Assignment #3

Analytical results: for M/D/1, n=20000 customers

Mean interarrival time (minutes)	1.000	1.000	1.000	1.000	1.000
Mean service time minutes	0.300	0.400	0.500	0.600	0.700
Mean time an item spends in the system	0.3*(2-0.3) / 2*(1-0.3) = 0.364	0.4*(2-0.4) /2*(1-0.4) = 0.533	0.5*(2-0.5) /2(1-0.5) = 0.75	0.6*(2-0.6) / 2(1-0.6) = 1.05	0.7*(2-0.7) /2(1-0.7) = 1.517
Mean no of items waiting to be served	0.3*0.3/2(1-0.3) =0.064	0.4*0.4/2(1-0.4) = 0.133	0.5*0.5/2(1-0.5) = 0.25	0.45	1.817
Mean waiting time (includes items that have to wait and items with waiting time=0)	0.3*0.3/2(1-0.3) =0.064	0.133	0.5*0.5/2(1-0.5) = 0.25	0.45	1.817
Average delay in queue	0.064	0.133	0.250	0.450	1.817
Average number in queue	0.064	0.133	0.250	0.450	1.817

Simulation results:

For M/D/1

Mean interarrival time (minutes)	1.000	1.000	1.000	1.000	1.000
Mean service time minutes	0.300	0.400	0.500	0.600	0.700
Mean time an item spends in the system	0.3*(2-0.3) / 2*(1-0.3) = 0.364	0.4*(2-0.4) /2*(1-0.4) = 0.533	0.5*(2-0.5) /2(1-0.5) = 0.75	0.6*(2-0.6) / 2(1-0.6) = 1.05	0.7*(2-0.7) /2(1-0.7) = 1.517
Mean no of items waiting to be served	0.3*0.3/2(1-0.3) =0.064	0.4*0.4/2(1-0.4) = 0.133	0.5*0.5/2(1-0.5) = 0.25	0.45	0.817
Mean waiting time (includes items that have to wait and items with waiting time=0)	0.3*0.3/2(1-0.3) =0.064	0.133	0.5*0.5/2(1-0.5) = 0.25	0.45	0.817
Average delay in queue	0.064	0.132	0.245	0.440	0.806
Average number in queue	0.065	0.133	0.247	0.444	0.812

Program:

```
/* External definitions for single-server queueing system. */
#include <stdio.h>
#include <math.h>
#include<stdlib.h>
#include "lcgrand.h"
/* #include "lcgrand.h" /* Header file for random-number generator. */
#define Q_LIMIT 100 /* Limit on queue length. */
#define BUSY 1 /* Mnemonics for server's being busy */
#define IDLE 0 /* and idle. */
int next_event_type, num_custs_delayed, num_delays_required, num_events,
 num_in_q, server_status;
```

```
float area_num_in_q, area_server_status, mean_interarrival, mean_service,
 sim_time, time_arrival[Q_LIMIT + 1], time_last_event,
time_next_event[3],
 total_of_delays;
FILE *infile, *outfile;
void initialize(void);
void timing(void);
void arrive(void);
void depart(void);
void report(void);
void update_time_avg_stats(void);
float expon(float mean);
int main() /* Main function. */
{
 /* Open input and output files. */

 infile = fopen("md1.in", "r");
 outfile = fopen("md1.out", "w");
 /* Specify the number of events for the timing function. */
 num_events = 2;
 /* Read input parameters. */
 fscanf(infile, "%f %f %d", &mean_interarrival, &mean_service,
 &num_delays_required);
 /* Write report heading and input parameters. */
 fprintf(outfile, "Single-server queueing system\n\n");
 fprintf(outfile, "Mean interarrival time%11.3f minutes\n\n",
 mean_interarrival);
 fprintf(outfile, "Mean service time%16.3f minutes\n\n", mean_service);
 fprintf(outfile, "Number of customers%14d\n\n", num_delays_required);
 /* Initialize the simulation. */
 initialize();
 /* Run the simulation while more delays are still needed. */
 while (num_custs_delayed < num_delays_required) {
 /* Determine the next event. */
 timing();
 /* Update time-average statistical accumulators. */
 update_time_avg_stats();
```

```
/* Invoke the appropriate event function. */
switch (next_event_type) {
case 1:
arrive();
break;
case 2:
depart();
break;
}
}
/* Invoke the report generator and end the simulation. */
report();
fclose(infile);

fclose(outfile);
return 0;
}
void initialize(void) /* Initialization function. */
{
/* Initialize the simulation clock. */
sim_time = 0.0;
/* Initialize the state variables. */
server_status = IDLE;
num_in_q = 0;
time_last_event = 0.0;
/* Initialize the statistical counters. */
num_custs_delayed = 0;
total_of_delays = 0.0;
area_num_in_q = 0.0;
area_server_status = 0.0;
/* Initialize event list. Since no customers are present, the
departure
(service completion) event is eliminated from consideration. */
time_next_event[1] = sim_time + expon(mean_interarrival);
time_next_event[2] = 1.0e+30;
}
void timing(void) /* Timing function. */
```

```
{
 int i;
 float min_time_next_event = 1.0e+29;
 next_event_type = 0;
 /* Determine the event type of the next event to occur. */
 for (i = 1; i <= num_events; ++i){
 if (time_next_event[i] < min_time_next_event) {
 min_time_next_event = time_next_event[i];
 next_event_type = i;
 }
 }
 /* Check to see whether the event list is empty. */
 if (next_event_type == 0) {
 /* The event list is empty, so stop the simulation. */

 fprintf(outfile, "\nEvent list empty at time %f", sim_time);
 exit(1);
 }
 /* The event list is not empty, so advance the simulation clock. */
 sim_time = min_time_next_event;
}
void arrive(void) /* Arrival event function. */
{
 float delay;
 /* Schedule next arrival. */
 time_next_event[1] = sim_time + expon( mean_interarrival);
 /* Check to see whether server is busy. */
 if (server_status == BUSY) {
 /* Server is busy, so increment number of customers in queue. */
 ++num_in_q;
 /* Check to see whether an overflow condition exists. */
 if (num_in_q > Q_LIMIT) {
 /* The queue has overflowed, so stop the simulation. */
 fprintf(outfile, "\nOverflow of the array time_arrival at");
 fprintf(outfile, " time %f", sim_time);
 exit(2);
 }
```

```
 /* There is still room in the queue, so store the time of arrival
of the
 arriving customer at the (new) end of time_arrival. */
 time_arrival[num_in_q] = sim_time;
 }
 else {
 /* Server is idle, so arriving customer has a delay of zero. (The
 following two statements are for program clarity and do not
affect
 the results of the simulation.) */
 delay = 0.0;
 total_of_delays += delay;

 /* Increment the number of customers delayed, and make server
busy. */
 ++num_custs_delayed;
 server_status = BUSY;
 /* Schedule a departure (service completion). */
 time_next_event[2] = sim_time + mean_service; /*changes here*/
 }
}
void depart(void) /* Departure event function. */
{
 int i;
 float delay;
 /* Check to see whether the queue is empty. */
 if (num_in_q == 0) {
 /* The queue is empty so make the server idle and eliminate the
 departure (service completion) event from consideration. */
 server_status = IDLE;
 time_next_event[2] = 1.0e+30;
 }
 else {
 /* The queue is nonempty, so decrement the number of customers in
 queue. */
 --num_in_q;
 /* Compute the delay of the customer who is beginning service and
```

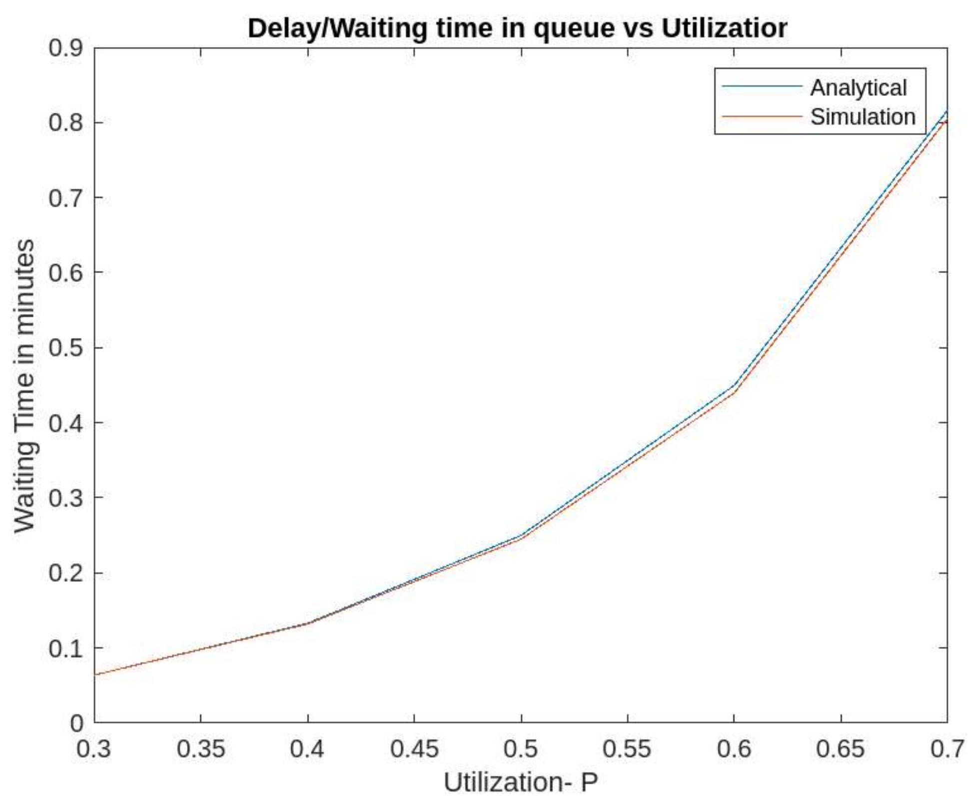
```
update
 the total delay accumulator. */
 delay = sim_time - time_arrival[1];
 total_of_delays += delay;
 /* Increment the number of customers delayed, and schedule
departure. */
 ++num_custs_delayed;
 time_next_event[2] = sim_time +  mean_service; /* changes here */
 /* Move each customer in queue (if any) up one place. */
 for (i = 1; i <= num_in_q; ++i)
 time_arrival[i] = time_arrival[i + 1];
 }
}
void report(void) /* Report generator function. */
{
 /* Compute and write estimates of desired measures of performance. */
 fprintf(outfile, "\n\nAverage delay in queue%11.3f minutes\n\n",
 total_of_delays / num_custs_delayed);
 fprintf(outfile, "Average number in queue%10.3f\n\n",
 area_num_in_q / sim_time);
 fprintf(outfile, "Server utilization%15.3f\n\n",
 area_server_status / sim_time);
 fprintf(outfile, "Time simulation ended%12.3f minutes", sim_time);
}
void update_time_avg_stats(void) /* Update area accumulators for timeaverage
 statistics. */
{
 float time_since_last_event;
 /* Compute time since last event, and update last-event-time marker.
*/
 time_since_last_event = sim_time - time_last_event;
 time_last_event = sim_time;
 /* Update area under number-in-queue function. */
 area_num_in_q += num_in_q * time_since_last_event;
 /* Update area under server-busy indicator function. */
 area_server_status += server_status * time_since_last_event;
}
```

```
float expon(float mean) /* Exponential variate generation function. */

{
 /* Return an exponential random variate with mean "mean". */
 return -mean * logf(lcgrand(1));
}



/*
float expon(float mean) /* Exponential variate generation function.*/
/*
{
float u;
/* Generate a U(0,1) random variate. */
/*
u = lcgrand(1);
/* Return an exponential random variate with mean "mean". */
/*
return -mean * log(u);
} */
```

Plot using MATLAB:

```
x = [0.3,0.4,0.5,0.6,0.7];
y1 = [0.064,0.133,0.25,0.45,0.817];
y2 = [0.064,0.132,0.245,0.44,0.806];
plot(x,y1,x,y2)
legend({'Analytical','Simulation'},'Location','northeast')
title('Delay/Waiting time in queue vs Utilization')
xlabel('Utilization- P')
ylabel('Waiting Time in minutes')
```

Figure:



The above figure shows the results for the measures both analytic and from simulation as a function of ρ(utilization) $= \lambda/\mu$ where λ is arrival rate(= inverse of mean interarrival time) and μ is service rate (inverse of mean service time)

***