ECM2433 Coursework Report

1 Design Decisions

1.1 Queues

To implement the queues in my program, I have created a linked list. Each car is represented as a node which stores the car's wait time and a 'next' attribute which recursively references the next node. I have used a structure to define this.

```
struct node
{
    int val; /* Time car has queued. */
    struct node * next; /* Next car in queue */
};
typedef struct node QUEUE;
```

Figure 1: The queue's structure definition in queue.h.

I have chosen to use a linked list as they have dynamic size. Due to the random element, we do not know how many cars will be in the queue at a given time. Having a linked list means that memory will not be wasted or reallocated to a bigger size depending on the queue length.

The queue has a few functions. The functions 'carcome()' and 'cargo()' are the push and pop functions of our queue. They allocate and free space when an item is added to the tail of the queue or removed from the head. The function 'wait()' increments through the cars in the queue and increases their wait time by one. The 'createQueue()' function is used to mark the queue as an empty queue when it is first created via 'malloc()'. There is also a 'printQueue()' function which prints all the values in the queue, though this is not used and is there for debugging purposes.

With the queue I have assumed that it takes the same amount of time for each car to leave the queue, the cars do not have different speeds/accelerations. In addition, I have assumed that cars only do what is expected of them. The program does not take into account people who go through just as the light turns red or do not see the light change to green and stay still for a few seconds. It also doesn't take into account if a car were to break down. turn around, or do some other non expected behavior.

1.2 Simulation

I have created another structure called a simulation. These are designed to store the results of the function 'runOneSimulation()'. It stores the number of vehicles, average wait time, maximum wait time, and time taken to clear the queue on both the left and the right.

I have created this structure so that the results can be returned to the 'runSimulations()' function as you cannot pass multiple variables back from a function in C.

```
struct simStruct
{
    int timeTaken; /* Time taken for simulation to run. */
    int totalCarsL; /* The number of vehicles that passed through the traffic
        lights. */
    float averageWaitL; /* The average waiting time. */
    int maxWaitL; /* The maximum waiting time. */
    int clearTimeL; /* The time taken to clear the queue once vehicles have stopped
         arriving. */
    int totalCarsR; /* The number of vehicles that passed through the traffic
        lights. */
    float averageWaitR; /* The average waiting time. */
    int maxWaitR; /* The maximum waiting time. */
    int clearTimeR; /* The time taken to clear the queue once vehicles have stopped
         arriving. */
};
typedef struct simStruct SIMULATION;
```

Figure 2: The simulation's structure definition in simulation.h.

2 Experiments

This section will provide a description of an experiment that I have performed using the simulator. I will showcase the results that have been obtained and discuss what they mean.

2.1 Scenarios - the effect of changing the car arrival probability

I have experimented with the simulator by seeing the effect of changing the probability that cars will join either queue. Firstly I chose a few example scenarios which involve different probability values. For each of these scenarios the traffic light period equal 40 time units for the left and 40 time units for the right.

2.1.1 Equally Busy Road:

In this scenario, both sides of the road have an equal 75% chance of having a car arrive.

```
Parameter values:
    from left:
        traffic arrival rate: 0.750000
        traffic light period: 40
    from right:
        traffic arrival rate: 0.750000
        traffic light period: 40
Results (averaged over 100 runs):
    from left:
        number of vehicles: 364.029999
        average waiting time: 114.842529
        maximum waiting time: 241.600006
        clearance time: 233.320007
    from right:
        number of vehicles: 365.399994
        average waiting time: 137.953079
        maximum waiting time: 267.880005
        clearance time: 260.790009
```

Figure 3: The result of './runSimulations 0.75 0.75 40 40'.

2.1.2 Equally Quiet Road:

This scenario is similar to the one before, instead each queue has an equal 25% chance of having a car arrive.

```
Parameter values:
    from left:
        traffic arrival rate: 0.250000
        traffic light period: 40
    from right:
        traffic arrival rate: 0.250000
        traffic light period: 40
Results (averaged over 100 runs):
    from left:
        number of vehicles: 122.959999
        average waiting time: 12.803413
        maximum waiting time: 38.799999
        clearance time: 1.030000
    from right:
        number of vehicles: 121.089996
        average waiting time: 13.839461
        maximum waiting time: 39.290001
        clearance time: 25.180000
```

Figure 4: The result of './runSimulations 0.25 0.25 40 40'.

2.1.3 Morning Commute into the City:

In this scenario, I aim to simulate the conditions where commuters head into the city in the morning. The left queue (from the suburbs) has a 85% chance of a car joining whereas the right queue (from the city) has a 25% chance of a car joining.

```
Parameter values:
    from left:
        traffic arrival rate: 0.850000
        traffic light period: 40
    from right:
        traffic arrival rate: 0.250000
        traffic light period: 40
Results (averaged over 100 runs):
    from left:
        number of vehicles: 413.589996
        average waiting time: 157.072220
        maximum waiting time: 329.459991
        clearance time: 324.200012
    from right:
        number of vehicles: 121.629997
        average waiting time: 14.058442
        maximum waiting time: 39.240002
        clearance time: 25.250000
```

Figure 5: The result of './runSimulations 0.85 0.25 40 40'.

2.1.4 Natural Disaster (Godzilla Attack):

In this scenario, I aim to simulate the conditions where people evacuate from a city. The left queue (from the suburbs) has a 1% chance of a car joining, as almost nobody wants to head into the emergency whereas the right queue (from the city) will always (100% chance) have a car join.

```
Parameter values:
    from left:
        traffic arrival rate: 0.010000
        traffic light period: 40
    from right:
        traffic arrival rate: 1.000000
        traffic light period: 40
Results (averaged over 100 runs):
    from left:
        number of vehicles: 4.790000
        average waiting time: 10.203516
        maximum waiting time: 25.290001
        clearance time: 1.000000
    from right:
        number of vehicles: 487.000000
        average waiting time: 261.601593
        maximum waiting time: 515.000000
        clearance time: 514.000000
```

Figure 6: The result of './runSimulations 0.01 1 40 40'.

2.2 Arrival Probability Scenario Results

As these scenarios show, the higher the probability that cars arrive, the longer the average and max waiting times. This makes sense as the queue will be longer and so the time that the cars are waiting will increase.

Of course, the period of the traffic lights also has an effect. Even though a car was always added to the queue in Scenario 4, only 487 cars passed through the light on the right side. This is because cars cannot arrive in the time that the lights change in our simulation. As the time period was 40 time units for these scenarios, there are 13 times where cars could not arrive hence the 487 vehicles that passed the right light.

The clearance time is also related to the number of vehicles in the queue and so is changes based on the probability that a car arrives. The clearance time is quite close to the maximum wait as it is usually this last car that has the longest wait time. The right side often has the higher maximum wait and clearance times, which is most likely due to the left side always starting with the green light in my program and with an even number of traffic light swaps, the last light to be on.

2.3 Scenario - changing the traffic light period

I wanted to run an experiment to see the change that setting wildly different traffic light periods would have. One traffic light is only on for 10 time units whereas the other is on for 100. The

expected result is that the side with the larger period should have virtually no wait and the side with the small period will have dramatic wait times. Both queues have a 50% chance of a car arriving.

```
Parameter values:
    from left:
        traffic arrival rate: 0.500000
        traffic light period: 10
    from right:
        traffic arrival rate: 0.500000
        traffic light period: 100
Results (averaged over 100 runs):
    from left:
        number of vehicles: 245.009995
        average waiting time: 1204.808105
        maximum waiting time: 2451.510010
        clearance time: 2445.989990
    from right:
        number of vehicles: 244.649994
        average waiting time: 0.849421
        maximum waiting time: 9.470000
        clearance time: 1.000000
```

Figure 7: The result of './runSimulations 0.50 0.50 10 100'.

2.4 Traffic Light Period Scenario Results

As *Figure 7* above shows, the results from this scenario were what we were expecting. The right side (with the larger period) has an average wait time of about 0.85 time units. This means that most cars were not queuing and just going straight through the lights. The cars on the left side (the small period) were not so lucky. The average waiting time on this side was over 1200 time units with the max wait being over double that at around 2450 time units.

The smaller the traffic light periods, the quicker the queue clears. This is because not only do the cars leave their queues more frequently, but less cars can arrive as a time period is taken up changing the lights. As you increase the traffic light period, the queues will take longer to clear as more cars can arrive and have to queue.

To further experiment with this, I have run the command './runSimulations 0.50 0.50 x x' for multiple values of x up to 500. A plot of the results can be found below:

The effect on the average wait time of a simulation by changing the value of x in './runSimulations 0.50 0.50 x x':

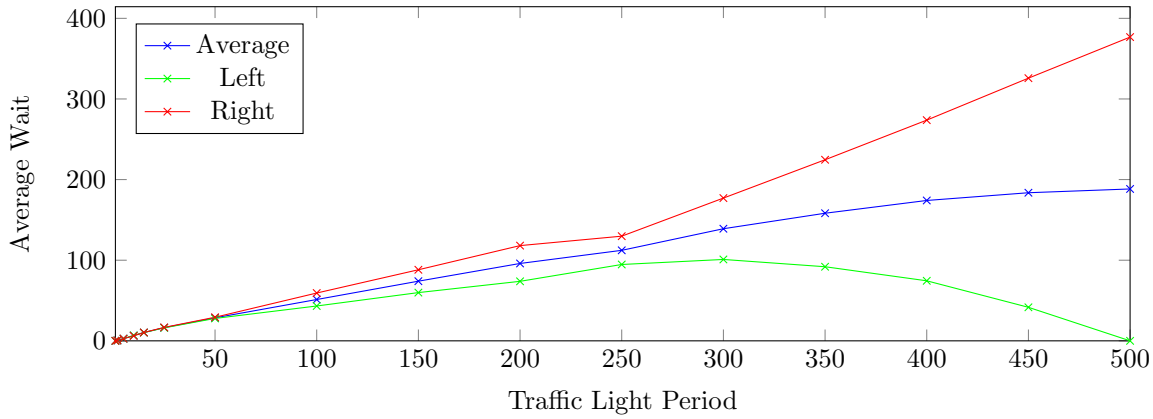

Figure 8: The result of './runSimulations 0.50 0.50 x x'.

As you can see, the values stay pretty similar up to 250, when they begin to diverge. This is because there are no longer 2 even periods in which cars arrive, there will now be time in which cars do not arrive within the first right period. More cars in the left side can head straight through, increasing the wait for the right cars.

I think seeing more combinations of different traffic light periods would be a fascinating further experiment. However, due to the countless permutations, I will not be conducting this at present due to the amount of time required.