Trevor Bright
CSE 622
5066452

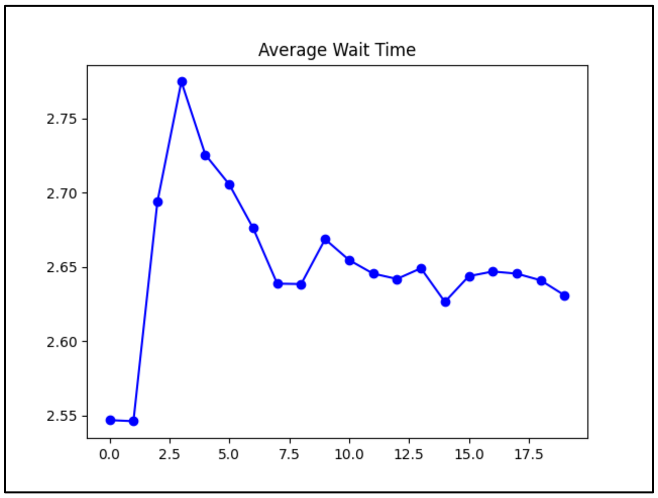
Final Project

1. **Result Summary**

After 20 runs of the simulation were ran, this was the overall averages for the wait time, number of people that walked to the second, third, and fourth floor, number of people waiting at 8:30, 8:45, and 9:00, as well as the time the last worker got onto the elevator at 9:00.

```
-----OVERALL AVERAGES-----
Wait time: 39.78078765084125 mins
Walked to second floor: 104.05
Walked to third floor: 84.45
Walked to fourth floor: 37.1
Waiting at 8:30 AM: 49.7
Waiting at 8:45 AM: 58.4
Waiting at 9:00 AM: 59.2
Last time each day: ['No workers', 'No workers', 'No workers', 'No workers', 'No workers', 'No workers',
'No workers', 'No workers', 'No workers', '8:59:45 AM', 'No workers', 'No workers', 'No workers', 'No
workers', '8:57:00 AM', '8:59:30 AM', 'No workers', 'No workers', 'No workers', 'No workers']
```

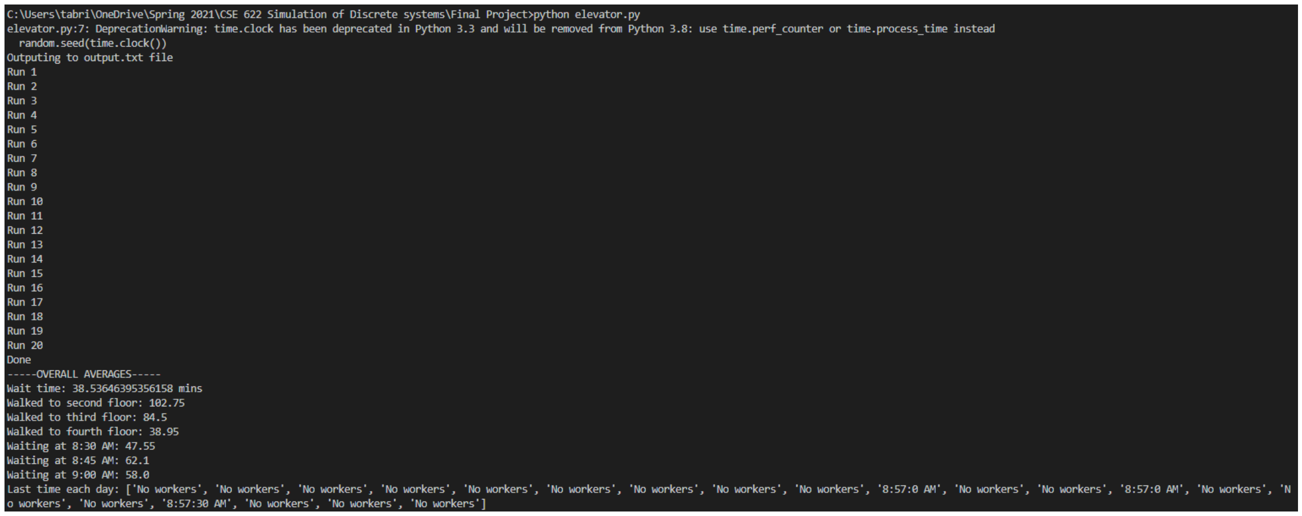
As you can see from the averages, the wait time is very long for the average person trying to get to work. Since there is about 6 people coming every minute, there is a possibility of 354 people from 8am-9am. Only one elevator that holds 12 people is not enough for that volume of workers. If I was a worker in this building, I would take the stairs every time if there was only one elevator. If I was the owner of the building, I would add a lot more elevators to properly get these people where they need to go in time.



This graph shows the average wait time between all 20 runs. The actual time in minutes can be found by taking the y value and multiplying by 15. The individual run data for all the outputs can be found in the output.txt file.

2. **Implementation Summary**

For this project, I used Python, coding using Visual Studio code and compiling in a terminal. I used mainly classes for my data structures to keep everything in order. I had a class for the building, containing the queue of people waiting, a list of people that walked up the stairs and arrived using the elevator, and an Elevator object. The elevator object had a list of people riding it, its current location, its destination, and its wait time. Each person was an object as well, having their wait time, time they got on the elevator, a Boolean variable indicating if they've tried taking the stairs, and their destination. I decided to run it 20 times due to how long each run takes and how much memory each run takes up. Since all variables in each run is saved, this adds up quick. Here is an example output



```
C:\Users\tabri\OneDrive\Spring 2021\CSE 622 Simulation of Discrete systems\Final Project>python elevator.py
elevator.py:7: DeprecationWarning: time.clock has been deprecated in Python 3.3 and will be removed from Python 3.8: use time.perf_counter or time.process_time instead
  random.seed(time.clock())
Outputing to output.txt file
Run 1
Run 2
Run 3
Run 4
Run 5
Run 6
Run 7
Run 8
Run 9
Run 10
Run 11
Run 12
Run 13
Run 14
Run 15
Run 16
Run 17
Run 18
Run 19
Run 20
Done
-----OVERALL AVERAGES-----
Wait time: 38.53646395356158 mins
Walked to second floor: 102.75
Walked to third floor: 84.5
Walked to fourth floor: 38.95
Waiting at 8:30 AM: 47.55
Waiting at 8:45 AM: 62.1
Waiting at 9:00 AM: 58.0
Last time each day: ['No workers', 'No workers', 'No workers', 'No workers', 'No workers', 'No workers', 'No workers', 'No workers', 'No workers', '8:57:0 AM', 'No workers', 'No workers', '8:57:0 AM', 'No workers', 'N
o workers', 'No workers', '8:57:30 AM', 'No workers', 'No workers', 'No workers']
```

3. **Critical Thinking Summary**
The most difficult part of this project was implementing all the requirements, especially the walking requirement. I was confused on how to only check once when the person initially got there. I solved this by adding a Boolean variable to my Person class and checking on every iteration to see if there are any people that have that variable set to false. If I did this project again, I would try to optimize it. It takes up a lot of memory and takes a long time to run. I could cut down the amount of data that is collected and only take the minimum amount to save memory. I would also like to add the ability to have multiple elevators to see how that effects the outcome and averages. I learned a lot about implementing different restrictions such as the different probabilities of taking the stairs instead of waiting for the elevator based on which floor they want to go to. I can use this in future probability programs that I create.