ECM2433 Simulation Report

Design choices and assumptions

In my code for the simulation, I chose to have a variety of structs to handle data and represent data structures. As I was simulating traffic lights, this meant I had to simulate 2 queues in each direction of traffic. The queues were made using linked lists, specifically I had a node struct and a queue struct to do this. The node struct contained the time since the start of the simulation that it was added to the queue and the next node in the queue. This was the linked list. The queue struct contained front and rear node pointers aswell as the number of nodes in the queue so that nodes could be inserted and deleted to the front and back of the queue, like in a real queue of cars. I also decided to create a struct called stats to store information about 2 queues. This made it easier to pass the data around the program as one condensed object.

For the random decisions the program had to make, I chose to use GSL to generate the random numbers as the numbers it generated were of a high quality and it was easy to get values between 0 and 1 for the probabilities of car movements. One assumption I made for this part was in when a car will pass through the lights when its side is green, as I set it to a flat 50% chance. It was not specified for it to be configurable so be making it 50/50, it kept the program simple.

Description of an experiment with the sim and its results

To test the functionality of the program, I ran it with 2 different settings to compare how the initial configurations affect the output.

For the first one I had the rate of traffic arriving at 50% per time unit and the lights both stayed green for 5 seconds before switching. The second one had the same arrival rate however I upped the light period to 10 time units.

```
$ ./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:   201
      average waiting time: 236
      maximum waiting time: 508
      clearance time:       507
   from right:
      number of vehicles:   201
      average waiting time: 344
      maximum waiting time: 435
      clearance time:       430
$
```

I hypothesised that with longer to go through the lights, the cars would filter through the system faster as there was more decision time for each queue. What actually happened though was that it took longer for cars to get through, which can be seen by an increased average waiting time for both sides of the lights and a higher build up of cars on both sides.

This shows that for a system like this, more frequent small intervals of movement allows traffic to flow faster than longer, less frequent ones.

```
$ ./runSimulations 0.5 0.5 10 10
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:   234
      average waiting time: 322
      maximum waiting time: 614
      clearance time:       608
   from right:
      number of vehicles:   250
      average waiting time: 340
      maximum waiting time: 600
      clearance time:       598
$
```

Errors and debugging

The program, unfortunately doesn’t work fully as intended. Sometimes when it is run the average wait time will be higher than the maximum wait time, which should not be possible. To try solve this I would put print statements throughout the program as well as using the display function I made to print out the contents of a queue with the time values to see how this result happened. Unfortunately I was unable to resolve this issue.