**Group 11:**

Members: 1) Achal Agarwal

2) Vikram Waradpande

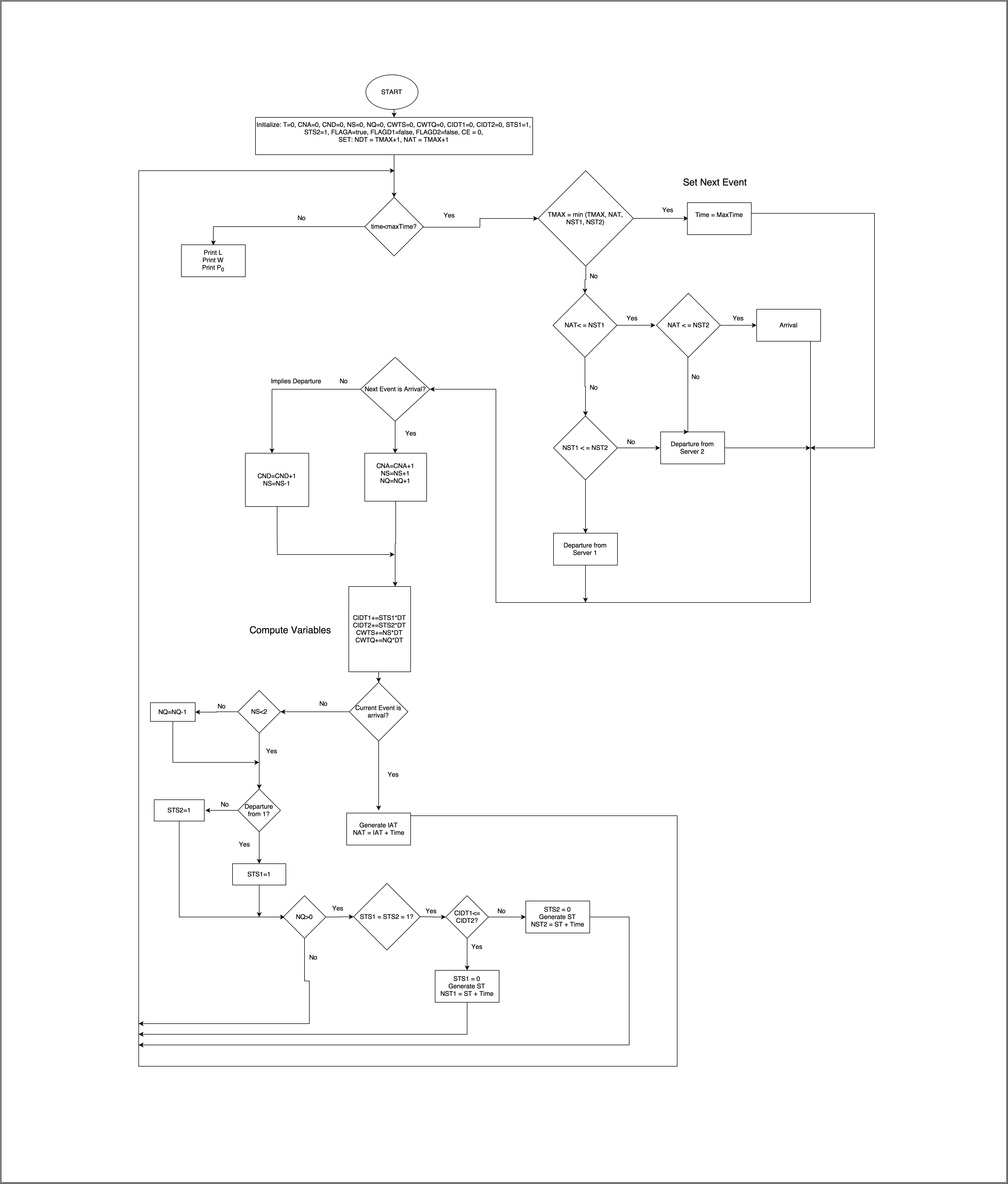
3) Nandit Bansal

**Question**: G11. In the machine repair model, suppose that the repair facility consists of two repairmen, each of whom takes an exponentially distributed random amount of time to repair a failed machine. It is assumed that when both the servers are free then customer goes to server that has been idle longer. Input variables are K = 12, = 30,  = 20, Tmax = 10 units of time.

1. Draw a flow diagram for this system.
2. Write a program to simulate the system for estimating.

Flowchart and Code is attached below.

**­­**



**Flowchart**

**Code**

import java.lang.Math;

import java.io.FileWriter;

import java.util.Random;

public class Queue {

private static int staticCounter = 0;

private int lambda;

private int mu;

private int maxTime; //max time to run the simulation for

//int servers; //number of servers

private double time; //stop watch

private double ptime; //previous time

private int ce; //0 start 1 Arrival 2 Departure 3 Both // change to enum nextEvent (start, arrival, departure, both, end)

private int cna; //cumulative number of arrivals

private int cnd; //cumulative number of departures

private int ns; //number of people in system

private int nq; //number of people in queue

private int k; //max Pop

private double cwts; //cumulative waiting time in the system

private double cwtq; //cumulative waiting time in the queue

private int sts1;

private int sts2;

private double cidt; //state of server at given time (1 = idle, 0 = busy)

private double cidt1;

private double cidt2; //cumulative idle time of the server

private double randA; //random number for Arrival

private double iat; //inter arrival time

private double nat; //next arrival time

private double randB; //random number for Departure

private double st;

private double nst1;

private double nst2; //next Service time

private boolean flagA;

private double net;

private int ne;

private int sumNs;

private Queue(int mt, int l, int m, int mp){

cidt = 0.0;

sumNs =0;

maxTime = mt;

nat = mt + 1;

nst1 = mt + 1;

nst2 = mt + 1;

//servers = 2;

time = 0.0;

cna = 0;

cnd = 0;

k = mp;

ns = 0;

// nq = ns - servers < 0 ? 0 : (ns - servers);

nq = 0;

cwts = 0.0;

cwtq = 0.0;

sts1 = 1;

sts2 = 1;

/\*if (ns >= servers){

sts = 0;

else

sts = 1;\*/

cidt1 = 0.0;

cidt2= 0.0;

lambda = l;

mu = m;

flagA = true;

ce = 0;

staticCounter++;

}

private void setNextTime() {

if (nat <= maxTime && nat <= nst1 && nat<=nst2 ) { //If arrival time is less than both departure times

net = nat;

ne = 1;

// System.out.println("The least time is for NAT - " + nat);

}

else if (nst2 < nst1 && nst2 <= maxTime) { // nst2<nst1 and min(nst1, nst2) < nat

net = nst2;

ne = 3;

// System.out.println("The least time is for NST2 - " + nst2);

}

else if(nst1<=nst2 && nst1<=maxTime){ // nst2<nst1 and min(nst1, nst2) < nat

net = nst1;

ne = 2;

//

//System.out.println("The least time is for NST1 - " + nst1);

}

else {

ne = -1;

net = maxTime;

// System.out.println("MaxTime Overload");

}

}

private double getRandom() {

Random r = new Random();

int a = 0;

while(a==0) {

a = r.nextInt(1000000);

}

return a\*1.0/1000000;

}

private double getExponentialTime(double r, int l) {

return 1.0 \* Math.log(r) \* (-1.0 / (1.0\*l));

}

private void setArrival() {

flagA = false;

randA = getRandom();

int tempLambda = (k-ns)\*lambda;

iat = getExponentialTime(randA, tempLambda);

nat = iat + time;

// System.out.println("nat" + nat);

}

private void setDeparture(){

randB = getRandom();

// System.out.println("rand departure" + randB);

//int tempMu;

st = getExponentialTime(randB, mu);

if(sts1==1&&sts2==1) {

if (cidt1 >= cidt2) {

sts1=0;

nst1 = st+time;

}

else {

sts2=0;

nst2 = st+time;

}

//server is 2

}

else if(sts1==1) {

sts1=0;

nst1 = st+time;

}

//server is 1

else if(sts2==1){

sts2=0;

nst2 = st+time;

}

// System.out.println("nst1 and nst2 " + nst1 + "," + nst2);

}

private void calculateWaitingTime() { //before updating variables as old sts required

cwts += ns \* (time - ptime);

cwtq += nq \* (time - ptime);

cidt1 += sts1 \* (time - ptime);

cidt2 += sts2\*(time - ptime);

cidt +=sts1\*sts2\*(time - ptime);

}

private void updateVariables() {

if (ce == 1) { //If event is an arrival

//nat=maxTime+1;

cna++;

ns++;

if (sts1 ==1 || sts2 ==1)

setDeparture();

else

nq++;

if(ns<k)

flagA = true;

nat = maxTime+1;

//flagD = true;

}

if (ce == 2) {

//If departure occurs from server 1

nst1 = maxTime+1;

ns--;

if(ns==k-1)

flagA=true;

sts1 = 1;

if (nq > 0) { //sts1 || sts2 == 1 is implied

setDeparture();

nq--;

//flagD1 = true;

}

cnd++;

}

if (ce == 3) {

nst2 = maxTime+1;

ns--;

if(ns==k-1)

flagA=true;

sts2 = 1;

if (nq > 0) { //sts1 || sts2 == 1 is implied

setDeparture();

nq--;

//flagD2 = true;

}

cnd++;

}

}

public static void main(String args[]){

//staticCounter++;

//System.out.println(staticCounter + "sooo");

// com.programming.operations o = new com.programming.operations(5, 1, 0, false, 2, 3);

char ce=' ';

char ne=' ';

// int a,b,c,d;

/\*if(args[1] == null)

a = 10;

else

a = Integer.parseInt(args[1]);

if(!args[2] == null)

b = 30;

else

b = Integer.parseInt(args[2]);

if(!args[3] == null)

c = 20;

else

c = Integer.parseInt(args[3]);

if(!args[4] == null)

d = 12;

else

d = Integer.parseInt(args[4]);

\*/

Queue q = new Queue(10,30,20,12);

boolean flag = true;

int d =0;

int c = 0;

String file;

if(args.length==0){

file="simulation";}

else

file=args[0];

try {

FileWriter fw = new FileWriter("./"+file+".txt");

String head = String.format("%-4s %-11s %-4s %-5s %-5s %-4s %-4s %-12s %-12s %-4s %-4s %-11s %-11s %-10s %-11s %-11s %-10s %-11s %-11s %-11s %-11s %-2s\n", "c", "time", "ce", "cna", "cnd", "ns", "nq", "cwts","cwtq", "sts1", "sts2", "cidt1", "cidt2", "randA", "iat", "nat","randB", "st", "nst1","nst2", "net", "ne");

//fw.write("index" +"\t" + "time" + "\t" + "ce" + "\t" + "cna" + "\t" + "cnd" + "\t" + "ns" + "\t" + "nq" + "\t" + "cwts" + "\t" + "cwtq" + "\t" + "sts1" + "\t" +"sts2" + "\t" + "cidt1" + "\t" + "cidt2" + "\t" + "randA" + "\t" + "iat" + "\t" + "nat" + "\t" +"randB" + "\t" + "st" + "\t" + "nst1" + "\t" + "nst2"+ "\t" + "net" + "\t" + "ne" +"\n");

fw.write(head + "\n");

while (q.time <= q.maxTime) {

c++;

if(q.flagA)

q.setArrival();

/\*if(q.nq>0 && (q.sts1 == 1 || q.sts2 ==1)) {

d++;

q.setDeparture();

}

\*/

// System.out.println("Set Arrival, Set Departure Flags " + q.flagA + " , departure 1 - " + q.flagD1 +" , departure 2 = " + q.flagD2);

/\* if (q.flagD2 == true) {

q.setDeparture(); //Schedule departure from server 2

}

\*/

// Decide next event

q.setNextTime();

// FileWriter fw = new FileWriter("/answer.txt");

//if(((q.nst1 == q.nst2)||(q.nst2==q.nat)||(q.nat==q.nst1))&&c!=1)

// System.out.println(c);

if(q.ne == 1)

ne = 'A';

if(q.ne == 2)

ne = 'D';

if(q.ne == 3)

ne = 'D';

if(q.ne == -1)

ne = 'E';//end

if(q.ce == 0)

ce = 'S';

if(q.ce == 1)

ce = 'A';

if(q.ce == 2)

ce = 'D';

if(q.ce == 3)

ce = 'D';

if(!flag){

ne = '-';

ce = 'E';

}

String temp = String.format("%-4s %-11s %-4s %-5s %-5s %-4s %-4s %-12s %-12s %-4s %-4s %-11s %-11s %-10s %-11s %-11s %-10s %-11s %-11s %-11s %-11s %-2s", c, String.format("%.6f",q.time), ce, q.cna, q.cnd, q.ns, q.nq, String.format("%.6f",q.cwts),String.format("%.6f",q.cwtq), q.sts1, q.sts2, String.format("%.6f",q.cidt1), String.format("%.6f",q.cidt2), String.format("%.6f",q.randA), String.format("%.6f",q.iat), String.format("%.6f",q.nat), String.format("%.6f",q.randB), String.format("%.6f",q.st), String.format("%.6f",q.nst1), String.format("%.6f",q.nst2), String.format("%.6f",q.net), ne);

//fw.write(c +"\t" + String.format("%.6f",q.time) + "\t" + q.ce + "\t" + q.cna + "\t" + q.cnd + "\t" + q.ns + "\t" + q.nq + "\t" + String.format("%.6f",q.cwts) + "\t" + String.format("%.6f",q.cwtq) + "\t" + q.sts1 + "\t" + q.sts2 + "\t" + String.format("%.6f",q.cidt1)+ "\t" + String.format("%.6f",q.cidt2) + "\t" + String.format("%.6f",q.randA) + "\t" + String.format("%.6f",q.iat) + "\t" + String.format("%.6f",q.nat) + "\t" + String.format("%.6f",q.randB) + "\t" + String.format("%.6f",q.st) + "\t" + String.format("%.6f",q.nst1) + "\t" + String.format("%.6f",q.nst2)+ "\t" + String.format("%.6f",q.net) + "\t" + q.ne +"\n");

fw.write(temp + "\n\n");

q.ptime = q.time;

//Store previous time

q.time = q.net;

if(!flag)

break;

if ( q.ne == -1||q.time==q.maxTime) {

q.time = q.maxTime;

// q.ne = null;

// q.net = null;

flag = false;

}//Set time = next event time

// System.out.println("The time is " + q.time);

q.ce = q.ne; //Set next event

// System.out.println("The event is (1 Arrival 2 Departure)" + q.ce);

q.calculateWaitingTime();

q.updateVariables();

// if(q.nat>o.maxTime&&o.nst>o.maxTime)

// o.time=o.maxTime+1;

}

fw.close();

}catch(Exception e){

System.out.println(e);

}

double l = q.cwts/q.maxTime;

double w = q.cwts/q.cna;

double p = q.cidt/q.maxTime;

System.out.println("Estimated number of People in System - "+ String.format("%.6f",l));

System.out.println("Estimated waiting time of People in System - "+ String.format("%.6f",w));

System.out.println("Estimated P - "+ String.format("%.6f",p));

System.out.println("To see the simulation, look for a file named 'simulation.txt' in the current folder");

}

}