CSE 6010 Assignment 2

Prob. 1

In this part, we generated two sets of sample print-outs to test the correctness of our codes. To simplify the testing, we used **CONSTANT** inter-arrival time (A) and **CONSTANT** service time (S) instead of random exponential distributed values so that we can visually inspect the results easily. We sent 5 parts to go through the complete processing system.

1. In the first case, A = 20 time units, S = 10 time units. The arrival rate is smaller than the processing speed, so there is no queuing delay in this scenario. The results meet our expectation.
2. In the second case, A = 5 time units, S = 10 time units. The arrival rate is larger than the processing speed, so there is supposed to be queuing delay in this scenario. As we can see from the result, for example, in terms of Part #2, though it was generated and sent to Station #1 at T=10, it waited 5 time units until T = 15 when Part #1 was released from Station #1 and the Station was free that it started being processing by the station. Waiting also happened to Parts #3, 4, and 5. By visual inspection of the output results, we can see our print-outs meet expectation. Hence, it more or less proved the correctness of our simulation.

Sample Print Out_1

```
Constant Interarrival Time(A):   20.000000
Constant Service Time(S):        10.000000
Total Part Number is:            5

Part #1:     Arrival Time   -->    Finish Time
Source Time: 20.000000
Station #1   20.000000    -->    30.000000
Station #2   30.000000    -->    40.000000
Station #3   40.000000    -->    50.000000
Sink Time:   50.000000

Part #2:     Arrival Time   -->    Finish Time
Source Time: 40.000000
Station #1   40.000000    -->    50.000000
Station #2   50.000000    -->    60.000000
Station #3   60.000000    -->    70.000000
Sink Time:   70.000000

Part #3:     Arrival Time   -->    Finish Time
Source Time: 60.000000
Station #1   60.000000    -->    70.000000
Station #2   70.000000    -->    80.000000
Station #3   80.000000    -->    90.000000
Sink Time:   90.000000

Part #4:     Arrival Time   -->    Finish Time
Source Time: 80.000000
Station #1   80.000000    -->    90.000000
```

```
Station #2   90.000000    -->    100.000000
Station #3   100.000000   -->    110.000000
Sink Time:   110.000000

Part #5:     Arrival Time   -->    Finish Time
Source Time: 100.000000
Station #1   100.000000   -->    110.000000
Station #2   110.000000   -->    120.000000
Station #3   120.000000   -->    130.000000
Sink Time:   130.000000
```

Sample Print Out_2

```
Constant Interarrival Time(A):   5.000000
Constant Service Time(S): 10.000000
Total Part Number is:            5

Part #1:    Process Start Time  -->  Finish Time
Source Time: 5.000000
Station #1   5.000000     -->    15.000000
Station #2   15.000000    -->    25.000000
Station #3   25.000000    -->    35.000000
Sink Time:   35.000000

Part #2:    Process Start Time  -->  Finish Time
Source Time: 10.000000
Station #1   15.000000    -->    25.000000
Station #2   25.000000    -->    35.000000
Station #3   35.000000    -->    45.000000
Sink Time:   45.000000

Part #3:    Process Start Time  -->  Finish Time
Source Time: 15.000000
Station #1   25.000000    -->    35.000000
Station #2   35.000000    -->    45.000000
Station #3   45.000000    -->    55.000000
Sink Time:   55.000000

Part #4:    Process Start Time  -->  Finish Time
Source Time: 20.000000
Station #1   35.000000    -->    45.000000
Station #2   45.000000    -->    55.000000
Station #3   55.000000    -->    65.000000
Sink Time:   65.000000

Part #5:    Process Start Time  -->  Finish Time
Source Time: 25.000000
Station #1   45.000000    -->    55.000000
Station #2   55.000000    -->    65.000000
Station #3   65.000000    -->    75.000000
Sink Time:   75.000000
```

Prob. 2

In this part, simulation software has been developed to perform a series of runs increasing the arrival rate (1 / A). Based on the base case model, the arrival rate has been varied to analyze the behavior of this system for different workloads. The codes are attached in the *"Problem_2_codes"* folder. As we can see from the graph, as the arrival rate increases, the time of a part in go through the system and the queuing delays both increase. Queuing delay became a bottle neck of the effectiveness of the parts' processing process. However, when the arrival rate increases to a certain level, the continuing creases won't cause significant increase in queuing delay. The average time in system and the queuing delay both level off.

Notes: When running the model for multiple times, the data could fluctuate within a certain range, due to the random interval generator, but the general trend keeps the pattern.

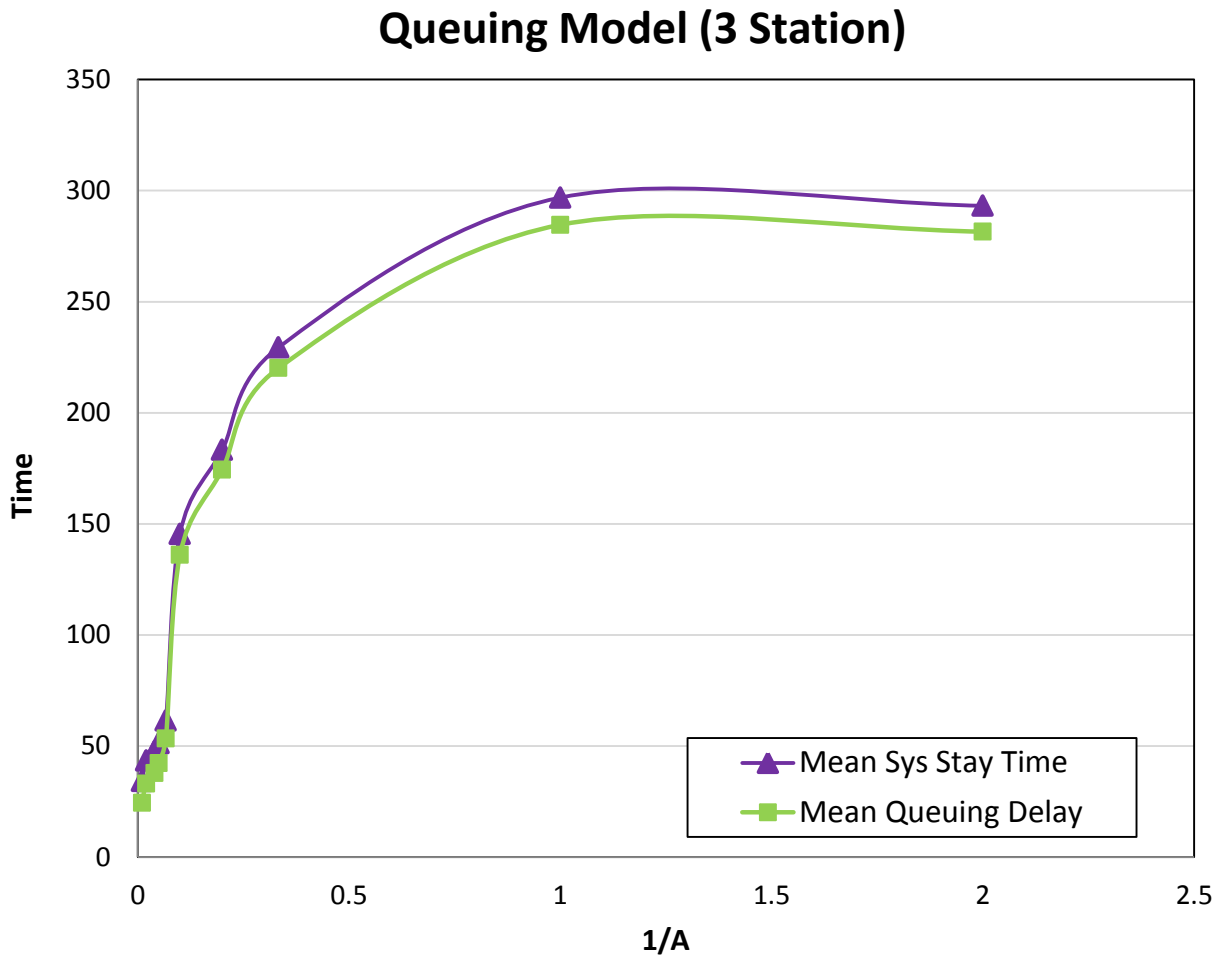
Sample Output (Simulation Time ~ 10,000 time units)

Part#	Time_in_the_System	Queuing Delay	Total Serv.Time
1	30.936046	20.624031	10.312015
2	37.873102	30.312813	7.560289
3	33.689724	28.808671	4.881053
4	34.585159	32.641528	1.943631
5	28.938712	26.262409	2.676303
6	33.006807	24.863035	8.143771
7	33.580009	29.224387	4.355622
8	94.267002	66.086910	28.180092
9	88.588521	84.984688	3.603833
10	95.209731	83.500096	11.709635
11	91.639651	90.092090	1.547561
12	88.055303	82.581761	5.473542
13	88.968662	88.011470	0.957192
14	86.428437	84.584753	1.843684
15	88.835087	85.781641	3.053446
16	90.652518	86.517587	4.134931
17	91.105553	83.313146	7.792406
18	98.726281	85.057072	13.669209
19	85.314786	84.392795	0.921991
20	87.785417	81.097841	6.687576
21	93.334041	85.508984	7.825057
22	139.737179	104.224206	35.512973
23	142.866883	137.242914	5.623968
24	161.065108	142.751265	18.313843
25	176.957854	160.100515	16.857339
26	177.525904	176.397999	1.127905
27	187.114861	174.064286	13.050574
28	188.744664	185.670903	3.073761
29	192.858297	187.603398	5.254900
30	201.262806	190.625460	10.637346
31	217.075457	196.855372	20.220085
32	232.630712	211.651799	20.978913

33	244.861699	226.655863	18.205837
34	247.205119	239.797702	7.407417
35	242.460744	232.909047	9.551697
36	237.540774	237.426905	0.113870
37	244.755849	226.963508	17.792341
38	259.221539	242.697835	16.523705
39	266.029061	254.238391	11.790671
40	276.545212	259.596880	16.948331
41	280.484111	268.062729	12.421381
42	276.919087	274.610063	2.309024
43	280.525008	274.308642	6.216367
44	261.944991	255.392312	6.552678
45	264.441232	259.344566	5.096666
46	259.690333	259.211507	0.478827
47	255.162173	252.283622	2.878550
48	252.775969	251.953544	0.822425
49	249.897587	249.449129	0.448458
50	269.269686	238.275597	30.994089
51	271.702547	268.438576	3.263971
52	265.381668	264.923310	0.458358
53	268.175091	264.967638	3.207452
54	269.476475	268.024998	1.451477
55	271.130297	268.389173	2.741124
56	269.255954	266.128096	3.127858
57	263.101520	262.263842	0.837678

A = 5.000000; S = 10.000000.
Average time in the system: 176.619544
Average queuing delay: 168.381567
Total simulation time: 10067.313998

(Notes: Simulation ended when time exceeded 10000 units.)


Queuing Model (3 Station)
350
300
250
200
150
100
50
0
Time
0
0.5
1
1.5
2
2.5
1/A
Mean Sys Stay Time
Mean Queuing Delay

Prob. 3.2 One Station Queuing Simulation

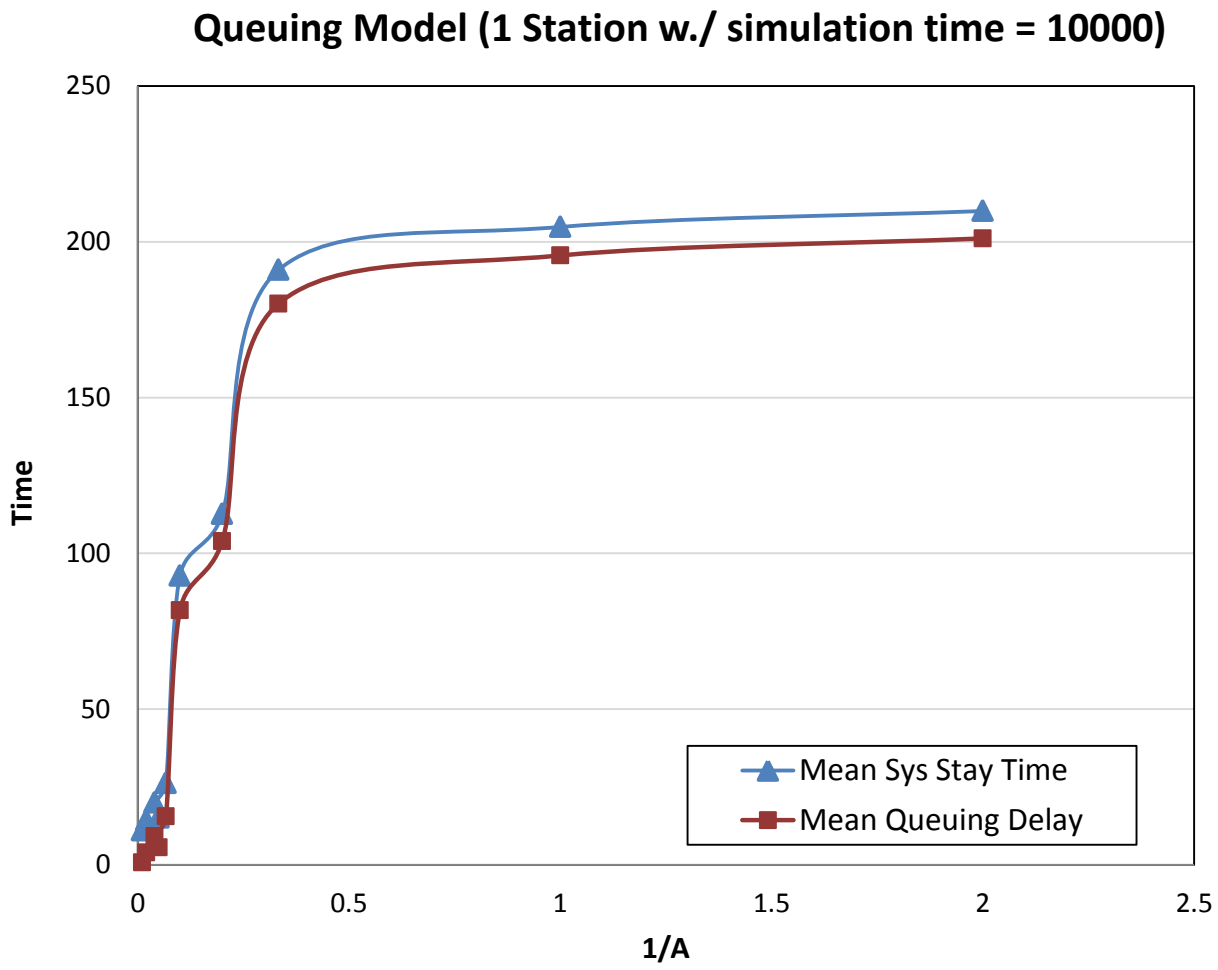
To do this problem with only one station, the main change would be in the *part.h* file. We changed the lines 3-4 from:

```
#define STATE_NUM 5
typedef enum {SOURCE = 0, STATE1, STATE2, STATE3, SINK} state;
```

to:

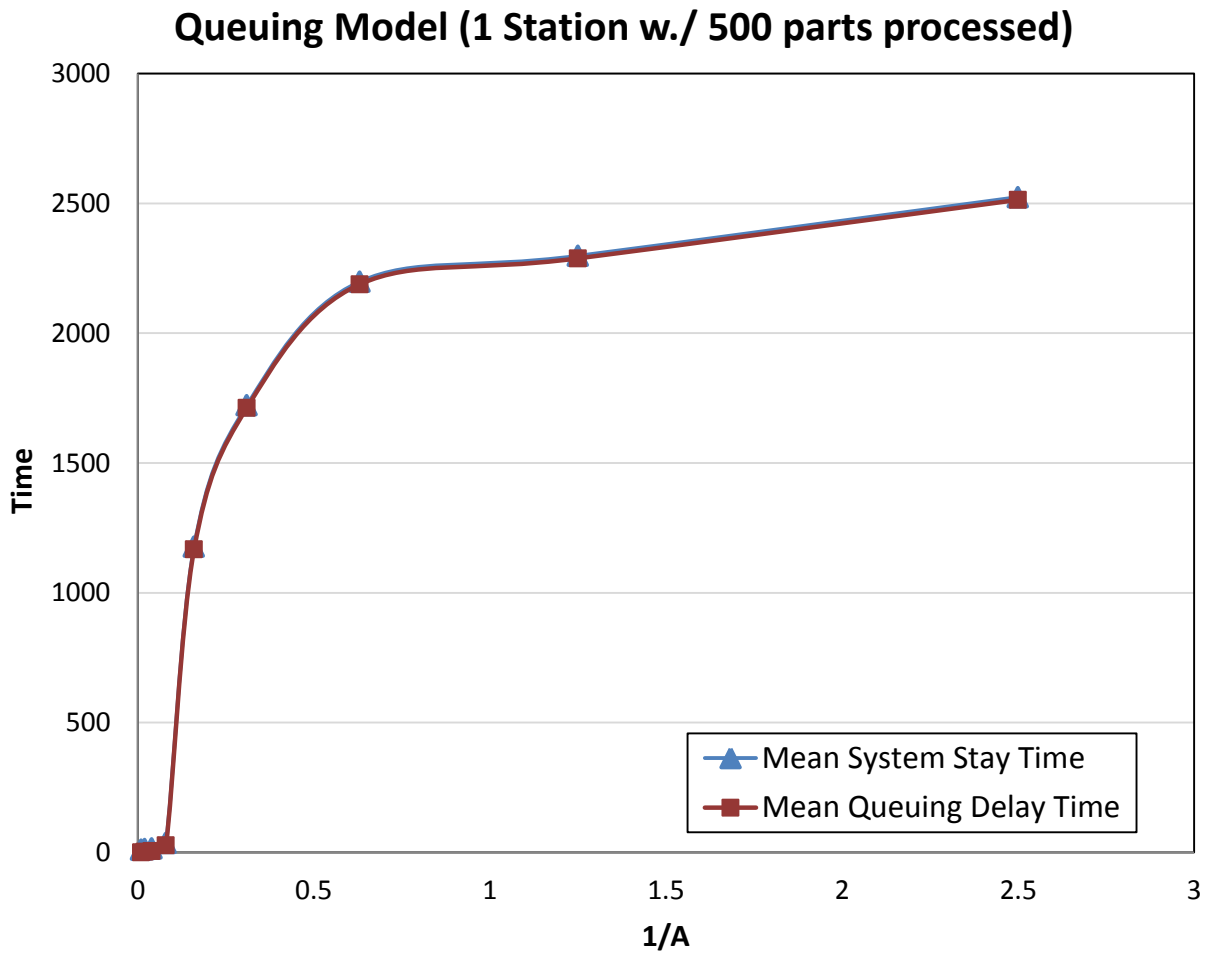
```
#define STATE_NUM 3
typedef enum {SOURCE = 0, STATE1, SINK} state;
```

The new plot (for one station) is attached below:



Due to the random time generation function, the model outputs after each run are slightly different. Below is a sample plot of one set of sample outputs. The trend generally meets the literature results.

To make a more accurate plot, we changed the codes to use a fixed number of 500 parts (much more than the total number of parts with a fixed simulation time of 10000 time units). The new codes are attached in the *"Problem_3-2_codes"* folder, and the new plot is attached below:



Apparently, with more parts being processed, the average system time and queuing delay time increased. Also the general trend agrees with that in the literature. Due to the limited time to finish this assignment, we did not have time to find the analytical solution for the queuing network problem, and hence did not plot the analytical results together with the simulated results from our codes in the same plot for the same set of data.

Literature Comparison

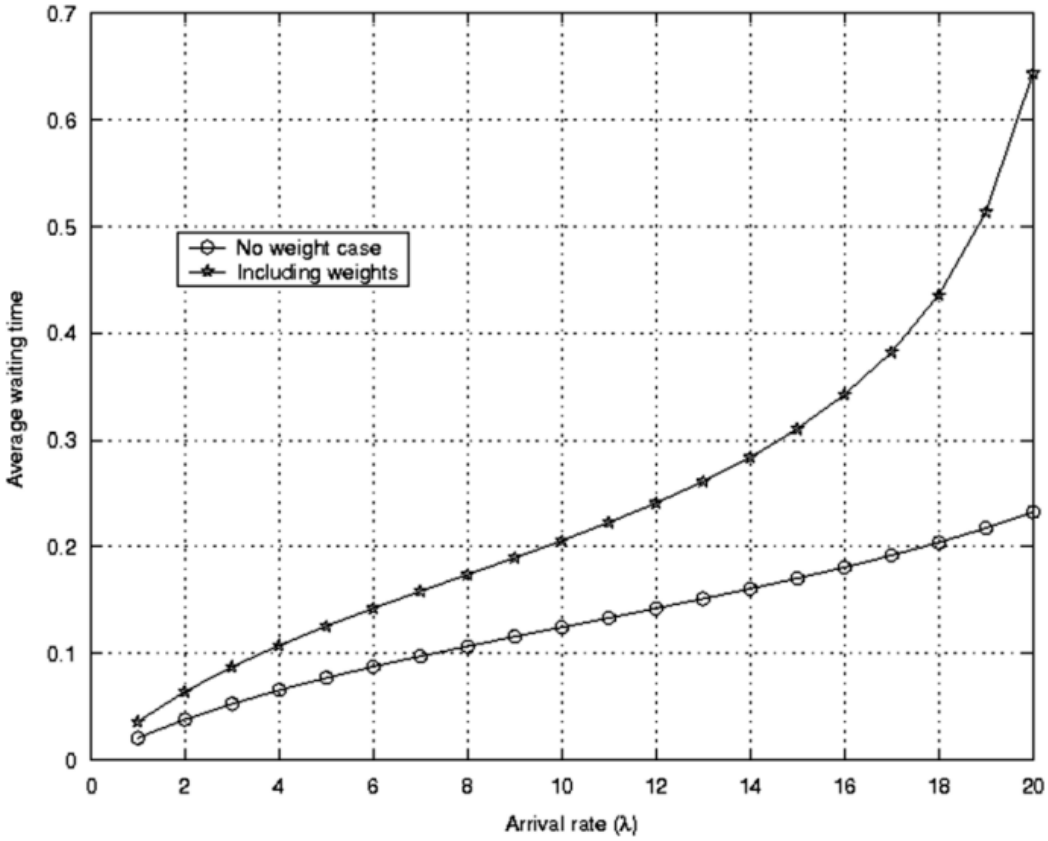


Fig. 6. Comparison of waiting times for the no weight case and the case including weights.

Citation:

Bhaskar, Vidhyacharan, and G. Lavanya. "Equivalent single-queue–single-server model for a Pentium processor." *Applied Mathematical Modelling* 34.9 (2010): 2531-2545.

Prob. 3.3

- Codes for computing time costs are attached as "`execution_time.c`"

- Number of loops: 10,000,000 (10,000,000 insertions and deletions have been performed).

- In the sample code, N(prior queue size) is set to 1000. You can also change the N value (global variable) to check time consumption with different N.

Sample Output

```
Running Log:
-----------------------------------------------
Num of events in the prior queue (N) = 1000
Num of complete operation loops: 10000000
Total execution time(ms):  858.000000
Average time for a single loop(ms):  0.000086
```