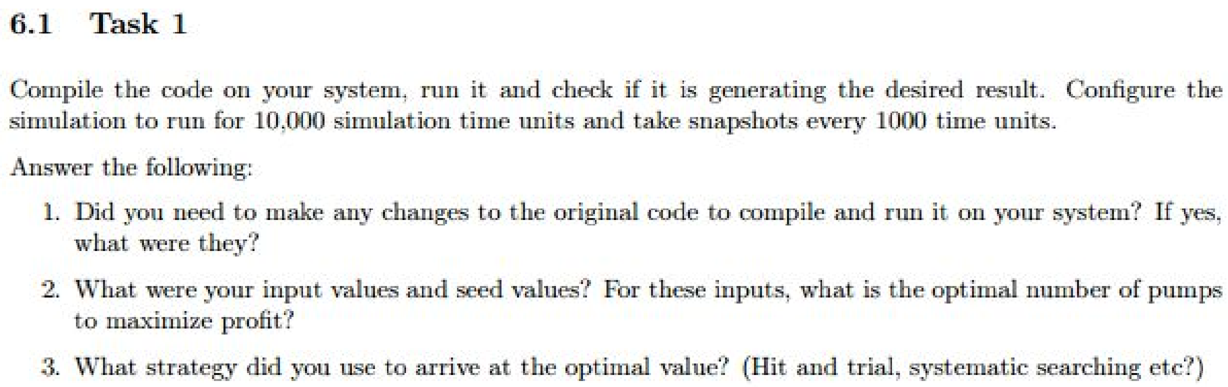
Rachel Law
861071722
CS 177 Modeling and Simulation
Section 21
Winter ‘16

Lab 2 Report



6.1 Task 1

Compile the code on your system, run it and check if it is generating the desired result. Configure the simulation to run for 10,000 simulation time units and take snapshots every 1000 time units.

Answer the following:

1. Did you need to make any changes to the original code to compile and run it on your system? If yes, what were they?
2. What were your input values and seed values? For these inputs, what is the optimal number of pumps to maximize profit?
3. What strategy did you use to arrive at the optimal value? (Hit and trial, systematic searching etc?)

1.

- **modified some header libraries to fit my system**
- **changed implementation of EventListClass**
    - **Queue -> Priority Queue**
- **changed implementation of carClass queue**
    - **Car Queue -> Queue (FIFO)**

```
rlaw@rlaw:~/Documents/cs177/lab2$ diff sim_original.cpp task2.cpp
20,21c20,21
< #include <iostream.h>
< #include <iomanip.h>
---
> #include <iostream>
> #include <iomanip>
24a25,27
> #include <queue>
> #include <vector>
> using namespace std;
127a131,136
> struct CompareEventClassTime {
>         bool operator()(eventClass* lhs, eventClass* rhs) {
>                 return lhs->whatTime() > rhs->whatTime();
>         }
> };
```

```
>
130,131c139,141
<
<	struct eventListItem {
---
>
>	priority_queue<eventClass*, vector<eventClass*>, CompareEventClassTime> eventList;
> /*	struct eventListItem {
135c145
<	eventListItem * firstEvent;
---
>	eventListItem * firstEvent;*/
138c148
<	eventListClass () {firstEvent = NULL;};
---
>	eventListClass () {/*firstEvent = NULL;*/};
146a157,158
>	eventList.push(event);
>	/*
164a177
>	*/
170c183
<
---
>	/*
182a196,199
>	*/
>	eventClass * eventToReturn = eventList.top();
>	eventList.pop();
>	return eventToReturn;
311a329
>	/*
315a334
>	*/
317,318c336,337
<	queueItem * firstWaitingCar, * lastWaitingCar;
<	int localQueueSize;
---
>	//queueItem * firstWaitingCar, * lastWaitingCar;
>	//int localQueueSize;
319a339,340
>
>	queue<carClass*> queueItem;
```

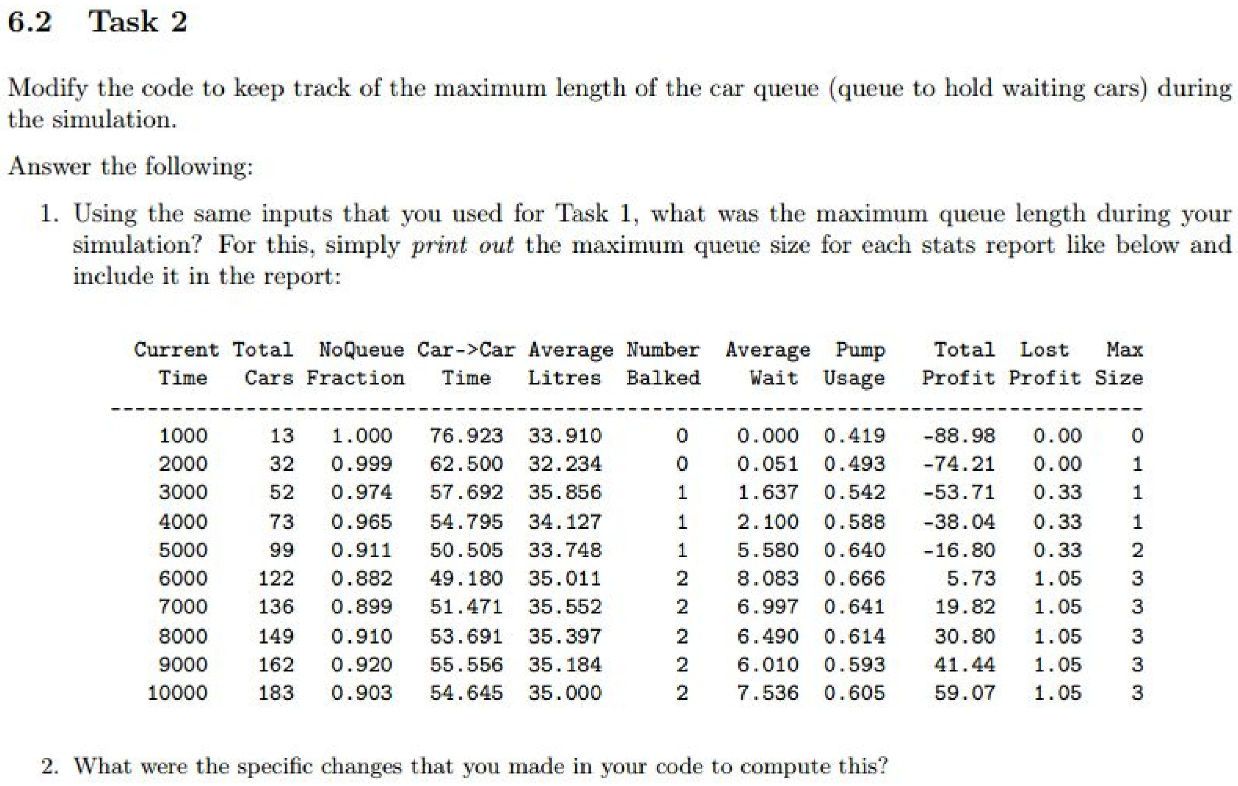
```
332a354
>       /*
335a358
>       */
341c364,365
<       return localQueueSize;
---
>       return queueItem.size();
>       //return localQueueSize;
345a370,372
>       if (queueSize() > 0) return totalEmptyQueueTime;
>       else return totalEmptyQueueTime + simulationTime;
>       /*
349a377
>       */
353a382,386
>       if (queueSize() == 0) {
>               totalEmptyQueueTime += simulationTime;
>       }
>       queueItem.push(newestCar);
>       /*
370a404
>       */
375a410,416
>       if (queueSize() == 0) {
>               return NULL;
>       }
>       carClass* carToReturn = queueItem.front();
>       queueItem.pop();
>
>       return carToReturn;
376a418
>       /*
395a438
>       */
404c447
<       int TotalArrivals, customersServed, balkingCustomers;
---
>       int TotalArrivals, customersServed, balkingCustomers, maxSize;
412a456
>       void accumMaxSize ();
424a469
>       maxSize = 0;
```

```
442a488,493
> void statsClass::accumMaxSize()
> {
>	if (carQueue->queueSize()>maxSize)
>		maxSize += 1;
> }
>
463c514,515
<	printf ("%7.2f\n", TotalLitresMissed * profit);
---
>	printf ("%7.2f", TotalLitresMissed * profit);
>	printf ("%7i\n", maxSize);
562c614,616
<
---
>
>	stats->accumMaxSize();
>
669c723
<	printf ("%9s%7s%8s%9s%8s%7s%9s%7s%8s%7s\n", " Current", "Total ",
---
>	printf ("%9s%7s%8s%9s%8s%7s%9s%7s%8s%7s%7s\n", " Current", "Total ",
671,672c725,726
<		"Total", " Lost ");
<	printf ("%9s%7s%8s%9s%8s%7s%9s%7s%8s%7s\n", "   Time ", "Cars ",
---
>		"Total", " Lost ", " Max ");
>	printf ("%9s%7s%8s%9s%8s%7s%9s%7s%8s%7s%6s\n", "   Time ", "Cars ",
674c728
<		"Usage ", "Profit", "Profit");
---
>		"Usage ", "Profit", "Profit", "Size");
```

2.
Report Interval: 1000
endingTime: 10000
numPumps: 3
seeds: 1 2 3 4

For those inputs, the optimal number of pumps to maximize profit is 3 pumps.

3.
To find the optimal value, I used hit and trial



6.2 Task 2

Modify the code to keep track of the maximum length of the car queue (queue to hold waiting cars) during the simulation.

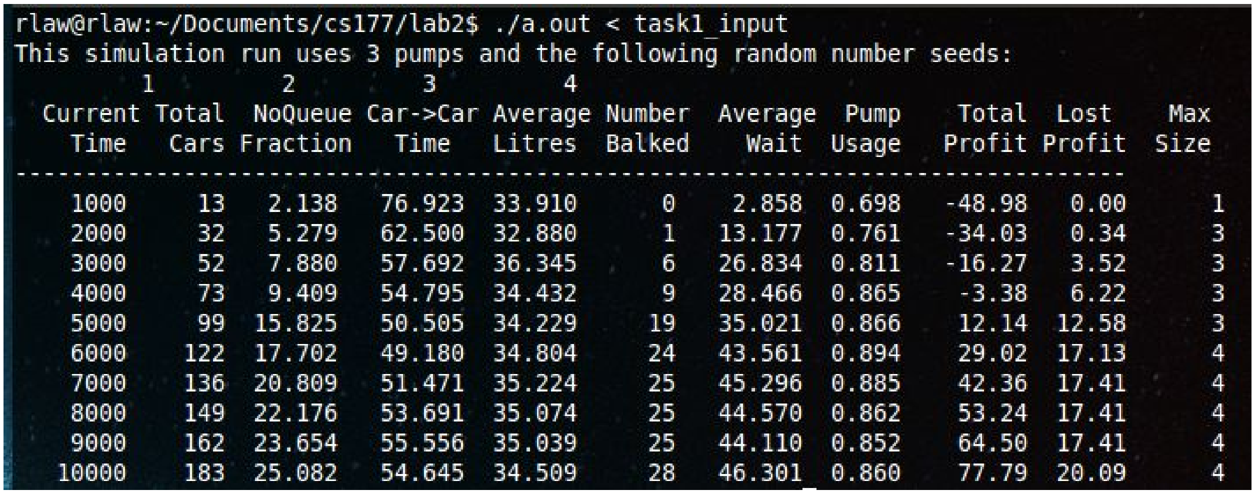
Answer the following:

1. Using the same inputs that you used for Task 1, what was the maximum queue length during your simulation? For this, simply *print out* the maximum queue size for each stats report like below and include it in the report:

```
    Current Total  NoQueue Car->Car Average Number  Average  Pump    Total  Lost   Max
      Time   Cars Fraction   Time   Litres  Balked   Wait   Usage   Profit Profit Size
  ---------------------------------------------------------------------------------------
      1000     13   1.000   76.923  33.910      0    0.000  0.419   -88.98   0.00     0
      2000     32   0.999   62.500  32.234      0    0.051  0.493   -74.21   0.00     1
      3000     52   0.974   57.692  35.856      1    1.637  0.542   -53.71   0.33     1
      4000     73   0.965   54.795  34.127      1    2.100  0.588   -38.04   0.33     1
      5000     99   0.911   50.505  33.748      1    5.580  0.640   -16.80   0.33     2
      6000    122   0.882   49.180  35.011      2    8.083  0.666     5.73   1.05     3
      7000    136   0.899   51.471  35.552      2    6.997  0.641    19.82   1.05     3
      8000    149   0.910   53.691  35.397      2    6.490  0.614    30.80   1.05     3
      9000    162   0.920   55.556  35.184      2    6.010  0.593    41.44   1.05     3
     10000    183   0.903   54.645  35.000      2    7.536  0.605    59.07   1.05     3
```

2. What were the specific changes that you made in your code to compute this?

1.
The maximum queue size was 4



```
rlaw@rlaw:~/Documents/cs177/lab2$ ./a.out < task1_input
This simulation run uses 3 pumps and the following random number seeds:
        1         2         3         4
  Current Total  NoQueue Car->Car Average Number   Average  Pump    Total  Lost    Max
    Time   Cars Fraction   Time   Litres  Balked     Wait  Usage   Profit Profit  Size
-------------------------------------------------------------------------------
    1000     13    2.138   76.923  33.910      0     2.858  0.698   -48.98   0.00      1
    2000     32    5.279   62.500  32.880      1    13.177  0.761   -34.03   0.34      3
    3000     52    7.880   57.692  36.345      6    26.834  0.811   -16.27   3.52      3
    4000     73    9.409   54.795  34.432      9    28.466  0.865    -3.38   6.22      3
    5000     99   15.825   50.505  34.229     19    35.021  0.866    12.14  12.58      3
    6000    122   17.702   49.180  34.804     24    43.561  0.894    29.02  17.13      4
    7000    136   20.809   51.471  35.224     25    45.296  0.885    42.36  17.41      4
    8000    149   22.176   53.691  35.074     25    44.570  0.862    53.24  17.41      4
    9000    162   23.654   55.556  35.039     25    44.110  0.852    64.50  17.41      4
   10000    183   25.082   54.645  34.509     28    46.301  0.860    77.79  20.09      4
```

2.

- **added print statements and stats class functionality to keep track of max car queue size**
- **stats class: added accumMaxSize() and maxSize variable to track max size**
- **call to function during snapshot()**

(These changes are reflected in the diff from answer 1. I forgot to save a task1 and task2 version of the code)