential Block.

GPSS Control Statements

CLEAR

A CLEAR Command returns the simulation to the unused state.

CLEAR A

Operand

A - ON or OFF. If the A Operand is omitted, ON is assumed. Optional. The operand must be ON, OFF or Null.

END

The ENDControl Statement has been replaced by EXIT, which can terminate a Session. END is now a keyword in the PLUS Language.

FUNCTION

A FUNCTION Command defines the rules for a table lookup.

There are several types of Function Entities. Each has its own rules pertaining to the table lookup. For each, the lookup table is specified in one or more Function Follower Statements. Type C Functions are a special case. They use a table lookup, followed by a linear interpolation.

The use of Function Commands to define probability distributions has been largely supplanted by the built-in distributions in the Procedure Library. This is discussed in Chapter 8. The old Function Types are still supported by GPSS World.

NAME FUNCTION A,B

Label / Operands

NAME - Entity Label this entity. Required. The field must be Name.

A - **Function argument. Required**. The operand must be Name, PosInteger, String, ParenthesizedExpression, SNA, or SNA*Parameter.

B - **Function type (one letter)** followed immediately by the number of data pairs in the

INITIAL

An INITIAL Command initializes a Matrix Entity, a Logicswitch Entity, Savevalue Entity, or an element of a Matrix Entity.

INITIAL A,B

Operands

A - Logicswitch, Savevalue, or Matrix element specified as SNA, or the name of a Matrix Entity. Operand A must have the form of an LS, X, or MX class SNA, or a Matrix Name. Required. The operand must be Name, LSPosInteger, LS$Name, XPosInteger, X$Name, MXPosInteger(m,n) or MX$ Name(m,n). Coordinates (m,n) must be Name or PosInteger.

B - Value to be assigned, or "UNSPECIFIED" The default is 1. Optional. The operand must be Null, Number, String, Name, or UNSPECIFIED.

GPSS Control Statements

RESET

A RESET Command marks the beginning of a measurement period.

RESET

Operands

None.

START

A START Command begins a simulation.

START A,B,C,D

Operands

A - Termination count. Required. The operand must be PosInteger.

B - Printout operand. NP for "no printout". Default is to print a standard report. Optional. The operand must be NP or Null.

C - Not used. Kept for compatibility with older dialects of GPSS.

D - Chain printout. 1 to include the CEC and FEC in the standard report. Optional. The operand must be Null, or PosInteger.

STORAGE

A STORAGE Command defines a Storage Entity.

NAME STORAGE A

Label / Operand

NAME - Entity Label for this entity. Required. The field must be Name.

A - Total storage capacity. Required. The operand must be PosInteger.

Example

MotorPool STORAGE 20

This Command defines a Storage Entity named MotorPool with a total capacity of 20 units.

Note Storage name must be defined before the simulation program and name of storage must start with 3 alphabets.

TABLE

A TABLE Command initializes a frequency distribution table.

NAME TABLE A,B,C,D

Label / Operands

NAME - Entity Label for this entity. Required. The field must be Name. The length of a Table name is limited to 32 characters.

A - Table argument. The data item whose frequency distribution is to be tabulated. Optional. The operand must be Name, Number, String, ParenthesizedExpression, or SNA. Ignored by ANOVA, but must be specified when used by TABULATE Blocks.

B - Upper limit of first frequency class. The maximum argument which causes the first frequency class to be updated. Required. The operand must be Number or String.

C - Size of frequency classes. The difference between the upper limit and lower limit of each frequency class. Required. The operand must be Number or String.

D - Number of frequency classes. Required. The operand must be PosInteger.

Entities

Transaction entities:

GENERATE, SPLIT, TRANSFER, TERMINATE ..

Facilities entities:

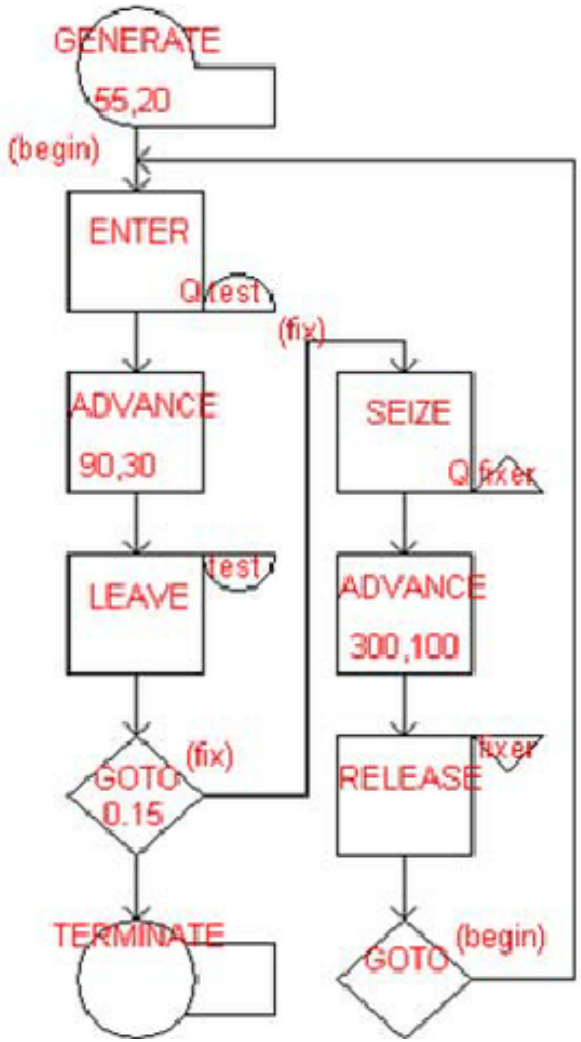
SEIZE, RELEASE ..

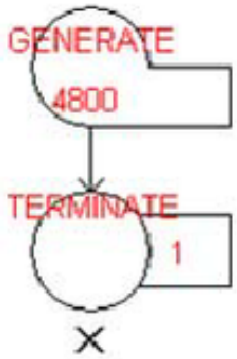
Queue entities:

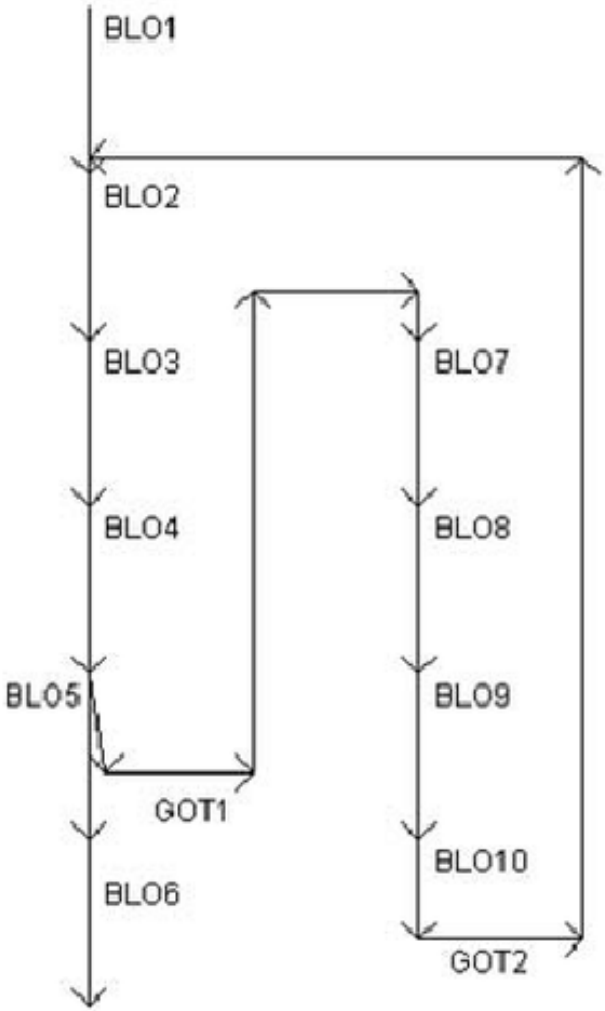
QUEUE, DEPART

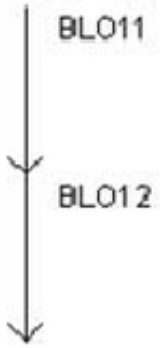
Storage entities:

ENTER, LEAVE


GENERATE
55,20
(begin)
ENTER
Qtest
(fix)
ADVANCE
90,30
LEAVE
test
GOTO
0.15
(fix)
TERMINATE
SEIZE
Qfixer
ADVANCE
300,100
RELEASE
fixer
GOTO
(begin)


GENERATE
4800
TERMINATE
1


BLO1
BLO2
BLO3
BLO4
BLO5
GOT1
BLO6
BLO7
BLO8
BLO9
BLO10
GOT2


BLO11
BLO12

A barber shop simulator

We are modeling a barber shop with the following qualities:

1. The shop contains one barber and one barber's chair, open for eight hours in a day.
2. Customers arrive on average every 18 minutes, with the arrival time varying between 12 and 24 minutes.
3. If the barber is busy, the customer will wait in a queue.
4. Once the barber is free, the next customer will have a haircut.
5. Each haircut takes between 12 and 18 minutes, with the average being 15 minutes.
6. Once the haircut is done, the customer will leave the shop.

A barber shop simulator

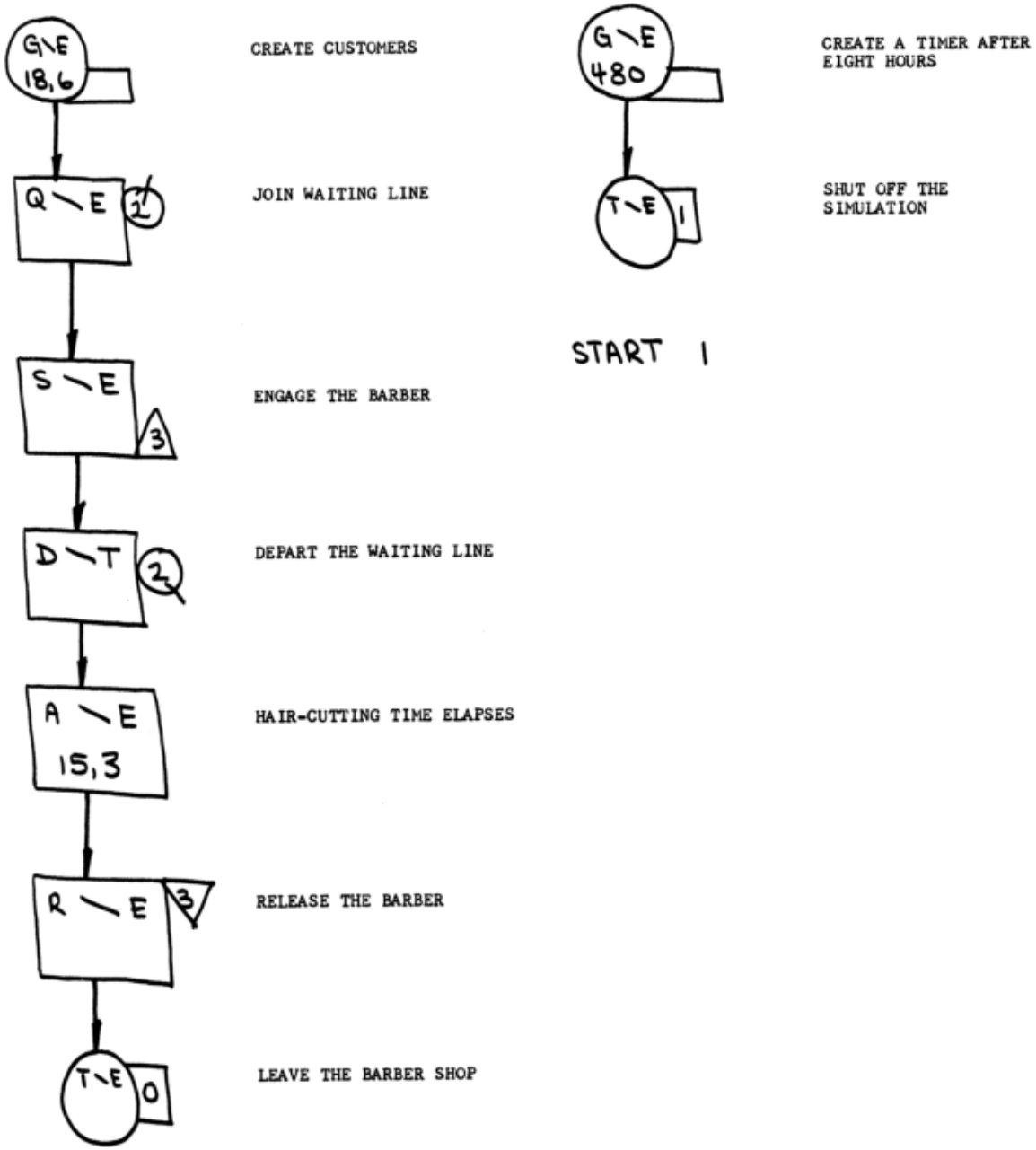
We want to answer these questions:

- How utilised is the barber through the day?
- How long does the queue get?
- On average, how long does a customer have to wait.

A barber shop simulator

```
SIMULATE
GENERATE 18,6          CREATE CUSTOMERS
QUEUE 2                CUSTOMERS QUEUE UP IF NECESSARY
SEIZE 3                ENGAGE THE BARBER WHEN HE BECOMES AVAILABLE
DEPART 2               CUSTOMER LEAVES THE QUEUE
ADVANCE 15,3           CUSTOMER GETS HIS HAIR CUT
RELEASE 3              RELEASE THE BARBER
TERMINATE 0            LEAVE THE BARBER SHOP

GENERATE 480           GENERATE A TIMER AFTER 8 HOURS OF SIMULATED TIME
  *
TERMINATE 1            SHUT OFF THE RUN
START 1                CARRY OUT THE SIMULATION
END                    RETURN CONTROL TO THE OPERATING SYSTEM
```


G\E
18,6
CREATE CUSTOMERS
Q\E
1
JOIN WAITING LINE
S\E
3
ENGAGE THE BARBER
D\T
2
DEPART THE WAITING LINE
A\E
15,3
HAIR-CUTTING TIME ELAPSES
R\E
3
RELEASE THE BARBER
T\E
0
LEAVE THE BARBER SHOP
G\E
480
CREATE A TIMER AFTER
EIGHT HOURS
T\E
1
SHUT OFF THE
SIMULATION
START 1

A barber shop simulator

- **GENERATE 18,6** means generate a transaction - a barber shop customer - every 18 minutes ± 6 minutes.
- **QUEUE 2** defines a queue with ID 2, denoting the queue where customers will wait.
- **SEIZE 3** defines a facility with ID 3. The facility is the barber and this line means if the barber is free, the next customer occupies the barber until released.
- **DEPART 2** says that the customer leaves the queue when occupying the barber.
- **ADVANCE 15,3** means that transactions in this state only move on after 15 minutes ± 3 minutes - modeling the time taken for a haircut. After that
- **RELEASE 3** shows that the customer no longer occupies the barber and
- **TERMINATE 0** ends the transaction, showing that the customer has left the shop.

A barber shop simulator

- That is all that is needed for the basic simulation, but if run like this it would never stop, as we have not modeled the 8 hour period the shop is open.
- To do this we generate a new transaction with
  - **GENERATE 480**, which means generate a transaction after 480 minutes, ie 8 hours. The next line,
  - **TERMINATE 1**, stops the simulation after this transaction is generated.

Finished Unit 8