The C Family Continuous Assessment

Report

Description of design decisions made.

Queue (queue.c and queue.h)
I first decided to use a queue as we had already implemented a stack program and I knew it would be easier to convert the stack to a queue than create it from scratch. Once the program was functioning effectively, this could display either side of the road as a list of cars. This means that any observed issues could then be fixed.
Another variable was also added to the program so that the length of time could be recorded that each individual car was waiting for.

Traffic lights (trafficLights.c and trafficLights.h)
The traffic light program was created purely to ensure that the code would never make the traffic lights equal. This also meant that a simple command controlled whether they changed rather than keeping track of what was changed within the code.

Stats (stats.c and stats.h)
The stats program was created to store each statistic that was required for the results when displayed at the end of the program. This meant that the stats could be stored for each side of the road before adding a new structure to combine the results before they are returned.

Run simulation (runSimulations.c and runSimulations.h)
Finally, for the simulation program, the random function that is built into C was used by seeding it at time 0 in the main function. This meant that any new random numbers generated would be truly random. The random function in C was used as it is assumed that this simulation does not need to be scientifically accurate.
In terms of the “runOneSimulation” function inside the ‘for loop’, there is an ‘if statement’ that controls each of the loops. Either the traffic lights will be changed, or a vehicle may arrive from either direction before a vehicle passes through the protected section of the road. Following that process, a ‘while loop’ continues the traffic system without any vehicles arriving to allow the final vehicles to pass through. Then the compiled statistics for that single run are returned.
The “runSimulations” function runs 100 simulations and compiles the stats from these runs. These are output to the user in the required format. There is also some error handling for invalid inputs from the command line.

Description of an experiment performed using the simulator

The first experiment I performed was to assess what happened if one side of the road was busier than the other with a light period to match that side.

Experiment 1:

```
$ runSimulations 0.7 0.2 7 2

Parameter values:
   from left:
       traffic arrival rate: 0.70
       traffic light period:    7
   from right:
       traffic arrival rate: 0.20
       traffic light period:    2
Results (averaged over 100 runs):
   from left:
       number of vehicles:    241
       average waiting time:    0
       maximum waiting:         4
       clearance time:          0
   from right:
       number of vehicles:     68
       average waiting time:    7
       maximum waiting:        27
       clearance time:          6
```

Both sides had a traffic arrival rate with a waiting time to match. The busier left side would clear almost immediately while the right would have to wait for the left side to clear. This is due to how the code changes the traffic lights so that if there are no cars on the right and cars waiting on the left then the lights change to green on the left.
The maximum waiting time on the right is high because, due to a build-up of vehicles on either side and a traffic light period of 2, this will only clear two vehicles each cycle, causing some cars to wait much longer.

```
Parameter values:
   from left:
       traffic arrival rate: 0.70
       traffic light period:    7
   from right:
       traffic arrival rate: 0.20
       traffic light period:    4
Results (averaged over 100 runs):
   from left:
       number of vehicles:    268
       average waiting time:    1
       maximum waiting:         9
       clearance time:          1
   from right:
       number of vehicles:     75
       average waiting time:    3
       maximum waiting:        11
       clearance time:          0
```

A very similar simulation was then run with the right side having double the light period than the first simulation. The results show that these stats would be favourable. Despite the busier side being held up a little more, overall, this simulation finishes quicker than the previous one as well as the maximum wait time being significantly reduced.

```
Parameter values:
   from left:
       traffic arrival rate: 0.70
       traffic light period:    7
   from right:
       traffic arrival rate: 0.20
       traffic light period:    5
Results (averaged over 100 runs):
   from left:
       number of vehicles:    266
       average waiting time:    1
       maximum waiting:        10
       clearance time:          1
   from right:
       number of vehicles:     76
       average waiting time:    2
       maximum waiting:         9
       clearance time:          0
```

After a few more tests it was discovered that having the right side with a traffic light period of 5 gave the most balanced results. As this system is made to control the flow of traffic, having both sides with very similar average and maximum wait times is optimal as it means that the flow of traffic will be constant.

Experiment 2:

```
$ runSimulations 0.5 0.5 5 5

Parameter values:
   from left:
       traffic arrival rate: 0.50
       traffic light period:    5
   from right:
       traffic arrival rate: 0.50
       traffic light period:    5
Results (averaged over 100 runs):
   from left:
       number of vehicles:    191
       average waiting time:    9
       maximum waiting:        25
       clearance time:         12
   from right:
       number of vehicles:    192
       average waiting time:    8
       maximum waiting:        23
       clearance time:         10
```

In terms of testing equal traffic arrival rates, the first simulation began with a traffic light period of 5 for both sides. After these results, the next few tests had lower light periods however, these increase all the of values of the results significantly.

```
Parameter values:
   from left:
       traffic arrival rate: 0.50
       traffic light period:   10
   from right:
       traffic arrival rate: 0.50
       traffic light period:   10
Results (averaged over 100 runs):
   from left:
       number of vehicles:    211
       average waiting time:    8
       maximum waiting:        25
       clearance time:         14
   from right:
       number of vehicles:    212
       average waiting time:    8
       maximum waiting:        24
       clearance time:         11
```

Following these results, the next few tests had traffic light periods that incrementally increased by 1 on each side for each test.
To conclude, the tests exposed that as long as the light period is equal for both sides, the traffic lights will perform to the same standard, down to a period of 5.

Experiment 3:

```
$ runSimulations 1 0.1 10 2

Parameter values:
   from left:
       traffic arrival rate: 1.00
       traffic light period:   10
   from right:
       traffic arrival rate: 0.10
       traffic light period:    2
Results (averaged over 100 runs):
   from left:
       number of vehicles:    416
       average waiting time:   23
       maximum waiting:        51
       clearance time:         49
   from right:
       number of vehicles:     41
       average waiting time:   39
       maximum waiting:        82
       clearance time:         52
```

The final test explored the two most extreme arrival rates. Initially it was thought that there should be 500 cars on the left side as it was guaranteed for a car to arrive, but the results did not reflect this because the system will either change the lights or add a car. Therefore, there cannot be 500 cars if it isn't guaranteed to add a car every single cycle.

These results were not optimal, so the next few tests were performed to improve them. There were tests done with various different traffic light periods increasing and decreasing both values.

```
$ runSimulations 1 0.1 10 3

Parameter values:
   from left:
       traffic arrival rate: 1.00
       traffic light period:   10
   from right:
       traffic arrival rate: 0.10
       traffic light period:    3
Results (averaged over 100 runs):
   from left:
       number of vehicles:    416
       average waiting time:   26
       maximum waiting:        60
       clearance time:         59
   from right:
       number of vehicles:     41
       average waiting time:    7
       maximum waiting:        20
       clearance time:          2
```

This was discovered to be the simulation that could most reduce average and maximum waiting time for both sides. With a system as extreme as this there is not a way to reduce the wait times enough without sacrificing the waiting time for one of the sides.

Example output from at least one run of your code.

```
$ compileSim
$ runSimulations 0.6 0.4 6 5

Parameter values:
   from left:
       traffic arrival rate: 0.60
       traffic light period:    6
   from right:
       traffic arrival rate: 0.40
       traffic light period:    5
Results (averaged over 100 runs):
   from left:
       number of vehicles:    235
       average waiting time:   10
       maximum waiting:        28
       clearance time:         16
   from right:
       number of vehicles:    156
       average waiting time:    6
       maximum waiting:        20
       clearance time:          4
```

This shows the code being compiled using “compileSim” and then run with the “runSimulations” command.

When using the “runSimulations”, 4 parameters are required as inputs on the command line.
The first 2 numbers are the traffic arrival rate for the left and right respectively.
The second 2 numbers are the traffic light period for the left and right respectively.