Course Project Report

Name: Morgan Krueger & James Fenimore

- **Program code explanation.**
  In this section, please answer the following questions by providing the corresponding pseudo codes if necessary. You may add extra explanation if needed.
  1. How many threads did you create, and what are their specific functionalities? Describe these based on pseudocode.
     a. We modeled our thread creation based on the parallel thread creation from the lectures (MO2d thread).

        ```
        For(x<=number of cars; x++){
            Create thread(x)
        }
        For(y<=number of cars; y++)
            Join thread(y)
        }
        ```

        There is a thread created for each car as the problem stipulated, we must create. Each thread acts like a car in a rollercoaster line rotating through every minute and subtracting from the queue.
  2. What is the duration of your virtual minute in the simulation (e.g., 10 ms, 100 ms)?
     a. We simulated our clock as one big iterator. There is no real time value being utilized or synchronization with a system clock. We are simply simulating the day by an iterator and loops cycling through their logic to simulate time passing.

        ```
        Clock iterator{
        While(current time<max time){
            Current time++
            (additional logic that needs to happen upon every simulated minute including
        synchronization)
        }}
        ```

        With this implementation we are simplifying the process of synchronization because this for loop will run, and it will iterate every time without causing a race condition when all processes have signaled each other to go with pthread broadcast and mutex locks in their respective functions.

  3. How many semaphores, mutex locks, and condition variables did you use? Explain the reasons for using these synchronization objects.
     a. We utilized one shared mutex, and two conditional variables. The conditional variables were covered in M04c slide 14-18, and the shared mutex was covered in M04c slide 8-13. With conditional wait we can have one process start when another says to go. Utilizing the pthread broadcast allows a process to communicate to all the threads of an associated process, which was key with multiple car threads active at one time. We can think of this problem of having two stoplights, one for the line and one for the ride itself. These stoplights cannot be green at the same time and

can only work when the other says they are ready to go, or is green. When the line light is red the ride is going. When the line light is green the cars are being filled until they are full, which the ride light turns to red stopping the line from being subtracted.

```
Clock{
    while(current time< max time){
    Lock shared mutex
    Signal conditional variable 1
    Wait conditional variable 2
    Iterate minute
    Unlock shared mutex
}}
riding{
while(current time< max time){
    lock shared mutex
    subtract people based on car number and capacity
    if full
    signal conditional variable 2
    wait conditional variable 1
    unlock shared mutex
 }}
```

4. Are there any race conditions in the project? If so, how do you avoid them?
    a. Yes, there are race conditions when establishing the arrival rate, queue, and the number of people per car. We avoided the race conditions by utilizing the mutex lock and the two conditional variables described above. This led us to properly design the code to avoid any unnecessary shared resource access. By utilizing broadcast specifically in

```
Clock{
    while(current time < max time){
            lock shared mutex
            call poissondefs()
            signal thread to continue
            signal threat to wait
            unlock shared mutex
    }
    Signal thread to continue
}
```

    This ensures that only one thread has access to the critical section, which being the queue resource, at a time until an event occurs. In our logic, we signal all threads and wait for one to finish its event which notifies the next thread to wait until the signal is received from the first thread. This ensures that the second thread will wait for synchronization before continuing. Ultimately, allowing only one thread at a time to execute a block of code that is interacting with our shared resources.

5. How do you synchronize the threads every minute?
    a. The synchronization is achieved by the proper implementation of conditional variables and semaphore lock. The simulated minute does not move until all threads and processes

are in a steady state, which is signaled by the conditional variables and permitted by the shared mutex lock to proceed onward on iterating the minute.

- **Table output**

(N,M)	Total Arrival	Total GoAway	Rejection Ratio	Average Wait Time (mins)	Max Waiting Line
N=2, M=7	29481	10549	0.357824	69.65106	824
N=2, M=9	27089	8157	0.301119	59.863991	820
N=4, M=7	22024	3092	0.140392	29.28762	805
N=4, M=9	19820	888	0.44803	27.154348	800
N=6, M=7	18932	0	0	15.584666	606
N=6, M=9	18932	0	0	7.430295	65

For N=2 M=7

```
queue is:800.000000 at time:598
broadcast
car go
last car go
wait

queue is:800.000000 at time:599
broadcast
car go
last car go
wait

We are CLOSED!
Total Arrival, Total Rejected, Reject Ratio,  Average Wait,   Max Waiting Line
29481.0,         10549.0,        0.357824,       69.651016,      824
PS C:\Users\Morgan\Documents\437\ECE437Project\ECE437Project>
```

For N =2 M=9

```
queue is:800.000000 at time:598
broadcast
car go
last car go
wait

queue is:800.000000 at time:599
broadcast
car go
last car go
wait

We are CLOSED!
Total Arrival, Total Rejected, Reject Ratio,  Average Wait,   Max Waiting Line
27089.0,         8157.0,         0.301119,       59.863991,      820
PS C:\Users\Morgan\Documents\437\ECE437Project\ECE437Project>
```

For N=4 M=7

```
queue is:239.000000 at time:599
broadcast
car go
car go
car go
last car go
wait

We are CLOSED!
Total Arrival, Total Rejected, Reject Ratio,  Average Wait,   Max Waiting Line
22024.0,         3092.0,         0.140392,       29.287682,      805
PS C:\Users\Morgan\Documents\437\ECE437Project\ECE437Project>
```

For N=4 M=9

```
queue is:27.000000 at time:599
broadcast
car go
car go
car go
last car go
wait

We are CLOSED!
Total Arrival, Total Rejected, Reject Ratio,  Average Wait,   Max Waiting Line
19820.0,         888.0,  0.044803,       27.154348,      800
```

For N=6 M=7

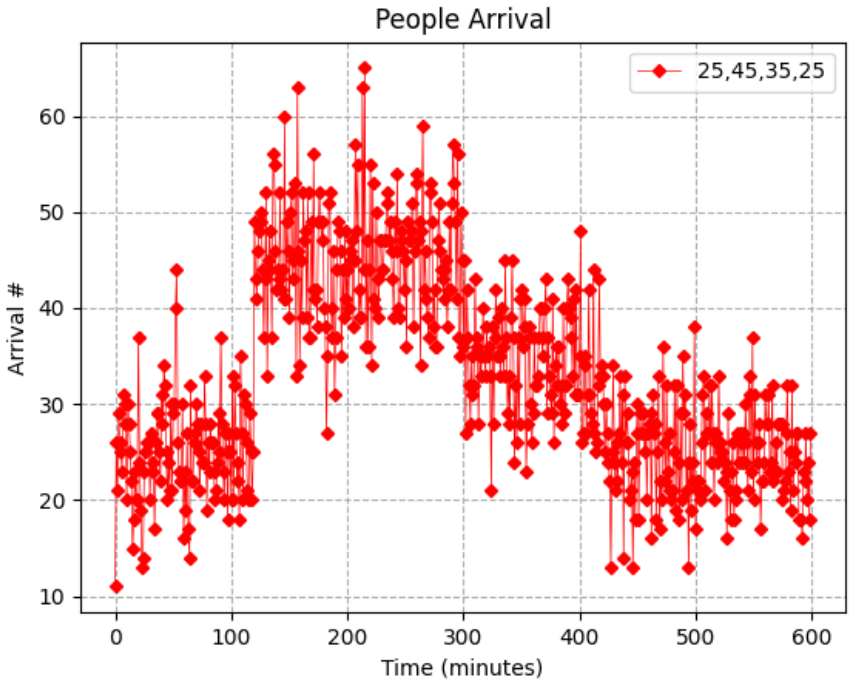
```
queue is:27.000000 at time:599
broadcast
car go
car go
car go
car go
car go
last car go
wait

We are CLOSED!
Total Arrival, Total Rejected, Reject Ratio,  Average Wait,   Max Waiting Line
18932.0,         0.0,    0.000000,       15.584666,      606
```

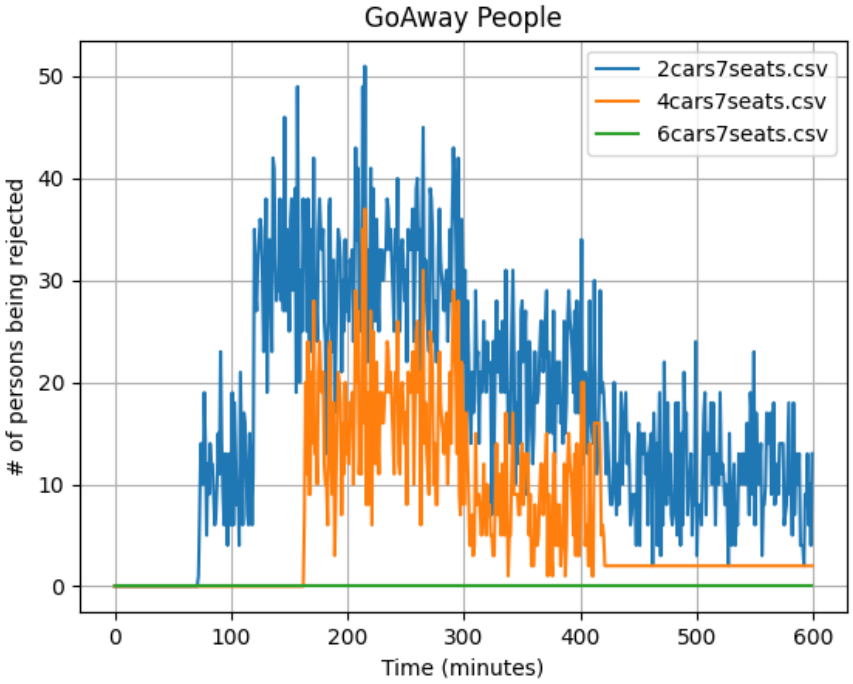
For N=6 M=9

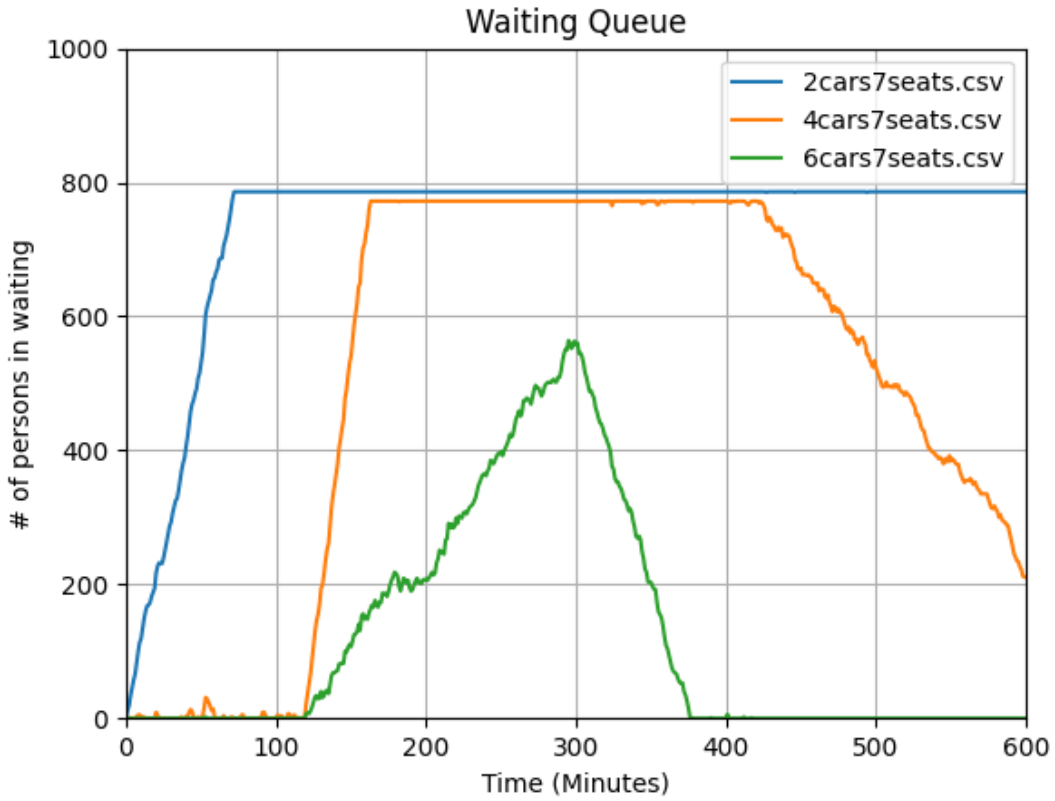
```
queue is:27.000000 at time:599
broadcast
car go
car go
car go
car go
car go
last car go
wait

We are CLOSED!
Total Arrival, Total Rejected, Reject Ratio,  Average Wait,   Max Waiting Line
18932.0,         0.0,    0.000000,       7.430295,       65
```

Figure outputs and related descriptions.

When using our code, please run the main.c by compiling with ./main -N 2 -M 7 for all N and M values. This will then produce the .csv file for each car number and seats number. These .csv files can then be used in the python code to produce the following figures. If using different values than what is required for the assignment, ensure to change the name of the file within the python code. For our python code, we utilized chat GPT to help us create the data parsing for our CSV sheets in python, as neither of us are proficient in python. The following lines were logic that we took inspiration from Chat GPT on, lines 11-15, 28-32, 43-46.







Here are the screenshots of the output data to the .csv files
For N=2 M=7

2cars7seats.csv > data

You, 3 minutes ago | 1 author (You)

```
TimeMins, TotalArrivalPerMinute, WaitingQueue, Rejected
0, 11.0, 0.0, 0
1, 26.0, 12.0, 0
2, 21.0, 19.0, 0
3, 29.0, 34.0, 0
4, 26.0, 46.0, 0
5, 25.0, 57.0, 0
6, 23.0, 66.0, 0
7, 31.0, 83.0, 0
8, 30.0, 99.0, 0
9, 28.0, 113.0, 0
10, 20.0, 119.0, 0
11, 30.0, 135.0, 0
12, 28.0, 149.0, 0
13, 25.0, 160.0, 0
14, 22.0, 168.0, 0
15, 15.0, 169.0, 0
16, 18.0, 173.0, 0
```

For N=4 M=7

4cars7seats.csv > data

You, 5 minutes ago | 1 author (You)

```
TimeMins, TotalArrivalPerMinute, WaitingQueue, Rejected
0, 11.0, 0.0, 0
1, 26.0, 0.0, 0
2, 21.0, 0.0, 0
3, 29.0, 1.0, 0
4, 26.0, 0.0, 0
5, 25.0, 0.0, 0
6, 23.0, 0.0, 0
7, 31.0, 3.0, 0
8, 30.0, 5.0, 0
9, 28.0, 5.0, 0
10, 20.0, 0.0, 0      You, 2 days ago • addding finalized code and c
11, 30.0, 2.0, 0
12, 28.0, 2.0, 0
13, 25.0, 0.0, 0
14, 22.0, 0.0, 0
15, 15.0, 0.0, 0
16, 18.0, 0.0, 0
```

For N=6 M=7

6cars7seats.csv > data

You, 6 minutes ago | 1 author (You)

```
TimeMins, TotalArrivalPerMinute, WaitingQueue, Rejected
0, 11.0, 0.0, 0
1, 26.0, 0.0, 0
2, 21.0, 0.0, 0
3, 29.0, 0.0, 0
4, 26.0, 0.0, 0
5, 25.0, 0.0, 0
6, 23.0, 0.0, 0
7, 31.0, 0.0, 0
8, 30.0, 0.0, 0
9, 28.0, 0.0, 0
10, 20.0, 0.0, 0
11, 30.0, 0.0, 0
12, 28.0, 0.0, 0
13, 25.0, 0.0, 0
14, 22.0, 0.0, 0
15, 15.0, 0.0, 0      You, 2 days ago • addding finalized code and c
16, 18.0, 0.0, 0
```