SIMULATION AND MODELLING

Sujan Karki

Email: sujan074bct045@ioepc.edu.np

Contact No.: 9819387234

Master's in Information System Engineering (MscIne) *Ongoing
Bachelor in Computer Engineering – Purwanchal Campus,IOE

UNIT 9

Simulation software

1. ***Simulation software (3 hours)***

a. simulation in Java

b. simulation in GPSS

c. Simulation in SSF

d. Other simulation software

Simulation Software

- *Simulation software is a program that allows the user to observe an operation through simulation without actually performing the real operation.*

- *Simulation software can be used to study wide variety of systems, including manufacturing and logistics systems, energy systems, transportations systems, healthcare systems, and many others.*

- *It can be used to perform Modelling, Experimentation, Analysis.*

- *Some examples of simulation software includes, Arena, Extendsim, Simio, Anylogic, Simpy, NetLogo, Vensim, Simulink, GPSS etc.*

Simulation Tools

- *Simulation Tools are software programs that are used to create and run models of systems to study and analyze their behaviours.*
- *Software used to develop simulation models can be divided into three categories:*

  1. General-purpose simulation software: designed to be used for modeling and simulating a wide range of systems. Examples: Arena, AnyLogic, and Simulink.

  2. Hardware simulation tools: designed specifically for modeling and simulating hardware systems, such as computer systems or manufacturing equipment. Examples SPICE, VHDL, and Verilog.

  3. Software simulation tools: designed specifically for modeling and simulating software systems, including applications, operating systems, and other software. Examples include UML tools like Rational Rose and simulation software packages like Simula.

  4. Network simulation tools: designed specifically for modeling and simulating networked systems, including communication networks, transportation networks, and other types of systems.

- *Examples include NS-2, NetSim, Packet Tracer, etc.*

Selection of Simulation Software

1. *Model Building features: Simulation programming, Input flexibility, Randomness, Syntax, Process interaction , event perspective, User built custom object, Interface with general purpose programming language etc.*
2. *Runtime Environment: Execution speed, debugging, Model size, number of variables and attributes, etc.*
3. *Animation of layout features: Types of animation, Dimension(2D, 3D etc.) Quality of motion, Views, Navigation, Hardware requirements(like video cards, RAM) , Libraries etc.*
4. *Output features: Run manager, Independent replications, Optimization, Standard and customized reports, statistical analysis, File export etc.*
5. *Vender support and production documentation: Training, documentation, Help and support system, tutorials, Upgrade , Maintenance, Track record etc.*

Advice when evaluating and selecting simulation software:

➢*Do not focus on single issue, such as ease of use. Consider the accuracy and level of detail obtainable, ease of learning, vendor support, and applicability to your applications.*

➢*Execution speed is important. Speed affects development time.*

➢*Beware of advertising claims and demonstrations.*

➢*Ask the vendor to solve a small version of your problem.*

➢*Beware of "checklists" with "yes" and "no" as the entries, e.g. many packages claim to have a conveyor entity, however, implementations have considerable variation and level of fidelity.*

➢*Determine whether the simulation package and language are sufficiently powerful to avoid having to write logic in any external language.*

➢*Beware of "no programming required," unless either the package is a near-perfect fit to your problem domain, or programming is possible with the supplied blocks, nodes, or process-flow diagram.*

Simulation in Java

- *Java is widely used programming language that has been used extensively in simulation.*
- *It does not provide any facilities, tools or features designed to assist simulation analysts directly.*
- *The runtime library provides a random-number generator.*
- *It supports modular construction of large models.*
- *Simulation libraries such as SSG alleviate the development burden.*

  - Provides access to standardized simulation functionality and hide low-level scheduling minutiae.

Simulation in Java

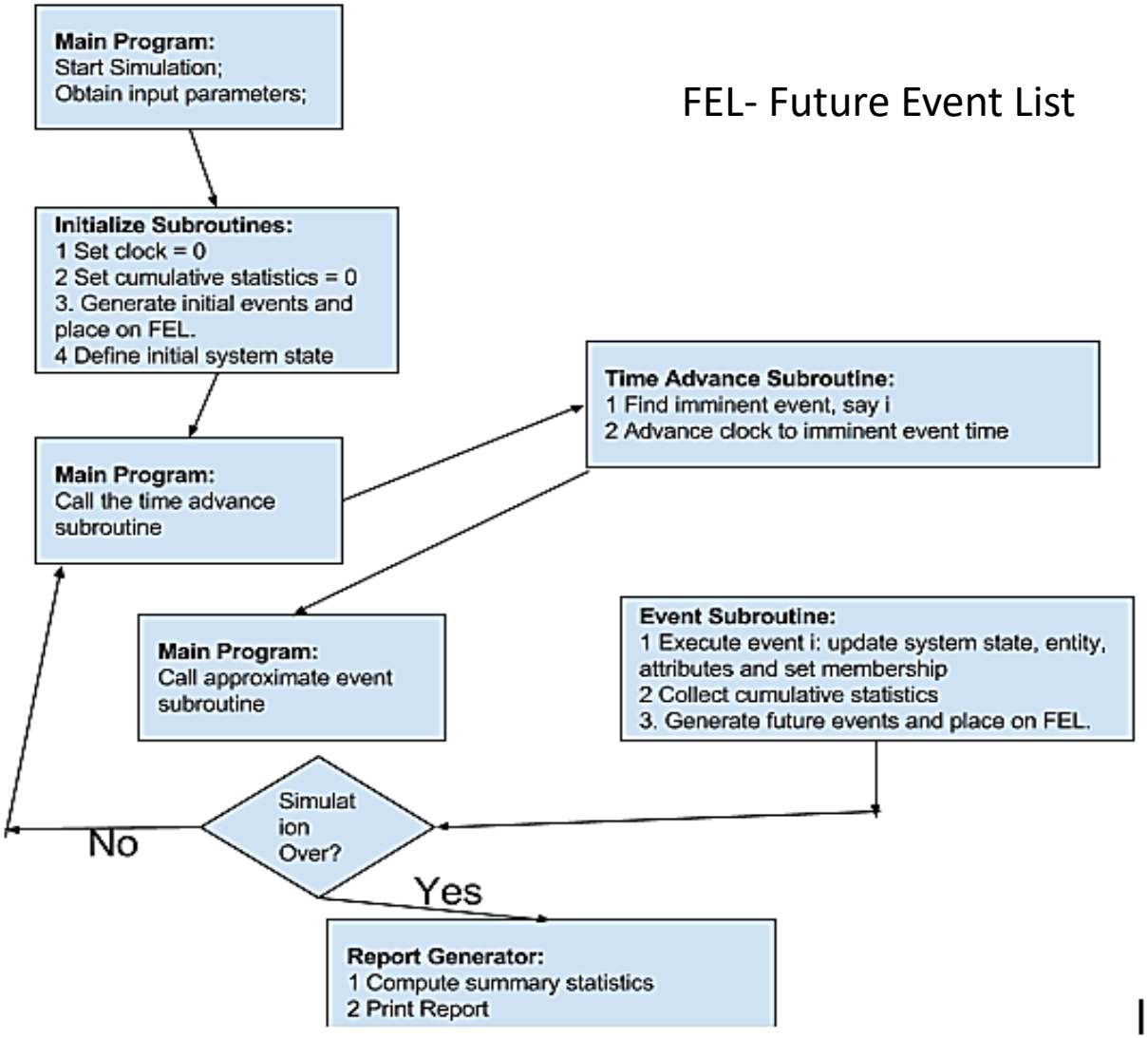
- *Discrete-event simulation model written in Java contains the following :*

1. *Basic components:*
   - *System state*
   - *Entities and attributes*
   - *Events*
   - *Activities*
   - *Delay*
2. *Common components (when organizing model in a modular fashion by using methods):*
   - *Clock*
   - *Initialization method*
   - *Min-time event method*
   - *Event methods*
   - *Random-variate generators*
   - *Main program*
   - *Report generator*

- *Clock: It is a variable that defines the simulated time.*
- *Initialization method: It is a method to define the system state at initial time.*
- *Min-time event method: It is a method the defines the imminent(about to happen) event.*
- *Event methods: It is a method for each event that update the system state when it occurs.*
- *Random-variate generators: It is a method to generate random samples form the desired probability distributions.*
- *Main program: It is the core of the simulating system that controls the overall event scheduling algorithms.*
- *Report generator: It is a method that summarizes the collected statistics to give report at the end of the simulation.*

Simulation in Java

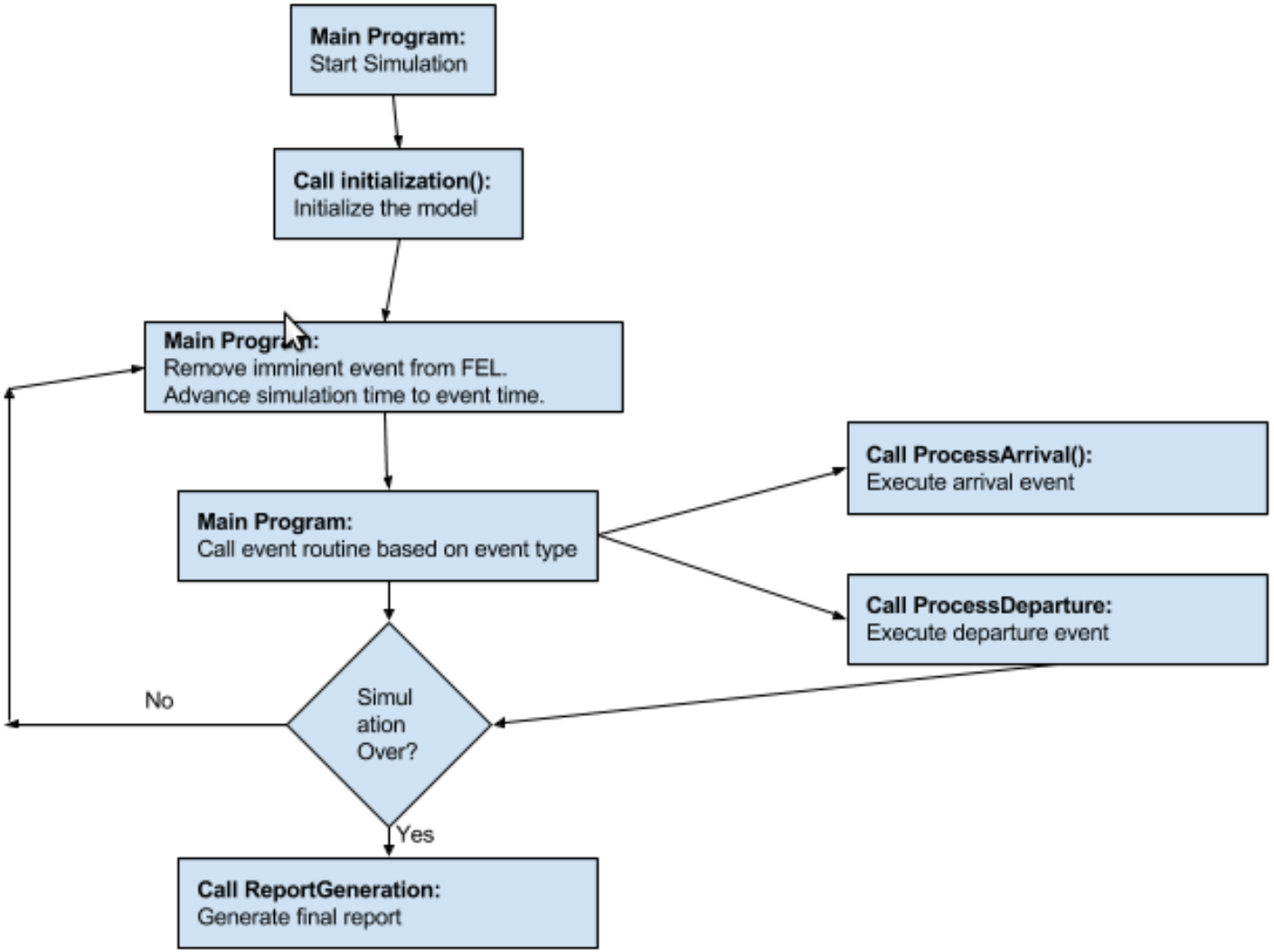
The overall structure of Simulation in Java:



FEL- Future Event List

Simulation in Java

The overall structure of Simulation in Java: Single server queue



Definition of variables, Functions and subordinate in the JAVA model of single-server queue:

1. ***Variables:***
   - ***a. System state:*** ***Queue length and number in service***
   - ***b. Entity and attributes :*** ***Customers- FCFA queue of customer***
   - ***c. FEL(future event list):*** ***Priority- ordered list of pending events***
   - ***d. Activity duration:*** ***Mean inter arrival time, Mean service time etc.***
2. ***Input Parameters:*** ***Mean inter arrival time, Mean service time, SIGMA(Standard deviation of service time), Total customers, Clock, Last event time, Total busy time of server, Max queue length, Number of departures, Long service(no. of customer who spent more time)***
3. ***Function:*** ***Exponential(function to generate exponential distribution), Normal distribution function (XMU(Mean) , SIGMA(standard deviation)***
4. ***Methods:***

   Initialization method,
   Process Arrival method
   Process departure method,
   Report generator method

Structure of the main program:

```
class Sim {

// Class Sim variables
public static double Clock, MeanInterArrivalTime, MeanServiceTime, SIGMA,
LastEventTime,
      TotalBusy, MaxQueueLength, SumResponseTime;
public static long  NumberOfCustomers, QueueLength, NumberInService,
      TotalCustomers, NumberOfDepartures, LongService;

public final static int arrival = 1;
public final static int departure = 2;

public static EventList FutureEventList;
public static Queue Customers;
public static Random stream;
```

Structure of the main program (continued):

… continued from last slide …

```
public static void main(String argv[]) {

  MeanInterArrivalTime = 4.5; MeanServiceTime = 3.2;
  SIGMA                = 0.6; TotalCustomers  = 1000;
  long seed            =1000;  //Long.parseLong(argv[0]);

  stream = new Random(seed);          // initialize rng stream
  FutureEventList = new EventList();
  Customers = new Queue();

  Initialization();
 // Loop until first "TotalCustomers" have departed
  while(NumberOfDepartures < TotalCustomers ) {
    Event evt = (Event)FutureEventList.getMin();  // get imminent event
    FutureEventList.dequeue();                    // be rid of it
    Clock = evt.get_time();                       // advance simulation time
    if( evt.get_type() == arrival ) ProcessArrival(evt);
    else  ProcessDeparture(evt);
    }
  ReportGeneration();
 }
```

Structure of the initialization method:

```
// seed the event list with TotalCustomers arrivals
 public static void Initialization()  {
  Clock = 0.0;
  QueueLength = 0;
  NumberInService = 0;
  LastEventTime = 0.0;
  TotalBusy = 0 ;
  MaxQueueLength = 0;
  SumResponseTime = 0;
  NumberOfDepartures = 0;
  LongService = 0;
  // create first arrival event
  Event evt = new Event(arrival, exponential( stream,
MeanInterArrivalTime));
  FutureEventList.enqueue( evt );
 }
```

Structure of the departure event method:

- Obtain the job at the head of the queue

```
public static void ScheduleDeparture() {

 double ServiceTime;
 // get the job at the head of the queue

 while (( ServiceTime = normal(stream, MeanServiceTime, SIGMA)) < 0 );
 Event depart = new Event(departure,Clock+ServiceTime);
 FutureEventList.enqueue( depart );
 NumberInService = 1;
 QueueLength--;
}
```

- Get the description of finishing customer
- Schedule departure of the next customer if queue is not emptied
- Collect statistics

```
public static void ProcessDeparture(Event e) {
 // get the customer description
 Event finished = (Event) Customers.dequeue();
 // if there are customers in the queue then schedule
 // the departure of the next one
  if( QueueLength > 0 ) ScheduleDeparture();
  else NumberInService = 0;
  // measure the response time and add to the sum
  double response = (Clock - finished.get_time());
  SumResponseTime += response;
  if( response > 4.0 ) LongService++; // record long service
  TotalBusy += (Clock - LastEventTime );
  NumberOfDepartures++;
  LastEventTime = Clock;
 }
```

Structure of the report generator:

```
public static void ReportGeneration() {
double RHO   = TotalBusy/Clock;
double AVGR  = SumResponseTime/TotalCustomers;
double PC4   = ((double)LongService)/TotalCustomers;


          System.out.println( "SINGLE SERVER QUEUE SIMULATION -
          GROCERY STORE CHECKOUT COUNTER ");
          System.out.println( "\tMEAN INTERARRIVAL TIME                    "
          + MeanInterArrivalTime );
          System.out.println( "\tMEAN SERVICE TIME                         "
          + MeanServiceTime );
          System.out.println( "\tSTANDARD DEVIATION OF SERVICE TIMES
          "
          + SIGMA );
```

```
          System.out.println( "\tNUMBER OF CUSTOMERS SERVED
          "
          + TotalCustomers );
          System.out.println();
          System.out.println( "\tSERVER UTILIZATION
          " + RHO );
          System.out.println( "\tMAXIMUM LINE LENGTH
          " + MaxQueueLength );
          System.out.println( "\tAVERAGE RESPONSE TIME
          " + AVGR + "  MINUTES" );
          System.out.println( "\tPROPORTION WHO SPEND FOUR ");
          System.out.println( "\t MINUTES OR MORE IN SYSTEM
          " + PC4 );
          System.out.println( "\tSIMULATION RUNLENGTH
          " + Clock + " MINUTES" );
          System.out.println( "\tNUMBER OF DEPARTURES
          " + TotalCustomers );
}
```

Sim class methods to generate exponential and normal random variates:

```
public static double exponential(Random rng, double mean) {
 return -mean*Math.log( rng.nextDouble() );
}
public static double SaveNormal;
public static int  NumNormals = 0;
public static final double  PI = 3.1415927 ;

public static double normal(Random rng, double mean, double sigma) {
      double ReturnNormal;       // should we generate two normals?
      if(NumNormals == 0 ) {
        double r1 = rng.nextDouble();
        double r2 = rng.nextDouble();
        ReturnNormal = Math.sqrt(-2*Math.log(r1))*Math.cos(2*PI*r2);
        SaveNormal   = Math.sqrt(-2*Math.log(r1))*Math.sin(2*PI*r2);
        NumNormals = 1;
      } else {
        NumNormals = 0;
        ReturnNormal = SaveNormal;
      }
      return ReturnNormal*sigma + mean ;
 }
```

The output:

SINGLE SERVER QUEUE SIMULATION - GROCERY STORE CHECKOUT COUNTER

MEAN INTERARRIVAL TIME	4.5	
MEAN SERVICE TIME	3.2	
STANDARD DEVIATION OF SERVICE TIMES	0.6	
NUMBER OF CUSTOMERS SERVED	1000	
SERVER UTILIZATION	0.7175	
MAXIMUM LINE LENGTH	7.0	
AVERAGE RESPONSE TIME	6.7358	MINUTES
PROPORTION WHO SPEND FOUR MINUTES OR MORE IN SYSTEM	0.675	
SIMULATION RUNLENGTH	4455.02	MINUTES
NUMBER OF DEPARTURES	1000	

- Note: Most of the output statistics are estimates that contain random error.

Simulation in GPSS

- *GPSS is a highly structured, special-purpose simulation programming language*
- *Based on the process-interaction approach.*
- *Oriented toward queueing systems.*
- *Use of block diagram:*
- *Provides a convenient way to describe the system.*
- *With-over 40 standard blocks.*
- *Blocks represents events, delays and other actions that affect transaction flow.*
- *Block diagram is converted to block statements, control statements are added, and result in a GPSS model.*
- *The 1st version was released by IBM in 1961.*
- *GPSS/H is the most widely used version today.*
- *Released in 1977*
- *Flexible yet powerful.*

Simulation in GPSS: Basic Commands

- *CLEAR - reset statistics and remove transaction*
- *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*

Simulation in GPSS: Blocks

1. GENERATE

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

GENERATE A,B,C,D,E

A- Mean inter generation time

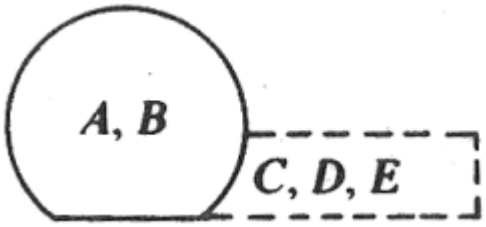
B- Inter generation time half range or function modifier(Change of A)

C- Start delay time

D- Creation limit

E- Priority level.

- *Example : GENERATE 5, 10*
- *This generates an entity every 10 time units, and the Inter generation time between two consecutive entities is 5 units.*



Simulation in GPSS: Blocks

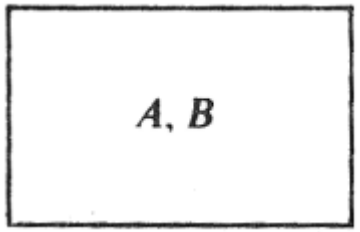
2. ADVANCE

- *An ADVANCE Block simulates a delay, advancing the simulation clock by a specified amount of time.*
- *ADVANCE A,B*

***A- Mean time** (req)*

B- Inter generation time half range or function modifier(Change of A)

- *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),*



Simulation in GPSS: Blocks

3. ASSIGN

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

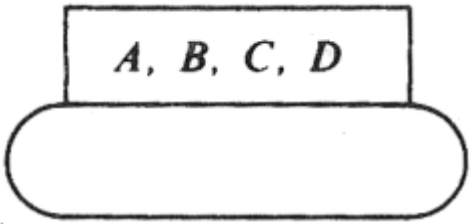
ASSIGN A,B,C;

A - Parameter number of the Active Transaction. (req)

B - Value. (req)

C - Function number

- *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*



Simulation in GPSS: Blocks

4. DEPART

A DEPART Block registers statistics which indicate a reduction in the content of a Queue Entity.

DEPART A,B

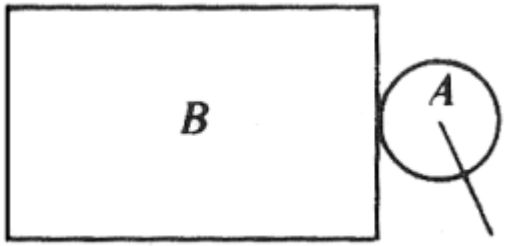
A - Queue Entity name or number. Required.

B- Number of units by which to decrease content of the Queue Entity. Default value is 1.

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.



Simulation in GPSS: Blocks

5. 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

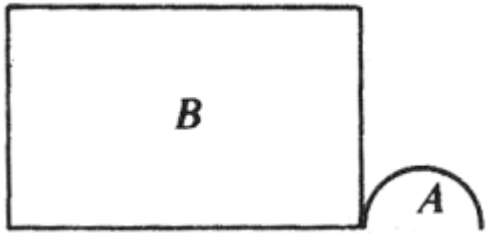
A - Storage Entity name or number. Required.

B- Number of units by which to decrease the available storage capacity. Default value is 1.

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.



Simulation in GPSS: Blocks

6. QUEUE

This block adds entities to a queue (waiting line) for resources or services.

QUEUE A,B

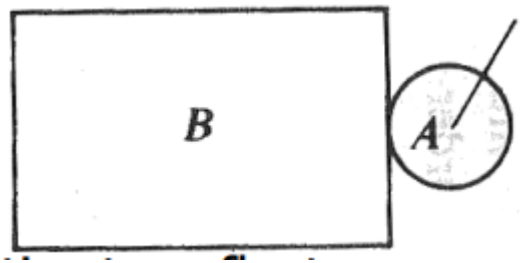
A - Queue Entity name or number. Required.

B- Number of units by which to increased content of the Queue Entity. Default value is 1.

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.



Simulation in GPSS: Blocks

7. RELEASE

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

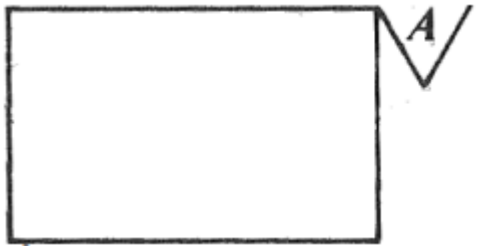
RELEASE A

A – Facility number (req)

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. .



Simulation in GPSS: Blocks

8. PRIORITY

A PRIORITY Block sets the priority of the Active Transaction.

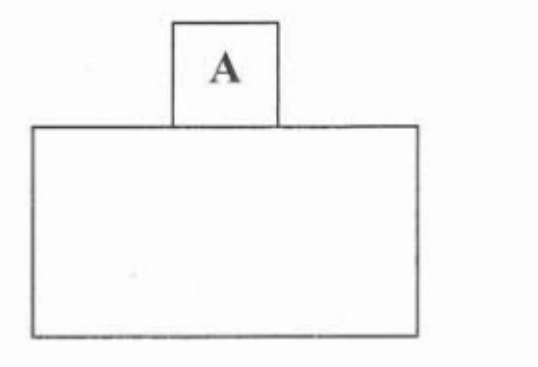
PRIORITY A

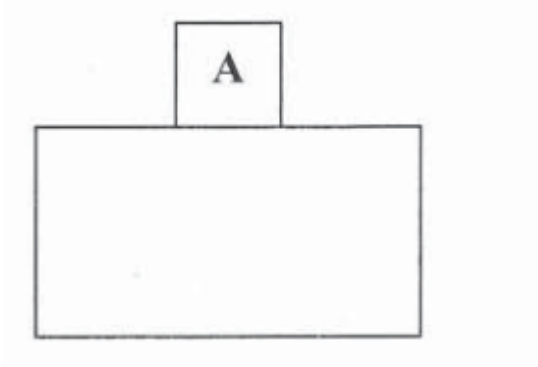
A - New priority value. Required.

Example:

PRIORITY 10

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





Simulation in GPSS: Blocks

9. LEAVE

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

LEAVE A,B

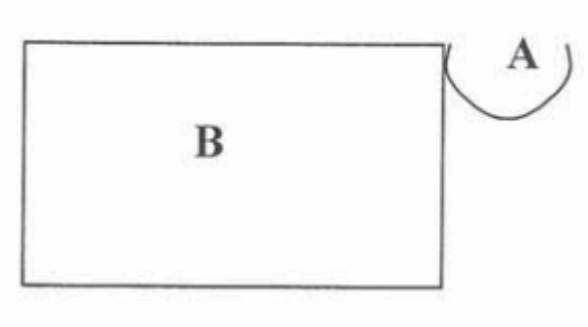
A - Storage Entity name or number.. Required.

B – Number of storage units. Default is 1.

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.



Simulation in GPSS: Blocks

10. TERMINATE

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

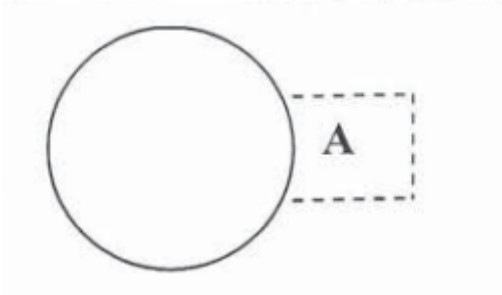
TERMINATE A

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

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.



Simulation in GPSS: Blocks

11. SEIZE

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

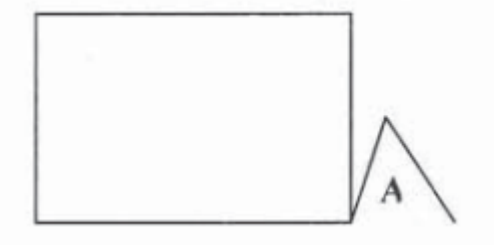
SEIZE A

A - - Facility name or number. Required.

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.



Simulation in GPSS: Blocks

12. MARK

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

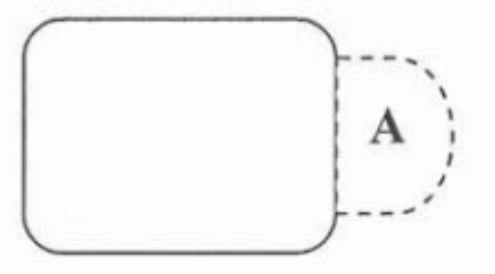
MARK A

A - Parameter number. Parameter to receive value of system clock.

Example:

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.



Simulation in GPSS: Blocks

13. TABULATE

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

TABULATE A,B

A - Table Entity name or number. Required.

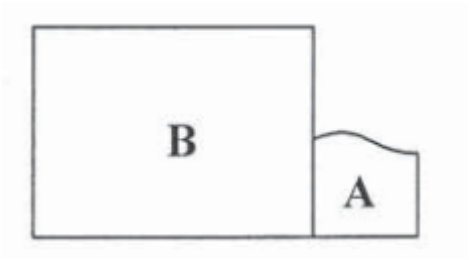
B - Weighting factor

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.



Simulation in GPSS: Blocks

14. TEST

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

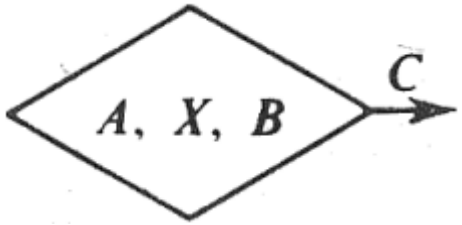
TEST O, A, B, C

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. B

A - Test value. Required.

B - Reference value. Required

C - Destination Block number.



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.

Simulation in GPSS: Blocks

14. TRANSFER

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

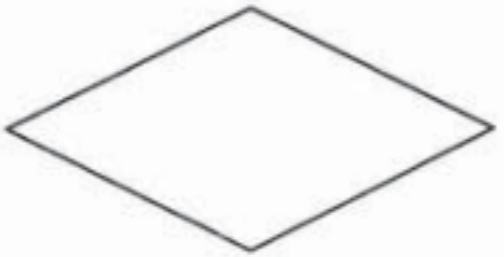
TRANSFER A,B,C,D

A - Transfer Block mode. Described below. Optional. The operand must be BOTH: Transfer control to the designated block when either of the specified conditions is true. ALL: Transfer control to the designated block when all of the specified conditions are true. PICK: Choose one of the specified blocks randomly based on probabilities. FN: Transfer control to the designated block based on a conditional function. P: Transfer control to the designated block based on a probability. SBR: Transfer control based on a subroutine call. SIM: Transfer control based on simulation time.

B - Block number or location. Parameter name or number when in P Mode.

C - Increment value in FN or P Mode.

D - Block number increment for ALL Mode. Default is 1.



Simulation in GPSS: Control Statements

1. CLEAR:

A CLEAR Command returns the simulation to the unused state.

CLEAR A

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

2. END

The END Control Statement has been replaced by EXIT, which can terminate a Session.

3. FUNCTION

A FUNCTION Command defines the rules for a table lookup.

NAME FUNCTION A,B

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

A - Function argument. Required.

B - Function type (one letter) followed immediately by the number of data pairs in the function. If the function type is "F" and it has 2 data pairs, it might be represented as "F2" in this context.

Entities

Transaction entities:

GENERATE, SPLIT, TRANSFER, TERMINATE ..

Facilities entities:

SEIZE, RELEASE ..

Queue entities:

QUEUE, DEPART

Storage entities:

ENTER, LEAVE

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.*

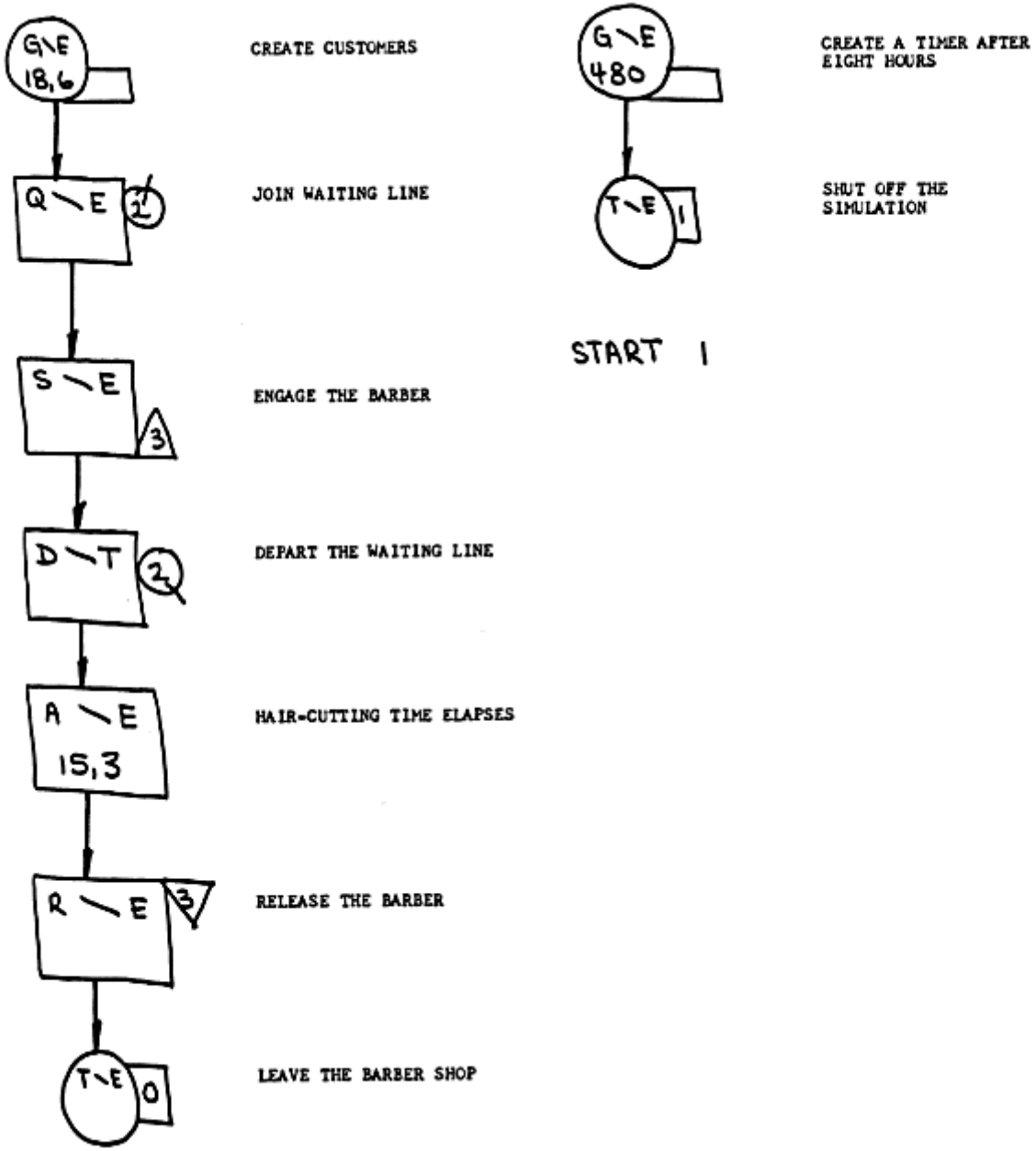
We want to answer these questions:

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

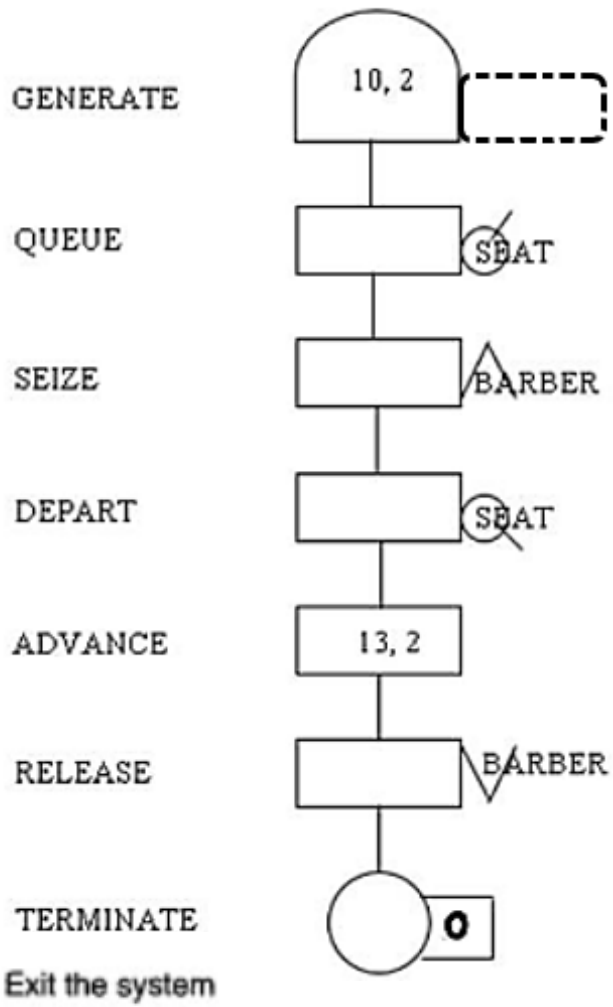
A barber shop simulator

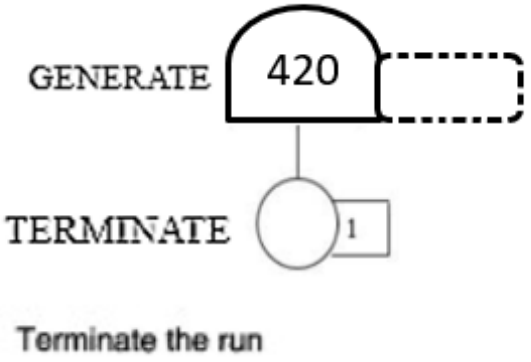
```
SIMULATE
GENERATE 18,6          CREATE CUSTOMERS
QUEUE 2                CUSTOMERS QUEUE UP IF NECESSARY
SEIZE 3                ENGAGE THE BARBER WHEN HE BECOMES AVAILABLE
DEPART 2               CUSTOMER LEAVES THE QUEUE
ADVANCE 15,3           CUSTOMER GETS HIS HAIR CUT
RELEASE 3              RELEASE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

1. *Create a GPSS model and program to simulate a barber shop for a day (9am to 4pm), where a costumer enters the Shop every 10 ± 2 minutes and a barber takes 13 ± 2 for a haircut.*





Program:

```
GENERATE 10,2
QUEUE SEAT
SEIZE BARBER
DEPART SEAT
ADVANCE 13,2
RELEASE BARBER
TERMINATE

TIMER GENERATE 420
TERMINATE 1
START 1
END
```

2. *A machine tool in a manufacturing shop is turning out parts at the rate of every 5 minutes. When they are finished, the parts are sent to an inspector, who takes 4±3 minutes to examine each one and rejects 15% of the parts. Draw and explain a block diagram and write a GPSS program to simulate using the concept of facility. and also explain the function of each block used in the block diagram in detail.*

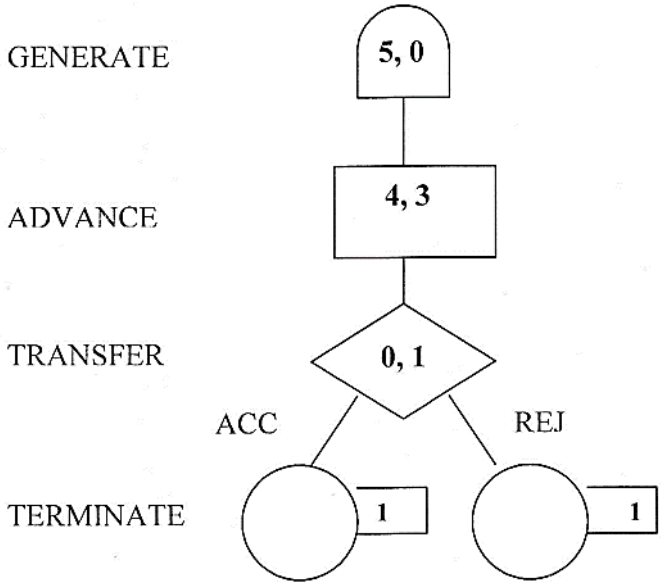


Fig: Manufacturing shop- model 1

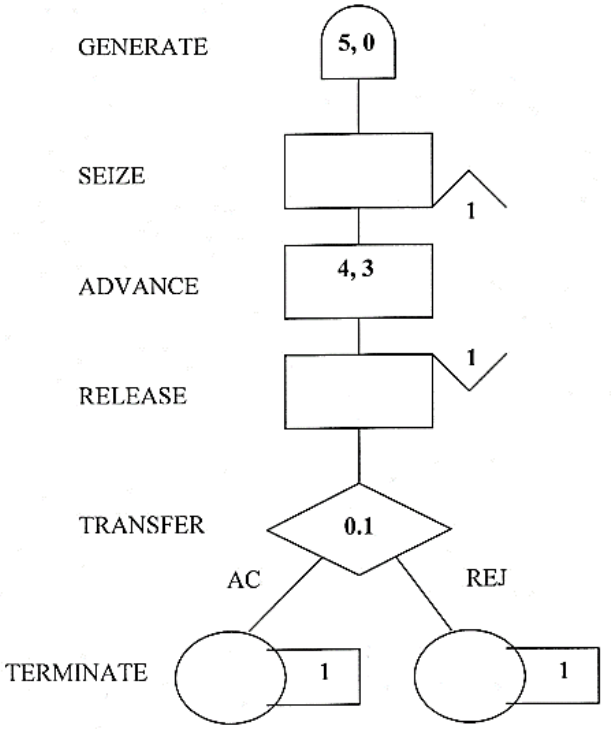


Fig: Manufacturing shop- model 2

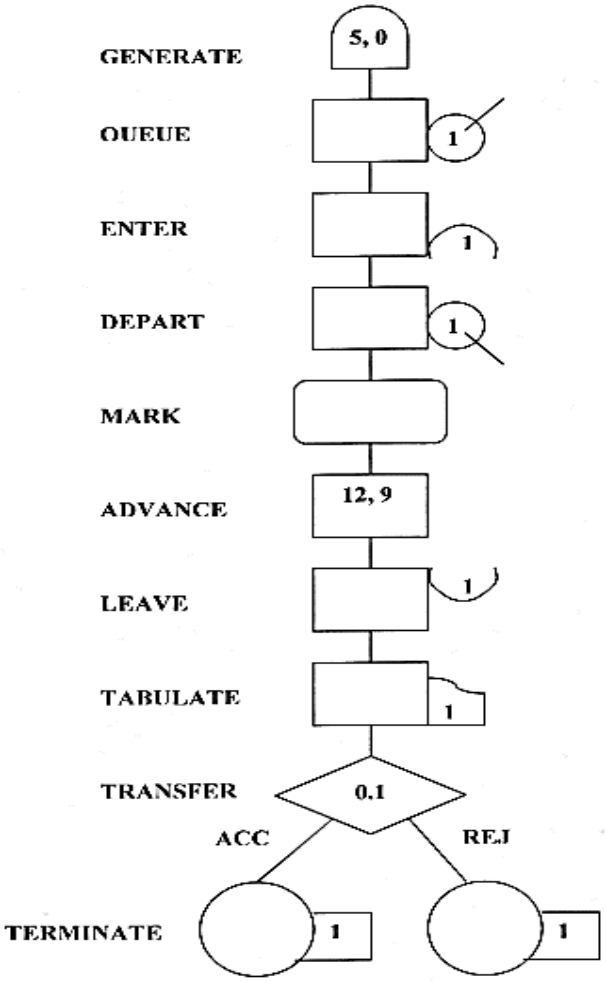
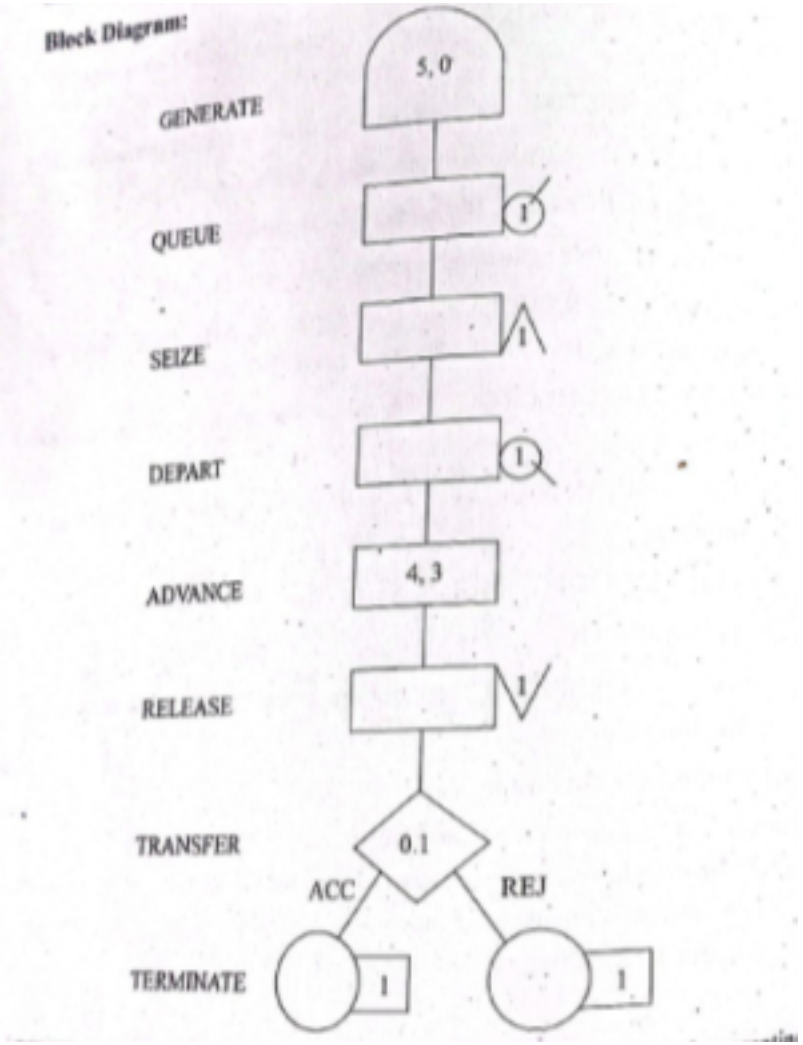


Fig: Manufacturing shop – model 3

Code:

```
GENERATE    5, 0
QUEUE       1
SEIZE       1
DEPART      1
ADVANCE     4, 3
RELEASE     1
TRANSFER 0.1 ACC  REJ
ACC TERMINATE   1
REJ TERMINATE   1
```

Block Diagram:

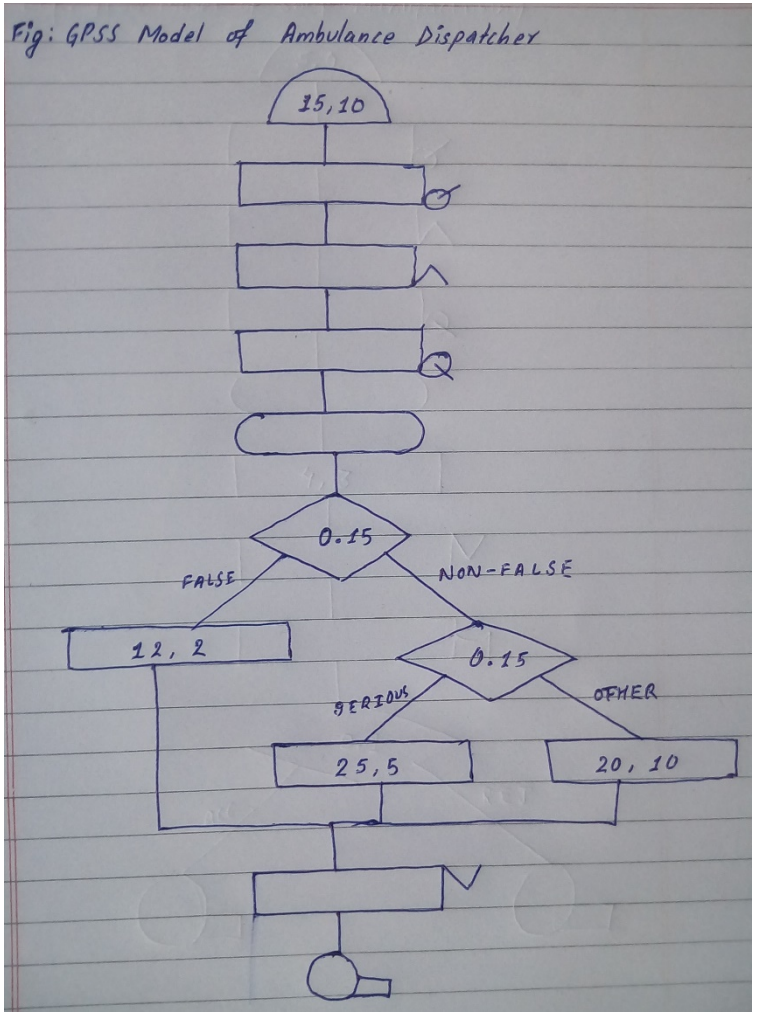


Code:

```
GENERATE    5, 0
QUEUE       1
SEIZE       1
DEPART      1
ADVANCE     4, 3
RELEASE     1
TRANSFER    0.1  ACC  REJ
ACC TERMINATE    1
REJ TERMINATE    1
```

Example 3 - Ambulances are dispatched at a rate of one every 15 (+ or -) 10 mins. Fifteen percent of the calls are false alarms, which require 12 (+ or -) 2 mins to complete. All other calls can be one of two kinds. The first kind are classified as serious. They constitute 15% of non false alarms and take 25 (+ or -) 5 mins to complete. The other calls take 20 (+ or -) 10 mins. Simulate the model using GPSS.

```
GENERATE 15, 10
QUEUE
SEIZE
DEPART
MARK
TRANSFER 0.15 FALSE NON-FALSE
FALSE ADVANCE 12, 2
NON-FALSE TRANSFER 0.15 SERIOUS OTHER
SERIOUS ADVANCE 25, 5
OTHER ADVANCE 20, 10
RELEASE
TERMINATE
```



1. Super Market GPSS:

Function number 1 of the simulation model controls the inter-arrival time at a GENERATE block with a mean of 36.

To represent the baskets, a storage denoted by BSKT is used with capacity equal to the number of baskets.

The block count at AWAY will show the number of customers turned away for lack of baskets.

The number of items is determined by an ASSIGN block which has SNA called FN2 in field B.

The counters are regarded as a service unit with five servers. They are represented by storage, named CKT that has a capacity of 5.

Field A of ADVANCE block representing the checking-out time is set to V1.

Transit time is tabulated in a table called TRT.

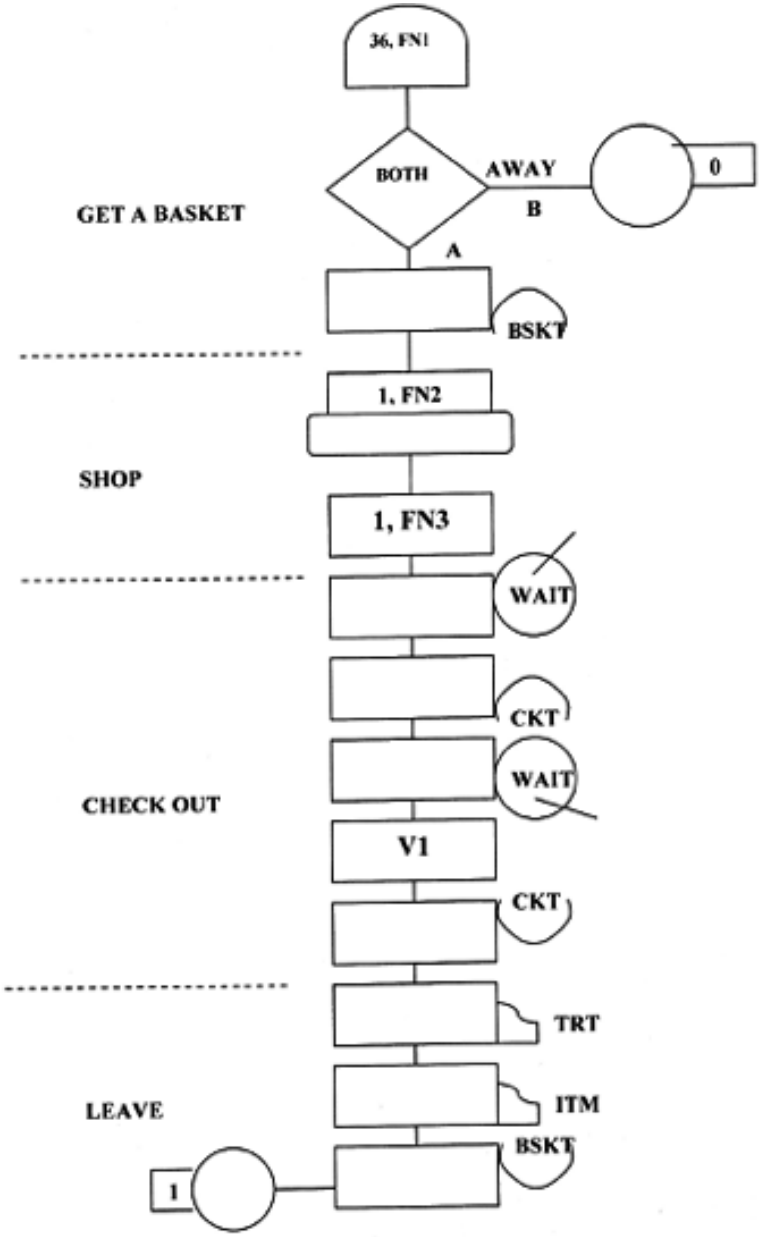


Fig: Simulation of a supermarket

Simulation in SSF

SSF (Scalable Simulation Framework)is a library used for building complex simulations for a wide variety of domains such as telecommunications, transportation, logistics, and network systems.

- *SSF is based on a modular approach, allowing developers to create and combine components like entities, processes, and communication channels.*
- *The API is sparse and allows implementations to achieve high performance, e.g. on parallel computers.*
- *The SSF API is minimalist and not tied to any particular domain where simulation models might be developed.*
- *SSF API defines 5 base classes: Entity, InChannel, OutChannel, Process and Event*
- *SSF bridges the gap between models developed in pure Java and models developed in languages specifically designed for simulation.*
- *It also provides the flexibility offered by a general-programming language, yet has essential support for simulation.*

Simulation Software

- *Software package are:*

➢*Arena*
➢*AutoMod*
➢*Delmia/QUEST*
➢*Extend*
➢*Flexsim*
➢*Micro Saint*
➢*ProModel*
➢*Simul8*
➢*WITNES*

Assignments

1. *Explain overall structure of simulation in JAVA for single server queue. [6]*
2. *What do you mean by simulation software? Explain 7 basic blocks of GPSS with syntax, symbol and example [3+7]*
3. *Explain about simulation in JAVA with single server queue model example. [6]*
4. *Define GPSS. [2]*
5. *Explain structure of JAVA simulation. [6]*
6. *GPSS simulation of telephone system. [6]*
7. *GPSS simulation of supermarket. [6]*
8. *GPSS simulation of ambulance. [6]*

. . . to be continued !!!