Rachel McGuigan
Assignment Number 5
Discrete Event Simulation of a Queue

In this simulation assignment, a single server M/M/1 queueing system will be simulated. The inter-arrival and service times are exponentially distributed. The essential code is given in the handout on simulation. This code is given in C language.

1. Use any language to code the algorithm. That is you can use either C or C++ or Java or Pascal or FORTRAN.
2. Use the algorithm given in Jain for generating uniformly distributed random numbers between 0 and 1. It is given on pages 443 and 444. Try to use integer arithmetic if possible (that is page 443 algorithm), else try the algorithm given on page 444 (real arithmetic)
3. **Use two different streams for arrival processes and service times.** That is use two different seed for these streams. Use the seeds given in Table 26.2 on page 455. **You need to track two random streams.**
4. Given Ta = 200 time units, Ts = 100 time units.
   - Compute the theoretical values of X, U, L, and W. Use the formulae for the M/M/1 queue.
   - For te = 100, 1000, 10000, 100000, and 1000000 time units, simulate the queue and note the values of X, U, L, and W. Present results in a tabular form. Note that te is the simulation period.)

te	100	1000	10000	100000	1000000	
throughput	4.96530348136387 8E-4	4.96530348136387 8E-4	2.93085913698665 2E-4	2.6E-4	2.43740294421642 93E-4	
utilization =	1	1.0	0.60739087928349 98	0.54219135457748 22	0.50005367486768 38	
mean no. in system =	0.99951559079854 87	0.99951559079854 87	0.60717631787906 8	0.51584	0.49992433002201 7	
mean residence time =	2013.00000000000 02	2013.00000000000 02	2071.66666666666 65	1984.0	2051.05327868852 46	
This is p:	0.5 1.0	0.5	0.60739087928349 98	0.54219135457748 22	2051.05327868852 46	
This is Wq:	100.0	100.0	100.0	100.0	100.0	

5. Given Ta = 200 time units, te = 1000000 time units. Vary Ts.

   For Ts = 5, 10, 20,40,60,80,100,120,140,160,180 time units:

   - Using M/M/1 formulae compute X, U, L, and W. These results are independent of te!
   - Run simulation program and observe X, U, L, and W.

Ts	5	10	20	40	60	80
throughput	2.45E-4	2.45E-004	2.45E-4	2.45E-4	2.43932239590466 97E-4	2.43836229231785 07E-4
utilization =	0.02593311659938 714	0.05096244073252 426	0.10102108899880 015	0.20113838553135 144	0.30026847989665 66	0.40008228774251 8
mean no. in system =	0.024916	0.04994	0.1	0.200116	0.30015362220460 257	0.40008228774251 8
mean residence time =	101.697959183673 48	203.836734693877 56	408.163265306122 47	816.8	1230.47950819672 13	1640.78278688524 6
This is p:	0.02593311659938 714	0.05096244073252 426	0.10102108899880 015	0.20113838553135 144	0.30026847989665 66	0.40020039454717 665
This is Wq:	0.12820512820512 822	0.52631578947368 42	2.22222222222222 2	9.99999999999999 8	25.71428571428571 5	53.33333333333333 3
L	0.0266235481785919665855496305720708172608915950328811	0.0536990767487222913053525535743367347296047709625107	0.1123731466473358895957406881070523249853797218474086	0.2517812620964843531504319180286945396249997628811852	0.42911955695851	0.6672235041652657

Ts	100	120	140	160	180	80
throughput	2.43740294421642 93E-4	2.43644435070904 3E-4	2.43548651090573 12E-4	2.43452942391793 06E-4	2.43357308885847 6E-4	
utilization =	0.50005367486768	0.59982841363523	0.69952470348100	0.79914263689053	0.89868230620406	

	38	63	26	38	56	
mean no. in system =	0.4999243300220177	0.5997138016999973	0.6994008572672594	640.00000000000001	0.9009855546789497	
mean residence time =	2051.0532786885246	2461.4303278688526	2871.709016393442	3282.9549180327867	3702.315573770492	
This is p:	0.5000536748676838	0.5998284136352363	0.6995247034810026	0.7991426368905338	0.8986823062040656	
This is Wq:	100.0	179.99999999999997	326.66666666666663	640.0000000000001	1620.0000000000005	
L	1.0002147225211410219526030125556278251872079218843430	1.498928045052358515661666728383715119613367684 7908072	2.32806061458292129	3.978657413992	8.86994435556	

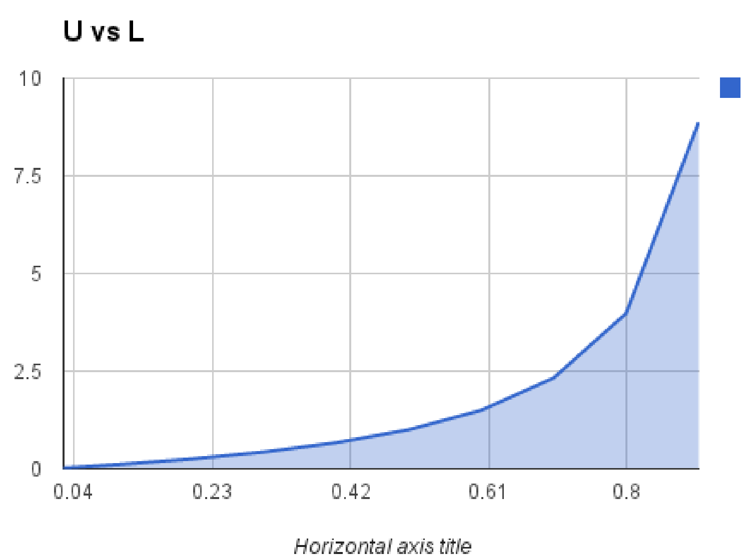
6. Present your results in a table.
    - Plot U versus L, simulation and theoretical results on the same graph. Note that U is the independent variable, therefore it should be on the x-axis. There should be two curves in this graph.

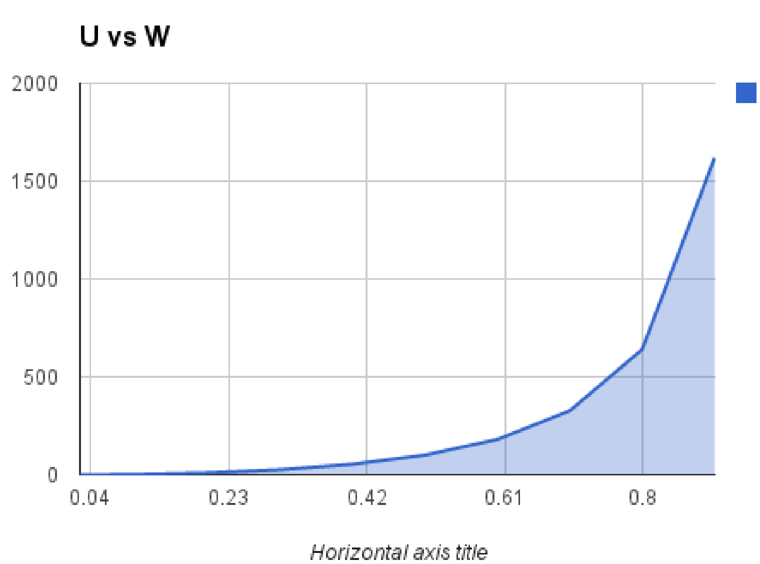
U	L	U	W
0.025933116599387	0.026623548178592	0.025933116599387	0.128205128205128
0.050962440732524	0.053699076748722	0.050962440732524	0.526315789473684
0.1010210889988	0.112373146647336	0.1010210889988	2.22222222222222
0.201138385531351	0.251781262096484	0.201138385531351	10
0.300268479896657	0.42911955695851	0.300268479896657	25.7142857142857
0.400082287742518	0.667223504165266	0.400082287742518	53.3333333333333
0.500053674867684	1.00021472252114	0.500053674867684	100
0.599828413635236	1.49892804505236	0.599828413635236	180
0.699524703481003	2.32806061458292	0.699524703481003	326.666666666667

0.799142636890534	3.978657413992	0.799142636890534	640

- Plot U versus W, simulation and theoretical results on the same graph. Note that U is the independent variable, therefore it should be on the x-axis. There should be two curves in this graph.

Note: both of these graphs should have a label in the horizontal axis of “Utilization”





Code:

```
package rng;
import java.io.*;
import java.util.*;
import java.lang.Math;

public class assign5a {


	//Rachel McGuigan
	//Random number generator to test a simulated server FIFO queue
	//variables Assignment 5
	//
public static double T = 0;// period of time over which the system is measured.
public static double A = 0;// number of arrivals in time T.
public static double C = 0; //number of completions in time
//public static double T = 0;//arrival rate.
public static double X = 0; //throughput rate.
public static double B = 0;//server busy time.
public static double U = 0;//server utilization.
public static double Ts = 0;// mean service time per customer.
public static double L = 0; //average number of customers in the system.
public static double W = 0; //average time of the customers spent in the system.
public static double Lq = 0; //average number of customers queued in the system.
public static double Wq = 0; //mean queueing time.

/*Arrival rate:   = A=T

  Throughput rate: X = C=T

//ow balance: U =  T s
//Lq =  W q*/

    public void doSimulation(){

	//initialize the vars  */
	double Ta = 200.0, Ts =160, tn, tb = 0, time;
	double t1;
	double t2;
	int te = 1000000;
	double B=0, C=0.0, L, U, W, X;
	int n = 0;
	int s=0;
	 int ia=1;
	 int is=1;
```

```
        t1 = 0; t2 = te; time = 0.0; tn = time;

  //call first event t1= getRandom, t2=getRandom
        //727633698
        double nexta= getRandom(1);
        //t1=getdist(nexta, Ta);
        double nexts= getRandom(276090261);
        //t2=getdist(nexts, Ts);
        System.out.print(t1); System.out.print(t2);

  // do this by calculate the first two seeds t1 and t2 first i

  while (time < te){
          printC(C, t1, t2, tn, tb);
        if (t1 < t2){
                // the event has arrived
                time=t1; s+= n*(time - tn);
                n++; tn=time;

                System.out.println("current s in queue:  ");
                System.out.println(s);
                nexta= getRandom(nexta);
                t1= time + getdist(nexta, Ta);
                ia++;
                //if queue is empty time=total time->
                if(n==1){
                        tb=time;
                        nexts=getRandom(nexts);
                        t2= time + getdist(nexts,Ts); is++;}
                }
                // the next event is completed
        else{
                 time=t2;
                 s += n*(time - tn);
                 n--;
                 tn = time;
                 C++;
                //When the simulation completes, the average number in the system is
computed by dividing s by the observation period length, and the average residence time
then computed using Littleís Law.
                //print out each C here

                 printC(C, t1, t2, tn, tb);
                 if (n > 0) {
                        nexts=getRandom(nexts);
                        t2 = time + getdist(nexts,Ts);is++;
```

```
                }
            else{
                t2 = te;
                //B += time - tb;
                B = B + time- tb;
                }
            }
            }
    //end while
    X = C / time;System.out.println("throughput ="); System.out.println(X);
    U = B / time; System.out.println("utilization
=");System.out.println(B);System.out.println(time); System.out.println(U);
    L = s / time; System.out.println("mean no. in system ="); System.out.println(L);
    W = L / X; System.out.println("mean residence time ="); System.out.println(W);

    //print out: i, Ci, Interarrival time, Service time, #Ai, #Si, wi


    //print out m/m/i Queue
    System.out.println("This is p:");
    double z= calcU(Ts,Ta );
    System.out.println(z);
    System.out.println(B/time);
    System.out.println("This is Wq:");
    double o= calcWq(Ts, z);
    System.out.println(o);
    System.out.println("Wq really:");System.out.println(z/((1/Ts)*(1-z)));

     }
   //function to generate t1 s t2 a since they both have same behavior

        //function to calculate utilization
        private static double calcU(double s, double t){
            double calc;
             calc=s/t;
             return calc;
        }

        //function for Wq or the work in the queue
        public static double calcWq(double Ts, double util){
            double mu = 1/Ts;
            return (util)/(mu*(1-util));


        }
```

```
}

    private static double getRandom(double x) {

        //first calculate the rng
        double n = (16807*(x)) % 2147483647;
        System.out.print("This is n:");
        System.out.print(n);
            //then make sure it is exponentially distributed
        return n;
    }

    private double getdist(double randprev, double expected){
        // expected values dist: 1/m sum (an) and 1/m sum(sn)
       //expected value will be equal to -Taln(x) via
       //1-e^(-lambda*t)
            double u;
       double operator1;
       double operator2;
       System.out.println("this is randprev minus 1:");
       System.out.println(1-randprev);
       randprev=Math.abs(1-randprev);
       operator1= Math.log(randprev);
       System.out.println("This is the operator: ");
       operator2= operator1*expected;
       System.out.print(operator1);
       //operator2= Math.abs(operator2);
       //x= -a*ln(x);
       System.out.println("This is the operator2: ");
       System.out.println(operator2);
       return operator2;

    }

    private void printC(double v, double time1, double time2, double tn, double tb){
        System.out.println("      ");
       System.out.println("i Ci Interarrival time Service time  Ai Si wi");
       System.out.println(v);
       System.out.println(v);
       System.out.println(time1);
       System.out.println(time2);
       System.out.println(tb);
       System.out.println(tn);
       System.out.println(tb-tn);

    }

}
```