Aaron Fox
CECS 622
Dr. Elmaghraby

Final Project (Elevators Project)

Introduction

A simulation program was written in Python using several imported modules for graphing, randomness, and math, such as matplotlib and seaborn, in order to carry out an elevator simulation. This simulation required the modeling of workers taking the elevator during the morning hours of 08:00 to 09:00. The elevator has a capacity of 12 people, requiring those who arrived after the 12 workers already waiting in the queue to continue waiting if they did not choose to take the stairs. These conditions and all the other conditions were satisfied in the simulation program. A detailed report, summary graphs, results interpretations, implementation details, and critical thinking summaries were all included in this report to satisfy the remaining requirements of the Elevator Final Project.

Implementation Summary

Language/Tools/Platform Used

This project simulation was written in the Python programming language. Python was chosen over more verbose alternatives such as C# and C++ because of its simplicity, ease of use, and wealth of useful libraries for plotting and for manipulating data. Whereas a program in Python might take 300 lines of code to carry out a successful simulation, C++ would likely take over twice as many lines due to the nature of verbosity of the two languages. Python also allows for easy imports of a vast array of easy-to-use libraries that can simply be downloaded from the command line using the PIP installer. I was able to download Seaborn, Pandas and Matplotlib in under a minute, and all the libraries allowed me to create sleek and useful graphs and manipulate large quantities of data very quickly. Libraries in C++ and other languages are more bulky and would take more time to learn. Python is also an excellent quick prototyping language, allowing me to swiftly program this simulation much faster than I otherwise could have because of all the abstraction the interpreted language of Python allows for.
The chosen platform to develop on was my Lenovo Yoga P40 laptop out of convenience. It would have been much more convenient to use a more powerful computer which contained a more powerful processor and more RAM, but all I had during this development was my laptop, which, unfortunately, made running the thousands of simulations ran throughout this project take much longer than it would have otherwise. An alternative such as Amazon Web Services or Microsoft Azure could have been used and likely would have been faster, but those alternatives would have cost money and/or the set-up time on those platforms would have been prohibitive in themselves.

The main tools used were Matplotlib and Seaborn.These are both very powerful and easy-to-use graphing and visualization tools with excellent documentation. They are also very compatible and easy-to-use with the Python programming language. Seaborn was chosen in addition to Matplotlib for its excellence in displaying density curves and excellent histograms. Other alternatives MATLAB and Tableau for visualizing data would either have taken too long to develop with or to learn.

Data Structures Used

Python actually has very few native data structures in the language [3]. This is because Python's data structures are so versatile. That is why most structures used in this program are lists: these lists can function as C arrays, queues, stacks, dictionaries, and much more because of the many native functions associated with each array.

The core data structure used in this simulation, however, is a self-declared class called Person. The entirety of the Person class data structure can be found below:

```
class Person:
    def __init__(self, arrival_time):
        self.floor = 1
        # random.choice uniformly selects one of the choices, which is fitting since each floor
        # to fo to had a 33% chance of being selected
        self.floor_to_go_to = random.choice([2, 3, 4])

        # This is the time the person had to wait to board and the time it takes to leave the elevator
        self.wait_time = 0

        # arrival_time is the time the person initially arrives at the ground elevator
        self.arrival_time = 0

        # departure_time is the time the person gets off the elevator
        self.departure_time = 0
        self.set_arrival_time(arrival_time)

        # Identify each person in order of who got off first
        self.id = 0

        self.time_boarded_elevator = 0

        self.attempted_to_walk = False

        self.walked = False

    def set_walked_true(self):
        self.walked = True

    def get_walked(self):
        return self.walked

    def set_attempted_to_walk_true(self):
        self.attempted_to_walk = True

    # Sets the arrival time of the person to the ground floor elevator (either line or no line)
    def set_arrival_time(self, arrival_time):
        self.arrival_time = arrival_time
```

```
    # Sets departure time of the person from the elevator (and also wait time since wait = departure - arrival)
    def set_departure_time(self, departure_time):
        self.departure_time = departure_time
        self.wait_time = self.departure_time - self.arrival_time

    def get_time_boarded_elevator(self):
        return self.time_boarded_elevator

    def set_boarding_time(self, boarding_time):
        self.time_boarded_elevator = boarding_time

    def get_floor(self):
        return self.floor_to_go_to

    def set_id(self, id):
        self.id = id

    def get_wait_time(self):
        return self.wait_time

    def __str__(self):
        return "[Person #" + str(self.id) + ", Arrival Time: " + str(self.arrival_time) + ", Departure Time: "
+ str(self.departure_time) + ", Wait Time: " + str(self.wait_time) + ", Time Boarded Lift: " +
str(self.time_boarded_elevator) + ", Walker: " + str(self.walked) + "]\n"


    def __repr__(self):
        return "[Person #" + str(self.id) + ", Arrival Time: " + str(self.arrival_time) + ", Departure Time: "
+ str(self.departure_time) + ", Wait Time: " + str(self.wait_time) + "]"
```

This Person structure contained all of the essential data necessary to successfully run this simulation. It contained floats of the departure, arrival, and wait times, which were all used to help curate the data that would be later visualized. Each person had a `floor_to_go_to` integer which determined the floor that the Person would need to go to when they arrived in the building. This integer was uniformly chosen from an array list of `[2, 3, 4]`, which ensured that, as specified, there was a 33% chance that each person would try to go to each floor.

Each person had a unique ID integer called `id` which was assigned in order of the person's actions, regardless of if the person walked or took the elevator. The time that each person boarded the elevator was also recorded here in the `time_boarded_elevator` float so that the last time that someone boarded the elevator could be recorded later. Other boolean variables, such as `walked` and `attempted_to_walk` are necessary for the logic of the simulation in ensuring that, respectively, someone does not try to board the elevator more than once and that someone does not try to decide to walk more than once.

This person class/structure was the essential part of the whole simulation, for a large list array of Persons were created, and each single Person held all the essential data which could be referenced and set by each Person's getters and setters. Instead of storing each data point of the times they departed, waiting, floors to go to, departure times, and so on, it was much easier to store all of the necessary variables inside one large structure to keep track of all the data for me.

The **event list**, which kept track of all the future times that events will occur, was simply kept in a large list array for its ease of use. This was not efficient data-wise, but the number of times kept in the array rarely exceeds more than 420, and 420 floats in an array is not too data-heavy. The convenience of efficiency timewise made the trade-off worth it.

The **event generator** was simply a random generator structure native to Python's Random import module. It generated random uniform numbers capable of creating a random list of event times in the array list structure above.

The variable `elevator_occupants` was also a simple list with a capacity of 12, since the order of the workers didn't really matter; the workers only got off the elevator if the elevator was at their floor.

The variable `queue` was unsurprisingly a list that acted as a queue since it always operated as a first-in-first-out (FIFO) manner. The specifications of the project effectively and literally described the waiting line outside the elevator as a queue, and that was the necessary data structure to use for this purpose.

Deciding on the number of samples to take

First, a sample population was created of 100 samples of each run for the four categories:

- Average wait times for a worker
- Average number of people who walk to 2nd, 3rd, and 4th floor each day
- The times that the last person boards the elevator each day
- The average numbers of workers in the queue at 8:30, 8:45, and 9:00

The methodology on the number of simulations to run are detailed for each of the four categories below:

Average Wait Times:
Number of simulations run: **320**
After running the average wait times for a worker and placing 100 samples in a file, I then calculated the standard deviation using this standard deviation calculator found at calculator.net [1]. The samples looked like:

[37.69784, 32.55458, 32.70936, 38.18702, 32.32497, 36.67784, 30.98008, 35.06011, 37.59831, 37.99863, 34.85578, 35.56183, 35.33353, 35.03031, 36.69987, 38.3433, 31.6302, 33.55805, 37.68226, 41.01992, 28.04954, 34.16085, 34.33275, 41.02225, 34.95344, 37.68081, 30.55433, 33.94046, 35.35801, 36.88835, 30.78619, 33.25131, 33.41697, 39.81182, 33.2769, 41.71673, 36.30373, 31.44374, 39.57568, 34.17377, 30.74996, 32.97349, 30.55312, 31.45556, 39.85002, 34.23515, 34.32149, 31.44043, 30.71019, 32.53993, 31.37469, 37.12973, 33.63051, 33.54653, 39.63797, 35.13612, 40.39652, 37.45827, 34.1835, 30.04749, 34.17886, 38.08287, 33.72079, 39.5865, 27.79078, 38.28525, 37.72699, 28.44639, 26.29424, 36.54732, 38.15131, 34.24733,

39.32666, 36.58709, 36.0721, 36.0165, 35.52537, 32.98934, 32.01651, 41.28739, 32.31866, 34.32411, 34.08506, 37.11857, 33.66496, 31.22872, 29.38683, 31.8012, 31.17452, 33.0534, 32.68437, 33.49614, 30.60152, 28.69215, 32.75517, 33.06797, 30.38313, 36.65658, 36.75022, 35.48525]

and the standard deviation was found to be 3.30529. Then, using the chart below and being conservative and using a σ/4 as that's the closest standard deviation to the chart and using Tchebycheff's Theorem, it was found that **320** samples would be required in order to confidently state that the average falls between the true mean +/- *d*, the error, at a confidence level of α = 0.5, or a 99% accuracy.

Difference d	n Central Limit	n Tchebycheff's Theorem
$\sigma/2$	15	80
$\sigma/4$	61	320
$\sigma/6$	138	720
$\sigma/8$	246	1280
$\sigma/10$	384	2000
$\sigma/12$	553	2880
$\sigma/20$	1537	8000

After running all **320** iterations of the simulation, the results were output into an array which can be seen below so that a distribution of the average wait times could then be found.

[37.69784, 32.55458, 32.70936, 38.18702, 32.32497, 36.67784, 30.98008, 35.06011, 37.59831, 37.99863, 34.85578, 35.56183, 35.33353, 35.03031, 36.69987, 38.3433, 31.6302, 33.55805, 37.68226, 41.01992, 28.04954, 34.16085, 34.33275, 41.02225, 34.95344, 37.68081, 30.55433, 33.94046, 35.35801, 36.88835, 30.78619, 33.25131, 33.41697, 39.81182, 33.2769, 41.71673, 36.30373, 31.44374, 39.57568, 34.17377, 30.74996, 32.97349, 30.55312, 31.45556, 39.85002, 34.23515, 34.32149, 31.44043, 30.71019, 32.53993, 31.37469, 37.12973, 33.63051, 33.54653, 39.63797, 35.13612, 40.39652, 37.45827, 34.1835, 30.04749, 34.17886, 38.08287, 33.72079, 39.5865, 27.79078, 38.28525, 37.72699, 28.44639, 26.29424, 36.54732, 38.15131, 34.24733, 39.32666, 36.58709, 36.0721, 36.0165, 35.52537, 32.98934, 32.01651, 41.28739, 32.31866, 34.32411, 34.08506, 37.11857, 33.66496, 31.22872, 29.38683, 31.8012, 31.17452, 33.0534, 32.68437, 33.49614, 30.60152, 28.69215, 32.75517, 33.06797, 30.38313, 36.65658, 36.75022, 35.48525, 40.89122, 32.20408, 38.81221, 35.6545, 31.66875, 32.04669, 35.98412, 35.69371, 35.82078, 34.70716, 35.25641, 30.86362, 32.82198, 34.68656, 38.94752, 36.91502, 37.63518, 30.87216, 36.26022, 30.55183, 29.85325, 37.27147, 38.65306, 31.81061, 37.98331, 32.69509, 25.5561, 32.05256, 33.51682, 38.75918, 39.17035, 33.26124, 36.47547, 33.11978, 31.4456, 36.28063, 38.54513, 30.03664, 39.4103, 36.78709, 34.20809, 34.84916, 36.80889, 38.7479, 34.52008, 31.36796, 37.24959, 37.40714, 36.84317, 36.27482, 38.12926, 34.46907, 39.62099, 40.32065, 36.81044, 27.8274, 29.23096, 40.07716, 38.23005, 38.84036, 39.26085, 36.01429, 30.54256, 36.14818, 36.87762, 35.12435, 37.212, 36.30107, 34.75462, 30.81757, 32.25398, 37.22452, 38.10595, 34.12186, 37.14772, 36.93653, 40.04509, 34.44492, 36.18636, 32.71812, 36.84178, 40.61598, 38.77731, 38.5483, 33.19771, 33.71686, 36.48339, 30.97285, 32.67994, 30.96189, 41.72056, 34.43247, 39.28281, 33.55376, 38.43496, 32.35117, 38.79696, 34.29055, 32.62633, 33.26932, 32.13996, 40.33471, 34.99872, 36.89889, 35.40789, 35.73184, 37.04301, 37.01385, 40.56488, 30.65669, 31.09672, 37.1861, 32.2259, 32.09812, 36.93167, 38.99909, 37.23839, 39.30417, 32.13749, 32.9279, 35.89427, 34.03916, 35.29793, 38.08172, 37.9934, 33.49494, 30.34859, 34.96082, 32.42927, 37.6059, 35.50058, 33.72124, 40.69547, 29.46907, 32.69377, 38.90541, 36.10786, 41.68938, 31.62526, 36.86235, 33.92978, 29.30247, 31.57245, 42.08096, 36.88378, 35.09191, 36.78924, 37.11551, 29.86197, 35.17959, 40.54262, 34.884, 36.93443, 36.1238, 37.7952, 39.03348, 43.20849, 31.86124, 39.36549, 34.50207, 34.27875, 31.61318, 29.7386, 37.72204, 39.94726, 31.74289, 33.08104, 35.44814, 36.12312, 31.0392, 36.46662, 35.62474, 32.3224, 33.41869, 36.16184, 41.80312, 37.56781, 41.19712, 36.34142, 36.04591, 33.15978, 33.33856, 39.60246, 37.1032, 34.75124, 42.18724, 36.24174, 28.48527, 29.46314, 35.87113, 35.69562, 35.01932, 36.84336, 31.53006, 32.00403, 27.25476, 36.49905, 37.66352, 33.66475, 35.99109, 36.59605, 34.70702, 39.69591, 28.46675, 35.66402, 35.70068, 34.91797, 36.56838, 32.91488, 34.40905, 32.77355, 32.95283, 37.38664, 35.48522, 30.68582, 42.92839, 36.82212, 34.47093, 33.32292, 33.27738]

Average Number of People that walk to 2nd, 3rd, and 4th Floors:
Number of simulations run: **1280**
Following the pattern to discover the average number of wait times, the same process was run on finding the standard deviation of 100 samples of data for each of the floors. The standard deviation for each was found to be:

Standard Deviations of each floor data:
Second Floor: **6.213**
Third Floor: **5.0825**
Fourth Floor: **2.659**

Using the same chart described above for reference of the Tchebycheff's Theorem -being conservative- it was seen that, taking account for the worst standard deviation of the group, a

difference of σ/14 should be run to ensure a 99% accuracy, or at least 1280 simulations. Since I would like to graph all the floors next to each other, I went ahead and ran the gathered data from 1280 simulations for each population even though the fourth floor could have sufficed with just 320 samples.

The data found is below:

Second Floor:

[36, 35, 38, 34, 48, 25, 31, 28, 51, 28, 32, 33, 45, 28, 38, 41, 34, 21, 46, 43, 29, 36, 39, 39, 38, 34, 39, 42, 42, 36, 43, 33, 31, 48, 28, 45, 36, 30, 43, 38, 40, 45, 31, 37, 36, 33, 32, 32, 29, 42, 25, 42, 45, 23, 36, 40, 30, 42, 35, 46, 35, 34, 34, 42, 26, 34, 35, 29, 36, 30, 35, 35, 32, 37, 25, 36, 38, 46, 47, 29, 25, 38, 29, 35, 36, 39, 43, 27, 39, 29, 34, 35, 33, 33, 36, 36, 34, 27, 33, 39, 29, 45, 49, 36, 28, 44, 45, 31, 38, 33, 34, 31, 48, 44, 33, 33, 32, 35, 27, 35, 39, 33, 35, 40, 32, 43, 40, 36, 29, 36, 34, 35, 37, 35, 44, 36, 29, 26, 31, 34, 38, 38, 44, 32, 27, 42, 32, 39, 31, 27, 39, 36, 37, 34, 30, 17, 40, 38, 40, 32, 39, 40, 30, 33, 29, 43, 51, 38, 28, 39, 52, 37, 35, 45, 25, 36, 21, 28, 37, 39, 41, 50, 34, 24, 36, 32, 30, 35, 34, 18, 33, 32, 37, 40, 50, 37, 37, 29, 38, 28, 30, 43, 39, 38, 31, 54, 34, 35, 33, 39, 32, 44, 34, 35, 41, 46, 29, 41, 39, 34, 42, 37, 34, 59, 43, 42, 26, 47, 29, 30, 23, 32, 33, 47, 36, 39, 46, 33, 25, 38, 36, 30, 26, 37, 40, 24, 34, 32, 43, 39, 38, 26, 33, 35, 44, 23, 36, 38, 32, 36, 27, 33, 44, 37, 47, 41, 35, 34, 41, 46, 29, 44, 38, 49, 33, 26, 34, 29, 44, 27, 32, 33, 32, 25, 35, 55, 39, 34, 33, 33, 41, 41, 41, 22, 54, 37, 26, 35, 33, 36, 42, 30, 30, 35, 42, 31, 40, 31, 42, 34, 50, 44, 32, 39, 34, 38, 52, 26, 32, 41, 40, 29, 34, 36, 34, 36, 44, 46, 48, 40, 29, 31, 44, 38, 27, 32, 38, 42, 36, 32, 30, 40, 36, 45, 37, 36, 36, 32, 26, 29, 42, 30, 31, 39, 33, 32, 30, 44, 43, 45, 41, 34, 38, 33, 33, 39, 30, 33, 42, 34, 45, 33, 43, 40, 33, 40, 44, 40, 34, 38, 37, 45, 36, 26, 40, 34, 44, 45, 32, 37, 41, 37, 51, 29, 60, 25, 33, 32, 29, 52, 37, 36, 31, 36, 41, 34, 32, 28, 49, 43, 37, 37, 37, 32, 35, 39, 29, 39, 48, 40, 32, 36, 40, 37, 36, 32, 41, 35, 33, 27, 38, 33, 44, 41, 23, 36, 35, 33, 37, 26, 31, 52, 30, 35, 38, 33, 36, 40, 37, 34, 42, 37, 41, 34, 39, 32, 43, 23, 38, 37, 40, 31, 43, 34, 28, 44, 31, 29, 35, 29, 24, 42, 46, 31, 44, 41, 40, 29, 39, 33, 38, 28, 50, 38, 48, 41, 35, 30, 26, 34, 38, 45, 47, 30, 37, 43, 36, 30, 44, 35, 35, 25, 26, 37, 36, 35, 33, 30, 27, 30, 37, 34, 44, 30, 35, 47, 31, 38, 39, 33, 49, 33, 40, 28, 40, 36, 49, 41, 42, 28, 39, 28, 35, 27, 37, 38, 39, 27, 23, 35, 28, 29, 40, 40, 41, 49, 31, 33, 43, 33, 32, 40, 47, 38, 33, 36, 39, 31, 37, 32, 38, 37, 57, 34, 37, 30, 28, 23, 36, 30, 33, 28, 29, 36, 31, 23, 33, 37, 39, 41, 33, 36, 27, 29, 36, 27, 35, 38, 46, 37, 39, 35, 34, 32, 36, 41, 27, 29, 31, 30, 29, 32, 49, 28, 41, 33, 23, 22, 38, 33, 29, 37, 39, 42, 36, 37, 28, 39, 28, 34, 31, 35, 41, 29, 30, 25, 39, 44, 37, 41, 20, 33, 32, 43, 47, 37, 45, 29, 36, 34, 30, 41, 39, 34, 44, 41, 30, 43, 33, 50, 30, 37, 25, 33, 37, 30, 31, 52, 31, 32, 31, 37, 31, 37, 43, 35, 36, 40, 34, 28, 34, 34, 38, 32, 46, 46, 32, 27, 30, 37, 34, 36, 39, 34, 34, 32, 31, 30, 20, 37, 40, 27, 36, 48, 34, 42, 40, 34, 34, 33, 29, 35, 38, 43, 31, 50, 44, 32, 33, 27, 33, 32, 44, 28, 24, 39, 23, 41, 30, 52, 42, 46, 42, 43, 46, 46, 26, 38, 27, 46, 40, 41, 36, 36, 34, 38, 32, 41, 42, 35, 40, 37, 40, 33, 30, 43, 31, 36, 28, 47, 24, 35, 41, 45, 39, 35, 34, 32, 37, 42, 38, 41, 41, 29, 28, 33, 47, 28, 39, 28, 42, 42, 36, 42, 40, 31, 34, 38, 34, 26, 33, 37, 32, 40, 32, 33, 37, 35, 36, 36, 39, 41, 29, 41, 38, 25, 45, 31, 33, 42, 30, 24, 43, 25, 29, 38, 30, 44, 36, 38, 37, 38, 31, 43, 42, 35, 23, 42, 24, 32, 33, 43, 35, 40, 27, 34, 38, 41, 40, 35, 52, 33, 45, 30, 25, 36, 38, 41, 36, 29, 29, 38, 28, 37, 39, 30, 33, 32, 42, 33, 29, 27, 46, 33, 41, 32, 36, 41, 32, 30, 27, 30, 34, 43, 32, 40, 33, 49, 34, 32, 34, 32, 46, 37, 33, 41, 29, 41, 45, 43, 42, 26, 42, 32, 35, 40, 37, 43, 33, 30, 29, 46, 36, 36, 35, 39, 32, 37, 19, 33, 38, 35, 57, 37, 35, 40, 28, 25, 48, 23, 40, 31, 35, 29, 36, 30, 41, 36, 38, 35, 41, 43, 48, 38, 37, 38, 31, 36, 33, 40, 34, 43, 35, 39, 45, 41, 42, 23, 33, 43, 32, 36, 40, 26, 22, 38, 24, 37, 32, 28, 36, 38, 41, 41, 39, 28, 25, 39, 34, 47, 35, 28, 37, 43, 37, 32, 32, 39, 30, 34, 31, 29, 42, 29, 38, 32, 41, 24, 41, 29, 39, 27, 44, 34, 31, 30, 40, 30, 41, 39, 31, 51, 46, 27, 31, 31, 29, 34, 29, 36, 41, 32, 38, 47, 40, 46, 37, 46, 44, 40, 33, 39, 49, 48, 35, 24, 31, 42, 33, 30, 36, 25, 31, 54, 38, 41, 27, 41, 31, 39, 30, 38, 38, 32, 44, 28, 32, 30, 34, 50, 21, 51, 40, 38, 37, 48, 44, 32, 41, 39, 49, 30, 35, 35, 27, 39, 39, 37, 24, 25, 26, 37, 30, 33, 33, 28, 34, 41, 33, 28, 38, 41, 32, 44, 37, 38, 39, 29, 35, 24, 26, 41, 31, 42, 41, 28, 35, 40, 33, 40, 20, 44, 39, 32, 40, 35, 29, 46, 42, 39, 29, 38, 38, 42, 34, 29, 34, 28, 31, 45, 42, 40, 35, 34, 33, 44, 43, 31, 37, 39, 31, 34, 37, 38, 36, 46, 30, 29, 34, 32, 24, 27, 41, 39, 38, 33, 47, 46, 34, 27, 33, 30, 28, 37, 44, 50, 27, 30, 57, 23, 30, 38, 21, 36, 34, 28, 50, 43, 31, 37, 35, 37, 44, 28, 41, 30, 48, 24, 42, 37, 26, 28, 41, 31, 33, 33, 44, 29, 43, 32, 41, 42, 37, 37, 33, 38, 40, 40, 48, 34, 32, 26, 39, 38, 40, 38, 34, 41, 42, 35, 32, 56, 38, 31, 29, 31, 42, 40, 30, 35, 36, 37, 42, 39, 43, 32, 34, 32, 36, 41, 41, 39, 36, 24, 38, 35, 18, 30, 39, 29, 24, 41, 21, 35, 36, 36, 44, 47, 39, 33, 27, 37, 31, 37, 34, 34, 43, 36, 46, 42, 34, 39, 45, 36, 29, 30, 37, 31, 28, 37, 38, 23, 38, 37, 42, 43, 38, 42, 41, 32, 29, 34, 35, 30, 27, 32, 33, 29, 30, 29, 48]

Third Floor:

[16, 23, 31, 20, 34, 24, 16, 27, 32, 25, 30, 21, 28, 26, 26, 22, 17, 27, 23, 23, 14, 28, 15, 23, 29, 24, 22, 29, 20, 23, 27, 26, 20, 14, 18, 29, 26, 25, 24, 20, 23, 29, 35, 27, 25, 24, 23, 20, 25, 22, 27, 18, 18, 23, 25, 24, 28, 29, 16, 29, 34, 18, 28, 24, 25, 23, 24, 30, 33, 32, 13, 19, 19, 28, 21, 17, 17, 34, 31, 31, 22, 24, 16, 27, 21, 26, 24, 19, 28, 22, 24, 16, 32, 25, 17, 25, 26, 24, 22, 30, 25, 31, 22, 22, 19, 26, 17, 23, 41, 27, 20, 28, 19, 19, 28, 39, 22, 29, 21, 22, 21, 34, 31, 29, 21, 26, 18, 21, 29, 30, 21, 19, 34, 29, 31, 20, 28, 22, 18, 25, 15, 23, 21, 25, 23, 22, 23, 26, 18, 24, 23, 23, 20, 29, 17, 28, 37, 21, 24, 25, 30, 24, 24, 23, 17, 30, 22, 27, 27, 19, 36, 27, 25, 29, 22, 25, 23, 21, 20, 24, 25, 25, 28, 34, 21, 24, 20, 22, 30, 30, 31, 29, 17, 25, 19, 23, 31, 21, 26, 24, 20, 24, 25, 13, 26, 28, 17, 25, 21, 30, 19, 34, 27, 29, 27, 21, 28, 20, 25, 28, 30, 25, 22, 24, 31, 24, 20, 21, 22, 25, 13, 22, 17, 22, 29, 22, 26, 23, 25, 26, 17, 22, 22, 17, 22, 21, 20, 20, 32, 30, 22, 27, 22, 16, 24, 29, 25, 25, 27, 28, 23, 20, 19, 31, 34, 27, 26, 21, 25, 36, 23, 24, 26, 17, 24, 18, 18, 23, 21, 19, 29, 28, 23, 27, 23, 24, 26, 34, 33, 25, 33, 26, 32, 18, 13, 23, 19, 32, 28, 17, 23, 34, 26, 30, 11, 27, 24, 30, 22, 25, 22, 25, 30, 24, 24, 21, 28, 31, 18, 21, 33, 16, 19, 21, 32, 31, 21, 23, 35, 18, 18, 25, 26, 31, 19, 30, 21, 28, 21, 28, 22, 26, 25, 31, 28, 22, 25, 17, 21, 20, 21, 25, 18, 33, 24, 19, 27, 23, 22, 27, 22, 19, 24, 22, 26, 32, 30, 30, 35, 17, 24, 22, 22, 25, 12, 25, 29, 32, 24, 30, 20, 24, 18, 28, 21, 21, 21, 19, 20, 24, 24, 33, 21, 24, 26, 12, 27, 16, 19, 17, 15, 23, 26, 32, 27, 25, 25, 20, 28, 18, 21, 17, 16, 24, 25, 23, 17, 24, 26, 23, 20, 24, 22, 27, 39, 25, 24, 26, 35, 29, 18, 19, 30, 26, 15, 26, 24, 18, 28, 28, 30, 39, 23, 24, 29, 28, 23, 24, 22, 21, 19, 32, 23, 28, 13, 26, 20, 20, 15, 17, 25, 19, 26, 24, 28, 21, 25, 20, 20, 18, 23, 22, 29, 23, 20, 25, 27, 31, 27, 25, 20, 15, 20, 27, 34, 24, 28, 19, 26, 30, 28, 24, 19, 24, 20, 19, 31, 28, 29, 23, 24, 28, 20, 20, 19, 20, 17, 15, 24, 27, 30, 23, 26, 20, 22, 39, 23, 19, 17, 18, 35, 21, 30, 34, 29, 20, 28, 23, 18, 29, 24, 22, 16, 21, 29, 21, 23, 14, 21, 20, 14, 23, 23, 30, 13, 28, 22, 32, 27, 18, 23, 18, 24, 25, 25, 27, 24, 22, 30, 22, 22, 24, 15, 13, 22, 26, 26, 20, 20, 16, 19, 24, 20, 22, 20, 27, 32, 16, 24, 22, 20, 28, 20, 26, 22, 22, 26, 15, 30, 20, 19, 23, 14, 30, 30, 26, 29, 37, 27, 19, 22, 25, 22, 23, 16, 29, 21, 26, 25, 21, 20, 19, 16, 18, 23, 33, 21, 16, 20, 31, 31, 21, 25, 24, 21, 19, 28, 27, 21, 23, 20, 13, 26, 18, 19, 26, 31, 19, 23, 24, 20, 26, 22, 24, 33, 21, 28, 24, 13, 26, 24, 19, 21, 23, 20, 32, 16, 25, 26, 21, 28, 26, 22, 18, 28, 25, 20, 27, 19, 19, 28, 21, 26, 17, 33, 28, 23, 16, 27, 28, 21, 19, 24, 28, 26, 25, 26, 18, 23, 27, 26, 21, 26, 25, 29, 32, 22, 26, 16, 26, 22, 12, 28, 19, 27, 27, 25, 21, 33, 15, 23, 26, 19, 29, 26, 30, 23, 27, 16, 23, 22, 26, 25, 17, 17, 24, 22, 23, 8, 18, 22, 22, 27, 22, 26, 31, 22, 27, 23, 30, 24, 35, 29, 28, 24, 17, 28, 21, 20, 25, 18, 30, 19, 16, 19, 16, 29, 26, 18, 19, 17, 27, 28, 28, 28, 24, 23, 21, 35, 12, 26, 22, 30, 27, 22, 22, 30, 29, 32, 20, 32, 27, 37, 23, 26, 24, 25, 28, 24, 26, 20, 21, 22, 25, 21, 21, 21, 25, 28, 21, 19, 22, 17, 26, 27, 32, 24, 29, 23, 21, 28, 20, 24, 25, 20, 22, 28, 22, 17, 23, 24, 19, 28, 27, 19, 18, 24, 17, 21, 25, 34, 22, 30, 18, 20, 20, 31, 15, 24, 21, 15, 27, 19, 20, 24, 26, 24, 23, 21, 26, 25, 29, 30, 23, 25, 21, 22, 25, 19, 25, 20, 27, 19, 27, 21, 22, 25, 23, 26, 37, 22, 26, 28, 29, 20, 27, 23, 32, 27, 28, 27, 24, 17, 21, 31, 20, 24, 38, 17, 26, 31, 21, 26, 22, 28, 15, 29, 23, 17, 22, 28, 21, 33, 20, 16, 20, 28, 30, 27, 18, 25, 25, 25, 30, 30, 25, 27, 22, 16, 20, 25, 17, 27, 32, 22, 22, 26, 24, 21, 24, 24, 26, 23, 29, 27, 21, 27, 27, 26, 27, 15, 34, 26, 22, 26, 20, 24, 28, 30, 16, 31, 28, 28, 23, 30, 27, 32, 15, 20, 32, 20, 23, 30, 34, 25, 17, 23, 25, 25, 24, 26, 16, 24, 21, 25, 18, 22, 20, 22, 30, 20, 26, 26, 27, 28, 21, 16, 21, 31, 22, 26, 18, 26, 20, 22, 22, 19, 17, 28, 16, 26, 20, 32, 26, 21, 24, 19, 17, 18, 15, 28, 22, 30, 19, 23, 22, 37, 28, 20, 15, 26, 24, 26, 25, 18, 22, 27, 14, 16, 28, 21, 29, 18, 22, 21, 15, 19, 30, 28, 25, 27, 21, 25, 16, 24, 25, 25, 18, 28, 25, 24, 24, 13, 26, 18, 22, 20, 20, 16, 22, 24, 22, 17, 26, 21, 16, 21, 18, 24, 35, 25, 26, 28, 24, 23, 31, 20, 25, 27, 21, 22, 22, 21, 24, 22, 35, 21, 30, 25, 24, 26, 18, 23, 17, 18, 21, 24, 21, 26, 23, 23, 28, 21, 19, 17, 24, 17, 26, 31, 36, 25, 27, 24, 21, 27, 28, 26, 21, 24, 19, 30, 15, 25, 26, 27, 28, 28, 27, 21, 17, 22, 24, 25, 21, 23, 30, 30, 17, 26, 18, 22, 28, 28, 33, 25, 25, 29, 19, 31, 25, 23, 21, 20, 28, 16, 26, 23, 19, 27, 23, 15, 30, 21, 20, 36, 12, 33, 24, 20, 24, 23, 15, 11, 21, 17, 21, 31, 32, 36, 20, 36, 32, 35, 29, 27, 23, 33, 28, 24, 26, 26, 27, 30, 30, 22, 24, 24, 18, 28, 16, 23, 28, 27, 20, 14, 27, 23, 28, 18, 15, 21, 17, 20, 28, 26, 26, 21, 26, 21, 35, 23, 17, 19, 26, 20, 16, 18, 17, 25, 21, 35, 11, 20, 29, 26, 23, 22, 19, 24, 28, 22, 18, 24, 29, 33, 29, 32, 17, 18, 16, 22, 22, 31, 23, 23, 31, 25, 15, 29, 32, 30, 17, 29, 28, 25, 25, 25, 28, 29, 29, 23, 16, 17, 18, 22, 20, 20, 30, 28, 20, 23]

Fourth Floor:

[10, 13, 9, 11, 4, 10, 8, 2, 4, 9, 8, 6, 7, 4, 3, 6, 6, 4, 4, 5, 8, 8, 5, 6, 8, 7, 12, 11, 9, 12, 9, 3, 8, 3, 8, 7, 8, 6, 10, 7, 9, 7, 5, 5, 6, 7, 6, 9, 6, 5, 8, 10, 8, 6, 12, 6, 5, 8, 12, 8, 5, 7, 4, 10, 3, 8, 11, 5, 7, 6, 3, 12, 5, 8, 8, 8, 6, 10, 5, 3, 5, 5, 8, 6, 10, 8, 9, 6, 10, 9, 3, 7, 11, 7, 3, 9, 3, 4, 14, 10, 7, 3, 9, 8, 7, 4, 4, 6, 9, 11, 8, 11, 9, 8, 6, 7, 6, 7, 8, 9, 5, 8, 6, 8, 11, 9, 9, 10, 12, 9, 3, 7, 3, 7, 12, 2, 10, 9, 13, 10, 7, 9, 6, 11, 3, 9, 12, 9, 6, 8, 6, 7, 4, 4, 3, 9, 7, 9, 7, 7, 7, 7, 7, 9, 6, 8, 5, 8, 5, 9, 13, 5, 5, 14, 7, 11, 6, 11, 6, 3, 9, 7, 7, 5, 7, 9, 7, 4, 4, 10, 10, 4, 8, 6, 9, 7, 7, 5, 12, 7, 9, 4, 11, 8, 10, 5, 5, 4, 6, 9, 9, 7, 2, 8, 6, 4, 8, 4, 7, 11, 6, 6, 6, 6, 2, 9, 5, 6, 5, 14, 8, 9, 9, 4, 7, 4, 6, 5, 8, 5, 10, 2, 8, 4, 4, 5, 4, 6, 8, 7, 6, 6, 9, 6, 5, 8, 7, 10, 9, 5, 7, 10, 6, 8, 5, 5, 4, 14, 2, 8, 7, 10, 7, 14, 7, 4, 5, 8, 8, 9, 10, 6, 9, 5, 6, 8, 7, 6, 8, 10, 6, 4, 4, 7, 5, 5, 6, 4, 12, 6, 5, 3, 3, 12, 8, 9, 10, 11, 10, 7, 8, 4, 5, 7, 10, 5, 5, 5, 9, 11, 9, 8, 8, 14, 2, 7, 11, 10, 7, 4, 6, 8, 8, 7, 3, 9, 5, 7, 4, 6, 10, 11, 7, 6, 4, 6, 4, 4, 7, 8, 9, 10, 6, 9, 7, 7, 8, 9, 4, 6, 8, 7, 9, 7, 16, 7, 5, 4, 6, 7, 11, 5, 12, 8, 4, 2, 7, 6, 7, 9, 4, 6, 7, 9, 4, 8, 5, 8, 3, 12, 9, 8, 7, 5, 9, 7, 9, 7, 5, 6, 5, 13, 6, 8, 5, 5, 7, 4, 10, 10, 8, 7, 9, 7, 3, 3, 4, 5, 5, 9, 4, 5, 9, 10, 9, 10, 9, 7, 5, 5, 8, 8, 12, 8, 3, 4, 8, 10, 9, 9, 7, 5, 2, 8, 6, 9, 5, 7, 14, 5, 6, 9, 11, 8, 10, 6, 6, 9, 9, 2, 4, 5, 8, 11, 8, 11, 11, 5, 4, 8, 10, 9, 3, 8, 3, 7, 5, 11, 10, 3, 11, 10, 5, 10, 4, 7, 4, 4, 18, 9, 6, 9, 10, 5, 9, 11, 4, 14, 6, 9, 7, 7, 8, 6, 7, 2, 10, 9, 7, 7, 7, 10, 9, 5, 14, 12, 8, 9, 10, 11, 5, 11, 5, 8, 4, 9, 6, 6, 9, 9, 7, 6, 6, 4, 9, 4, 8, 11, 5, 9, 8, 13, 8, 5, 9, 8, 7, 8, 8, 7, 5, 3, 4, 6, 3, 7, 10, 11, 12, 12, 4, 7, 12, 5, 7, 11, 7, 4, 8, 11, 9, 9, 4, 6, 4, 11, 9, 7, 2, 10, 7, 4, 9, 4, 7, 2, 7, 9, 6, 8, 4, 8, 5, 7, 6, 4, 11, 6, 8, 9, 6, 6, 5, 4, 7, 8, 7, 12, 9, 8, 5, 6, 8, 6, 6, 6, 9, 5, 10, 5, 5, 6, 10, 7, 4, 5, 9, 8, 9, 14, 4, 3, 7, 4, 7, 4, 12, 4, 3, 7, 3, 7, 5, 5, 3, 4, 5, 4, 5, 4, 9, 9, 8, 7, 9, 5, 9, 7, 5, 10, 14, 5, 4, 4, 5, 14, 4, 7, 11, 9, 11, 8, 4, 7, 10, 5, 8, 7, 6, 9, 5, 6, 9, 5, 5, 7, 5, 11, 3, 5, 8, 7, 10, 6, 9, 8, 5, 6, 3, 7, 9, 3, 7, 10, 6, 7, 3, 13, 8, 9, 4, 6, 7, 3, 8, 7, 7, 7, 8, 6, 2, 5, 5, 7, 7, 11, 7, 10, 7, 8, 6, 11, 7, 6, 2, 4, 3, 5, 11, 8, 9, 6, 8, 4, 8, 8, 7, 8, 4, 4, 5, 12, 5, 6, 9, 11, 7, 9, 4, 10, 6, 12, 5, 7, 4, 3, 6, 8, 3, 6, 9, 8, 7, 7, 5, 8, 5, 7, 7, 10, 6, 8, 9, 6, 10, 6, 7, 8, 8, 10, 6, 6, 4, 9, 5, 9, 6, 6, 11, 4, 6, 2, 7, 7, 5, 3, 9, 7, 5, 2, 4, 10, 13, 7, 9, 10, 14, 7, 10, 7, 9, 6, 6, 10, 4, 6, 14, 9, 7, 6, 7, 12, 4, 8, 4, 10, 3, 1, 11, 6, 13, 5, 8, 8, 6, 11, 5, 5, 9, 16, 11, 8, 4, 14, 6, 11, 4, 9, 8, 9, 9, 10, 2, 4, 8, 3, 9, 3, 7, 6, 4, 8, 7, 6, 6, 4, 11, 10, 10, 9, 10, 8, 1, 9, 10, 5, 4, 5, 5, 3, 14, 4, 4, 7, 10, 6, 5, 13, 8, 7, 5, 10, 5, 10, 5, 5, 8, 11, 9, 9, 10, 3, 6, 10, 6, 4, 6, 4, 4, 9, 6, 9, 9, 7, 9, 3, 6, 9, 8, 6, 10, 7, 6, 7, 5, 5, 9, 6, 1, 9, 5, 3, 5, 11, 6, 10, 6, 9, 6, 5, 6, 6, 9, 9, 7, 8, 8, 7, 5, 5, 1, 6, 5, 7, 4, 4, 4, 8, 5, 5, 9, 6, 10, 5, 3, 4, 5, 7, 3, 8, 5, 4, 9, 3, 10, 5, 10, 7, 9, 7, 6, 7, 9, 6, 9, 7, 6, 12, 6, 13, 5, 11, 4, 4, 8, 5, 8, 10, 2, 10, 6, 12, 7, 7, 4, 8, 8, 10, 10, 9, 9, 2, 8, 5, 11, 8, 9, 3, 4, 7, 7, 10, 11, 4, 5, 3, 10, 9, 4, 6, 6, 8, 6, 5, 11, 3, 9, 4, 9, 5, 8, 6, 13, 5, 5, 11, 4, 6, 5, 12, 8, 3, 5, 7, 8, 3, 4, 9, 8, 16, 10, 6, 9, 5, 5, 6, 9, 5, 2, 6, 8, 4, 7, 4, 8, 6, 8, 9, 10, 9, 3, 11, 0, 9, 9, 5, 5, 11, 9, 11, 5, 6, 6, 6, 6, 10, 8, 16, 10, 6, 4, 9, 5, 6, 10, 4, 8, 3, 3, 5, 3, 5, 7, 8, 4, 9, 4, 6, 8, 7, 8, 10, 7, 10, 3, 3, 5, 6, 6, 7, 5, 2, 4, 4, 8, 8, 3, 4, 8, 8, 10, 9, 11, 7, 6, 11, 4, 4, 7, 6, 6, 7, 8, 6, 3, 7, 2, 6, 12, 2, 4, 8, 7, 2, 6, 9, 11, 9, 5, 8, 5, 5, 7, 7, 5, 8, 3, 6, 6, 8, 7, 4, 7, 10, 7, 7, 5, 9, 10, 6, 11, 5, 7, 2, 12, 5, 8, 4, 4, 3, 7, 10, 8, 4, 6, 9, 4, 10, 5, 7, 7, 9, 8, 7, 9, 6, 12, 11, 4, 9, 10, 9, 3, 8, 8, 6, 6, 7, 8, 9, 9, 4, 4, 8, 6, 6, 5, 4, 12, 8, 4, 8, 5, 6, 5, 7, 6, 5, 12, 8, 2, 3, 8, 6, 7, 9, 9, 8, 9, 7, 5, 4, 7, 8, 12, 7, 14, 3, 3]

The increasingly smaller numbers make sense as you go from the second floors to the fourth floor, for workers are much more likely to walk to the second floor, which has a 50% probability, than the fourth floor, which only has a 10% probability.

Time of Last Boarding of Elevator:
Number of simulations run: **1280**

Following the above pattern, the same process was run on finding the standard deviation of 100 samples of data for each of the times of the last boarding of the elevator. The standard deviation was found to be **6.05134**.

Using the chart for reference of the Tchebycheff's Theorem -being conservative- it was seen that a difference of σ/8 should be run to ensure a 99% accuracy, or at least 1280 simulations.

The resulting data of the 1280 simulations looks like:

last_board_lift_times = [132.54228, 128.29161, 123.30639, 133.55279, 127.81515, 126.15239, 125.06504, 120.01562, 141.10004, 125.86946, 124.27028, 136.90406, 131.01636, 127.92517, 127.06215, 124.77991, 134.55492, 126.49463, 131.5517, 133.53128, 130.52626, 134.32688, 127.46329, 120.71053, 119.0069, 130.62538, 128.93192, 152.8637, 120.21749, 128.02238, 134.06373, 129.40594, 130.31713, 134.62541, 131.53913, 130.09627, 140.084, 119.88924, 130.09547, 131.12949, 127.56093, 123.57965, 122.04183, 127.90073, 120.56341, 130.25598, 131.02458, 129.01062, 130.7712, 133.12506, 127.58829, 127.75867, 131.04141, 120.26171, 128.29427, 116.49566, 120.52662, 136.08137, 123.40316, 134.4089, 129.44575, 128.29153, 126.54659, 129.26973, 127.85915, 127.16846, 123.28229, 137.50635, 134.05811, 127.53543, 127.0003, 122.75796, 132.96931, 134.04217, 117.98505, 134.024, 124.76491, 141.98539, 126.03929, 124.74277, 131.50802, 128.18338, 132.26133, 123.39765, 133.00733, 130.04546, 128.89301, 123.44763, 133.73922, 130.27032, 129.90225, 133.54365, 117.81143, 123.61008, 120.51662, 131.42467, 112.13671, 117.52848, 124.31347, 133.75133, 119.99358, 123.41418, 130.39878, 115.79752, 128.26512, 139.36701, 130.6615, 138.24978, 122.29207, 127.00264, 129.56874, 116.48072, 137.59444, 126.95273, 121.24142, 131.11818, 129.87113, 129.58762, 127.84133, 127.75215, 124.07297, 124.0335, 128.38408, 131.78042, 131.53411, 124.61924, 139.07853, 132.43409, 132.56446, 123.60887, 129.76368, 134.02263, 127.23443, 135.13792, 137.10221, 126.55052, 124.42386, 127.6758, 136.81876, 117.26821, 119.55381, 126.56195, 129.83805, 136.83413, 135.09972, 123.08911, 130.31177, 125.17366, 124.41264, 127.96056, 122.53333, 128.09441, 141.75816, 131.14629, 124.54649, 130.69666, 119.87349, 130.55906, 131.55232, 133.65364, 131.45396, 124.22762, 134.79829, 134.5836, 137.29683, 129.53594, 124.79488, 127.3298, 120.07358, 141.41614, 130.17165, 130.41498, 130.09533, 127.82501, 137.0347, 135.37872, 116.64619, 127.82368, 123.712, 137.74138, 128.29026, 127.27725, 129.78261, 137.01478, 117.52287, 130.1766, 126.97577, 131.25488, 116.01967, 128.09588, 130.27672, 117.50055, 125.36925, 116.06304, 135.04085, 117.16078, 126.5545, 131.89656, 117.13484, 123.77353, 133.47891, 140.10641, 127.31397, 136.79286, 122.80199, 116.34999, 127.80429, 124.68202, 124.00218, 134.54366, 136.36777, 133.13172, 120.90551, 136.68343, 112.00913, 136.0864, 143.63083, 134.36981, 128.0224, 120.56913, 123.42187, 126.2839, 131.84384, 141.33953, 128.83866, 131.99531, 127.22425, 124.07628, 134.47119, 130.55923, 124.51307, 141.04176, 131.61082, 124.76637, 125.51093, 127.51374, 130.84898, 115.62745, 134.16989, 127.10205, 133.82106, 131.56037, 127.65198, 126.00269, 120.60575, 131.44119, 127.85963, 131.44037, 118.54811, 127.50229, 124.50859, 138.01925, 121.64538, 111.69584, 130.06907, 125.52414, 137.72636, 127.78173, 133.76994, 123.3226, 149.19159, 124.14923, 123.14911, 117.28614, 134.0481, 126.35161, 127.75923, 135.29011, 140.60253, 124.76771, 119.41753, 120.37509, 123.52847, 129.03247, 126.18853, 120.69849, 127.07982, 133.37821, 130.57959, 139.44034, 127.54221, 131.22163, 124.26538, 119.70724, 132.51608, 131.53045, 127.28048, 136.05328, 123.06871, 124.00601, 124.58775, 130.08153, 127.30263, 136.066, 135.27249, 123.63856, 126.56971, 117.94283, 127.77087, 119.14066, 133.60624, 127.51748, 116.6753, 117.05475, 128.26077, 141.00762, 127.07321, 124.77487, 117.37579, 121.55914, 130.90838, 124.52882, 151.9717, 136.76509, 129.81595, 127.26132, 117.66999, 131.7738, 137.2323, 126.48097, 130.60323, 130.02946, 130.86039, 113.5932, 138.37456, 131.38407, 116.86161, 131.09932, 113.38962, 122.66595, 136.09967, 134.78234, 130.57925, 135.49261, 124.102, 130.55396, 129.32602, 119.36287, 126.29478, 127.8847, 130.58138, 120.01022, 136.73242, 125.14933, 127.9449, 122.2544, 124.02081, 132.65601, 134.31777, 120.25735, 130.18781, 126.3676, 133.01199, 141.46017, 145.52076, 140.84367, 127.32473, 112.39025, 134.02467, 116.57342, 134.85477, 119.76948, 127.52408, 120.52481, 126.01339, 144.26561, 120.33648, 134.56377, 132.11593, 131.77987, 123.95917, 131.86751, 134.29418, 133.38606, 121.76308, 130.69388, 123.69008, 131.1568, 127.04531, 127.05855, 115.85172, 127.93888, 120.68938, 133.9521, 122.54464, 127.52753, 137.51769, 128.79681, 135.30435, 130.412, 130.65597, 127.12822, 135.40175, 124.32519, 133.40866, 122.753, 137.38163, 131.34982, 135.75623, 119.88544, 127.08904, 128.13969, 136.96268, 127.6577, 134.92067, 119.52513, 125.04756, 116.52404, 124.4064, 127.74277, 123.28356, 131.8384, 126.40306, 121.84793, 127.46645, 122.914, 127.78879, 124.89897, 126.75908, 120.00967, 141.51021, 145.08583, 127.27798, 130.60044, 126.23051, 123.44418, 134.03107, 134.80867, 121.01534, 130.51468, 133.15912, 134.59267, 116.51225, 127.50338, 134.5081, 124.54732, 127.86206, 137.84611, 113.84772, 131.01375, 121.05881, 132.15011, 126.04386, 126.31455, 124.86425, 140.36682, 117.73588, 132.62136, 124.5066, 133.84961, 128.83795, 153.9199, 129.62739, 120.51546, 123.56806, 124.89648, 123.79107, 127.26165, 136.562, 123.34404, 117.80923, 132.0211, 122.09218, 124.68793, 123.54128, 128.30358, 120.82747, 127.79475, 122.77322, 123.76058, 122.53024, 121.31865, 125.26886, 127.62877, 119.57066, 116.58644, 127.03782, 124.98305, 135.25617, 124.70565, 130.52914, 126.69076, 130.6707, 130.71749, 130.87223, 127.73975, 114.91786, 143.56988, 120.64717, 136.57153, 132.50709, 120.08061, 119.83528, 123.01614, 121.27013, 129.57087, 123.39513, 130.06286, 113.16396, 128.36587, 140.39862, 127.05627, 139.31157, 123.01051, 130.67007, 123.27053, 134.52701, 114.07995, 122.51995, 130.37033, 137.38486, 124.28725, 134.11244, 123.01904, 128.78216, 129.57498, 134.07066, 134.2628, 132.63762, 111.53728, 128.20623, 129.82059, 132.76097, 124.09357, 118.90366, 126.08432, 120.35009, 127.52012, 140.32854, 124.30901, 129.54844, 120.34761, 117.3044, 124.52207, 119.79727, 123.91937, 130.22325, 134.76878, 129.58495, 126.78381, 128.02734, 131.80019, 134.30048, 129.83651, 128.07679, 129.05543, 127.50277, 124.63815, 138.50168, 129.18967, 127.70615, 130.54596, 126.52269, 120.10145, 137.09323, 127.99291, 126.27201, 127.5469, 126.82499, 120.33146, 131.02219, 138.38244, 127.73154, 124.62522, 124.09889, 127.02213, 129.13224, 123.0657, 130.25466, 117.88051, 135.18497, 123.0985, 134.7732, 124.10308, 125.59531, 124.06075, 131.07701, 124.86854, 124.24642, 137.03478, 134.87895, 134.64446, 133.54617, 139.83927, 117.18415, 129.72874, 133.82916, 123.02437, 123.86156, 128.31075, 127.53045, 123.5025, 124.79877, 131.16897, 120.54841, 128.57272, 123.59901, 131.81247, 134.65809, 125.30829, 127.81912, 140.63433, 130.02175, 118.71614, 124.22247, 126.17436, 137.55072, 135.04711, 127.03132, 127.78462, 135.0836, 123.4184, 127.22667, 144.04826, 123.60499, 123.04053, 128.6098, 141.7838, 124.69789, 143.43674, 131.09119, 134.27292, 127.06719, 134.48933, 134.0234, 132.55661, 120.76331, 126.61818, 120.20754, 138.53348, 119.453, 123.27878, 134.05581, 138.73624, 125.6481, 130.25993, 130.44337, 123.26283, 134.87355, 126.03127, 136.0215, 129.96215, 126.62932, 134.61448, 120.8666, 117.81271, 125.57089, 120.77463, 117.02822, 141.04993, 120.27299, 145.56715, 134.11136, 130.73347, 136.44074, 144.61448, 123.02018, 113.00428, 137.04335, 123.58303, 116.68515, 128.03353, 133.86004, 121.3673, 119.25353, 126.25889, 129.76321, 120.96691, 131.59935, 126.45226, 126.35645, 125.29416, 122.78265, 127.56362, 124.00412, 130.5598, 144.33894, 132.78916, 126.04931, 117.25792, 122.92363, 116.0763, 117.87049, 131.08369, 129.01981, 127.58361, 130.23261, 131.42817, 121.00855, 135.18106, 127.00086, 125.52852, 128.48502, 139.09752, 136.62882, 129.64377, 126.05672, 124.54414, 127.78249, 124.00606, 136.37203, 130.08499, 116.93941, 144.23668, 121.34782, 128.20779, 125.34428, 126.87811, 132.25155, 136.68052, 141.30085, 141.35957, 130.8037, 137.65235, 115.27942, 133.47762, 126.19577, 136.71064, 131.47162, 137.58851, 126.66025, 117.26894, 130.11221, 127.0637, 130.02992, 121.04548, 127.26077, 134.21576, 134.9865, 118.51286, 127.0754, 129.17427, 126.93902, 131.00287, 133.80102, 126.59677, 131.82613, 118.40186, 129.88259, 144.3438, 141.25153, 131.92665, 119.30012, 124.26355, 127.13019, 135.06353, 131.53975, 121.03938, 129.41421, 126.00338, 124.51132, 116.64657, 127.77093, 131.72434, 126.0457, 130.27766, 131.32192, 121.02749, 128.37534, 135.29711, 126.51955, 130.8323, 124.89573, 118.4087, 133.34147, 138.77813, 124.84385, 117.78328, 129.60972, 141.54465, 130.53456, 140.31167, 140.79329, 127.29975, 127.50541, 126.2913, 140.53109, 144.61845, 141.03165, 131.26029, 127.04613, 127.7877, 124.55632, 130.36999, 128.2999, 137.23257, 136.80078, 124.33146, 127.04504, 120.78634, 127.56419, 145.75575, 133.88259, 124.02258, 138.54982, 124.36512, 133.26488, 122.92111, 126.98033, 137.04979, 124.7338, 130.0102, 126.62447, 124.75025, 126.07511, 129.07738, 126.85594, 121.3827, 126.68087, 132.0489, 135.06198, 138.51652, 131.75455, 130.54067, 134.7258, 125.45985, 125.76778, 123.7708, 128.56753, 124.18434, 127.5317, 122.50208, 117.55538, 127.6428, 123.13954, 124.05207, 127.30703, 128.16874, 127.02918, 134.45949, 128.09241, 127.54816, 131.8194, 132.54338, 128.40157, 129.87578, 121.33301, 128.37619, 130.87369, 131.02306, 130.60589, 123.22248, 126.56095, 118.89924, 119.80151, 131.28868, 121.22026, 130.192, 134.67895, 123.89639, 124.35044, 131.34577, 134.18052, 123.80634, 131.79483, 114.00598, 141.80021, 118.81893, 133.40675, 136.17309, 130.87933, 123.29492, 116.54457, 121.09633, 128.09793, 127.12083, 131.26577, 136.82751, 141.7592, 137.79202, 129.84833, 122.09547, 124.61046, 130.02064, 124.02425, 137.57971, 133.03446, 127.17447, 129.45285, 137.68595, 126.10363, 132.00798, 130.79346, 130.8598, 124.29503, 126.00638, 119.59556, 116.83984, 116.88835, 135.15805, 111.57968, 116.34251, 121.56708, 118.60174, 123.71509, 127.33291, 123.58571, 131.66312, 134.64439, 120.96554, 131.3466, 130.51609, 134.02736, 127.05196, 131.59706, 119.26508, 131.0942, 133.51097, 127.86953, 117.13083, 121.54106, 147.7508, 141.51878, 133.35659, 118.82753, 141.78992, 124.03161, 129.5925, 126.09263, 137.57323, 129.28533, 131.02212, 137.19469, 130.63527, 128.39098, 136.25004, 129.21339, 130.37172, 109.06655, 134.94299, 127.53264, 126.81389, 111.11779, 134.39515, 142.26065, 127.95856, 131.30111, 120.56592, 120.96227, 137.11478, 130.24746, 134.50568, 120.26354, 131.35702, 134.79213, 125.05936, 138.94089, 120.81219, 131.63406, 119.79745, 131.80065, 120.76105, 130.59255, 135.56778, 128.05147, 134.34786, 134.0647, 120.34455, 127.19482, 133.37012, 122.95788, 119.14222, 134.38906, 138.47439, 120.71785, 128.30103, 121.02206, 135.08215, 138.04035, 136.88857, 134.09421, 131.07632, 128.02108, 135.00707, 123.79684, 127.80875, 125.50604, 124.69913, 147.02118, 125.3151, 132.75672, 126.73189, 129.40695, 128.43812, 134.03012, 138.04335, 138.25644, 130.74872, 123.07718, 133.84958, 127.50926, 118.66574, 122.03915, 136.85147, 136.78771, 126.76198, 133.94149, 126.82539, 120.38363, 127.28196, 140.64251, 143.8299, 137.76149, 137.27188, 123.60757, 129.52506, 140.5182, 124.44258, 126.55425, 124.25921, 141.88432, 140.001, 130.79373, 118.00177, 117.21197, 124.35018, 127.18146, 132.56134, 124.54708, 135.25543, 123.13611, 135.13358, 126.31279, 131.51449, 133.57487, 121.35731, 136.33667, 136.35373, 122.76037, 116.00891, 133.57517, 130.07687, 128.87936, 141.66213, 120.55324, 140.69773, 124.0432, 133.57789, 134.8146, 131.08452, 129.64251, 127.55552, 113.50344, 134.52814, 125.21227, 135.06259, 126.32606, 109.02639, 134.04049, 124.32424, 126.12348, 126.56041, 127.14898, 119.2936, 129.45339, 116.55405, 123.82824, 129.10982, 130.81851, 130.50226, 130.86148, 131.56177, 128.05247, 130.16476, 129.8669, 127.35786, 126.6075, 119.26813, 118.81221, 124.82788, 126.73877, 112.73046, 124.0143, 125.81203, 122.01281, 132.52919, 112.65373, 133.36933, 133.6861, 131.2475, 133.00492, 129.21284, 120.35134, 127.29648, 138.56797, 138.00107, 130.358, 125.66011, 123.32867, 129.3869, 134.03929, 126.20495, 123.08408, 127.02662, 124.34441, 127.01601, 122.61052, 131.34162, 128.53799, 135.85992, 124.75021, 135.79495, 130.74732, 129.50232, 122.8058, 125.82053, 113.39791, 129.64542, 123.02809, 130.59147, 117.58851, 133.27837, 122.84928, 138.25256, 127.26398, 129.80494, 134.54196, 124.0077, 126.36288, 127.53502, 128.80282, 132.70988, 124.65092, 134.38933, 129.28626, 129.27899, 133.14336, 130.45498, 126.68037, 134.61301, 125.75044, 123.67471, 133.93833, 118.29594, 123.84819, 141.65555, 134.7768, 126.2317, 133.72743, 130.37877, 123.52503, 123.37026, 125.03771, 129.68296, 131.8496, 121.25599, 117.53953, 125.82294, 135.25955, 131.88085, 126.59946, 115.343, 131.24785, 131.04909, 130.58041, 134.00046, 118.56553, 136.509, 133.10447, 133.78892, 120.06426, 113.85356, 130.75578, 127.55636]

Average Numbers of Workers in Queue at 08:30, 08:45, and 09:00:
Number of simulations run: **1280**

Following the above pattern, the same process was run on finding the standard deviation of 100 samples of data of the number of workers for all three times specified. The standard deviations are listed below:

Standard Deviations of each time:
08:30: **10.3741**
08:45: **11.5342**
09:00: **11.9893** (was 13.7062 but after rerunning sims 11.9893 resulted)

Using the chart for reference of the Tchebycheff's Theorem -being conservative- it was seen that a difference of σ/12 should be run to ensure a 99% accuracy, or at least 1280 simulations.

The resulting data of the 1280 simulations looks like:

08:30:

[63, 62, 71, 59, 69, 57, 52, 48, 85, 67, 54, 50, 70, 66, 60, 62, 44, 60, 82, 76, 73, 50, 56, 52, 62, 58, 45, 62, 61, 70, 71, 73, 67, 56, 62, 67, 50, 71, 46, 43, 68, 66, 44, 55, 32, 57, 65, 63, 60, 61, 47, 69, 79, 63, 75, 53, 61, 56, 53, 74, 64, 55, 62, 85, 66, 42, 57, 67, 63, 62, 59, 79, 73, 63, 59, 61, 81, 70, 83, 56, 60, 44, 57, 52, 55, 59, 47, 64, 72, 65, 59, 55, 61, 51, 54, 48, 56, 80, 67, 75, 60, 51, 61, 51, 63, 57, 70, 77, 74, 66, 57, 58, 56, 62, 78, 60, 73, 65, 59, 61, 68, 68, 66, 52, 77, 54, 34, 68, 33, 44, 67, 70, 61, 65, 47, 73, 60, 63, 69, 48, 65, 58, 63, 61, 70, 52, 58, 67, 66, 53, 53, 75, 73, 60, 56, 42, 73, 47, 49, 57, 56, 87, 63, 56, 56, 66, 67, 51, 59, 54, 66, 55, 84, 70, 52, 77, 61, 65, 53, 56, 63, 60, 55, 55, 69, 62, 58, 71, 67, 65, 57, 37, 61, 50, 36, 44, 55, 66, 61, 59, 64, 51, 43, 67, 73, 66, 52, 72, 68, 58, 62, 60, 50, 82, 60, 67, 69, 39, 83, 83, 73, 68, 83, 51, 60, 63, 80, 59, 71, 40, 51, 53, 54, 70, 44, 62, 48, 64, 53, 49, 58, 55, 70, 51, 47, 63, 50, 46, 50, 62, 62, 68, 58, 46, 59, 58, 84, 47, 41, 56, 49, 63, 68, 61, 80, 62, 68, 67, 41, 71, 57, 45, 68, 64, 56, 57, 55, 76, 85, 51, 56, 61, 75, 64, 58, 67, 50, 51, 64, 66, 55, 49, 75, 62, 69, 47, 65, 71, 55, 57, 69, 79, 68, 62, 60, 64, 74, 63, 62, 63, 86, 63, 52, 57, 56, 71, 69, 46, 85, 64, 53, 55, 81, 73, 67, 50, 65, 52, 60, 55, 54, 67, 85, 48, 70, 66, 71, 43, 31, 52, 65, 82, 63, 47, 48, 65, 49, 71, 70, 58, 72, 47, 63, 68, 60, 58, 75, 64, 58, 55, 66, 69, 54, 62, 45, 61, 50, 65, 59, 53, 60, 50, 49, 66, 69, 57, 63, 46, 55, 64, 60, 51, 39, 68, 76, 65, 55, 55, 59, 69, 51, 55, 56, 55, 81, 69, 62, 49, 58, 43, 50, 87, 56, 70, 79, 76, 51, 65, 65, 66, 66, 64, 42, 61, 60, 57, 65, 53, 49, 58, 70, 75, 84, 68, 67, 49, 48, 66, 42, 70, 64, 68, 60, 65, 64, 52, 59, 63, 63, 61, 52, 59, 43, 58, 48, 70, 56, 66, 64, 63, 57, 53, 67, 46, 74, 63, 63, 66, 69, 58, 58, 68, 45, 62, 60, 61, 70, 51, 80, 49, 53, 51, 44, 56, 75, 52, 54, 65, 62, 59, 54, 52, 62, 52, 52, 61, 69, 50, 38, 57, 57, 45, 72, 71, 57, 46, 59, 73, 73, 49, 58, 65, 59, 56, 58, 55, 69, 40, 49, 80, 64, 75, 62, 66, 41, 53, 47, 62, 54, 51, 59, 49, 50, 52, 49, 64, 60, 68, 57, 61, 65, 63, 54, 45, 61, 48, 60, 49, 54, 60, 67, 92, 58, 56, 56, 57, 45, 50, 69, 74, 55, 44, 58, 56, 71, 81, 55, 44, 41, 72, 61, 57, 66, 65, 58, 75, 61, 61, 51, 59, 76, 69, 81, 46, 54, 63, 83, 65, 66, 49, 72, 61, 81, 76, 69, 63, 67, 71, 60, 56, 53, 67, 44, 71, 59, 47, 47, 69, 59, 65, 45, 51, 70, 65, 74, 58, 47, 71, 43, 59, 61, 60, 71, 57, 43, 61, 45, 56, 61, 49, 42, 62, 69, 63, 58, 72, 77, 59, 70, 76, 54, 72, 70, 64, 57, 70, 61, 58, 74, 40, 61, 62, 63, 48, 62, 65, 58, 69, 79, 71, 62, 69, 57, 77, 48, 65, 45, 55, 63, 58, 67, 61, 61, 63, 57, 67, 68, 58, 93, 69, 68, 56, 50, 55, 68, 62, 65, 71, 70, 56, 80, 70, 55, 65, 66, 52, 76, 59, 75, 41, 49, 72, 54, 47, 48, 63, 70, 71, 52, 70, 45, 60, 51, 58, 60, 75, 76, 52, 58, 81, 61, 63, 57, 84, 54, 58, 51, 54, 62, 57, 86, 63, 69, 60, 72, 58, 56, 60, 43, 68, 59, 59, 64, 66, 68, 53, 50, 59, 40, 50, 49, 73, 73, 69, 57, 46, 59, 48, 57, 64, 55, 73, 55, 39, 65, 63, 54, 60, 60, 46, 69, 62, 42, 60, 54, 69, 41, 81, 81, 68, 60, 48, 53, 55, 58, 58, 59, 68, 67, 69, 72, 77, 48, 53, 57, 69, 46, 66, 81, 78, 59, 62, 66, 61, 56, 61, 66, 83, 61, 63, 58, 46, 64, 73, 75, 79, 56, 54, 56, 75, 55, 49, 60, 63, 56, 51, 58, 57, 60, 76, 62, 48, 59, 67, 52, 43, 67, 77, 72, 49, 60, 60, 64, 60, 71, 54, 65, 46, 81, 68, 65, 58, 66, 75, 66, 51, 61, 73, 53, 68, 63, 49, 64, 86, 78, 63, 58, 66, 83, 56, 66, 49, 46, 61, 55, 56, 66, 57, 49, 74, 70, 81, 55, 44, 67, 56, 60, 71, 54, 78, 52, 63, 70, 71, 59, 61, 47, 55, 61, 64, 52, 50, 52, 68, 72, 51, 89, 64, 51, 84, 74, 55, 67, 64, 50, 59, 63, 55, 67, 56, 63, 61, 66, 63, 68, 58, 59, 63, 56, 71, 54, 53, 55, 57, 54, 66, 58, 64, 51, 72, 55, 61, 41, 64, 61, 74, 79, 38, 71, 86, 64, 57, 65, 52, 70, 46, 62, 57, 61, 47, 52, 54, 55, 61, 64, 67, 46, 70, 55, 67, 63, 71, 59, 55, 61, 61, 70, 42, 59, 72, 52, 48, 56, 58, 79, 60, 73, 83, 73, 51, 73, 68, 73, 89, 61, 66, 64, 52, 46, 53, 58, 53, 75, 65, 48, 48, 54, 79, 68, 58, 59, 76, 61, 53, 72, 57, 60, 45, 63, 55, 85, 54, 64, 53, 72, 70, 68, 77, 57, 65, 69, 65, 52, 60, 64, 77, 55, 73, 69, 51, 82, 70, 51, 53, 59, 65, 75, 66, 65, 54, 45, 52, 57, 51, 56, 67, 53, 51, 67, 60, 60, 62, 46, 50, 78, 63, 56, 64, 70, 62, 46, 44, 65, 53, 62, 60, 57, 56, 70, 69, 61, 63, 54, 65, 47, 46, 49, 57, 63, 70, 60, 44, 60, 72, 69, 60, 61, 56, 67, 55, 61, 48, 69, 38, 54, 65, 56, 71, 51, 64, 68, 49, 42, 63, 61, 67, 63, 64, 78, 54, 46, 66, 46, 67, 66, 65, 75, 50, 56, 67, 61, 64, 45, 55, 54, 66, 52, 61, 64, 66, 70, 59, 51, 54, 67, 66, 52, 56, 56, 70, 63, 49, 79, 69, 56, 71, 58, 56, 59, 50, 82, 61, 76, 74, 54, 63, 61, 61, 73, 61, 50, 52, 75, 71, 59, 88, 63, 53, 59, 56, 46, 68, 80, 58, 47, 64, 51, 61, 77, 70, 60, 58, 58, 62, 60, 74, 50, 70, 40, 47, 68, 76, 60, 74, 69, 50, 55, 56, 60, 78, 51, 64, 63, 49, 55, 83, 41, 72, 54, 56, 59, 77, 53, 75, 67, 60, 57, 48, 79, 63, 54, 52, 72, 60, 72, 60, 57, 66, 56, 61, 59, 42, 74, 44, 59, 60, 57, 74, 79, 45, 74, 66, 67, 62, 50, 70, 55, 51, 63, 56, 81, 48, 65, 54, 48, 47, 49, 53, 70, 65, 68, 35, 56, 44, 37, 53, 52, 64, 55, 77, 60, 45, 75, 46, 59, 49, 64, 50, 64, 52]

08:45:

[105, 94, 106, 92, 106, 104, 81, 89, 117, 108, 104, 84, 102, 104, 83, 95, 88, 89, 106, 109, 122, 98, 84, 70, 84, 91, 87, 100, 90, 106, 96, 112, 90, 86, 103, 97, 85, 110, 98, 77, 99, 105, 71, 92, 75, 92, 97, 107, 79, 99, 81, 106, 121, 88, 103, 92, 101, 84, 77, 119, 103, 79, 106, 113, 105, 75, 92, 96, 91, 97, 90, 100, 112, 92, 96, 94, 115, 100, 115, 105, 84, 84, 98, 80, 86, 99, 80, 94, 110, 102, 104, 86, 97, 93, 90, 87, 89, 115, 100, 108, 93, 98, 90, 91, 83, 92, 103, 109, 111, 105, 96, 91, 91, 89, 114, 101, 114, 94, 79, 93, 100, 106, 102, 87, 105, 94, 76, 101, 61, 81, 93, 110, 97, 97, 78, 105, 81, 100, 94, 83, 94, 96, 97, 88, 101, 100, 101, 107, 100, 95, 76, 104, 91, 105, 88, 83, 113, 87, 91, 87, 83, 133, 96, 92, 82, 109, 99, 80, 97, 87, 109, 78, 148, 111, 87, 131, 83, 107, 97, 84, 99, 90, 89, 93, 99, 94, 98, 116, 97, 127, 97, 64, 90, 93, 80, 95, 87, 99, 101, 108, 88, 73, 73, 113, 110, 97, 74, 107, 97, 82, 113, 94, 95, 116, 97, 109, 102, 72, 111, 124, 115, 96, 114, 79, 96, 102, 131, 88, 100, 84, 91, 84, 88, 102, 76, 103, 91, 92, 89, 82, 99, 87, 103, 85, 88, 103, 79, 91, 84, 115, 87, 112, 94, 81, 100, 94, 114, 80, 83, 96, 88, 113, 95, 80, 121, 116, 110, 102, 60, 103, 97, 68, 112, 97, 93, 106, 99, 109, 115, 66, 98, 115, 103, 103, 96, 101, 82, 89, 102, 99, 94, 83, 110, 92, 104, 86, 87, 98, 94, 86, 112, 105, 94, 87, 101, 99, 115, 98, 94, 88, 140, 96, 93, 92, 89, 114, 95, 77, 126, 91, 108, 102, 110, 109, 109, 85, 92, 90, 83, 86, 102, 106, 129, 83, 97, 101, 115, 73, 60, 87, 107, 119, 91, 83, 81, 101, 86, 106, 117, 76, 115, 92, 85, 101, 102, 89, 113, 98, 100, 100, 99, 105, 99, 78, 77, 101, 82, 93, 87, 90, 96, 85, 84, 113, 103, 104, 93, 90, 118, 86, 96, 82, 73, 106, 104, 100, 97, 84, 93, 94, 89, 89, 80, 111, 123, 102, 106, 88, 97, 90, 78, 117, 88, 125, 116, 115, 78, 106, 103, 94, 102, 100, 70, 94, 99, 84, 102, 102, 96, 97, 100, 108, 127, 103, 96, 84, 78, 101, 77, 101, 96, 100, 96, 101, 113, 89, 104, 90, 83, 103, 77, 92, 78, 111, 94, 98, 103, 89, 107, 97, 83, 84, 87, 82, 120, 105, 103, 113, 116, 94, 96, 108, 84, 88, 87, 93, 96, 85, 101, 88, 88, 106, 92, 102, 112, 95, 81, 92, 94, 88, 87, 89, 88, 74, 87, 101, 97, 96, 68, 88, 74, 78, 125, 106, 79, 78, 97, 119, 103, 91, 81, 102, 103, 99, 97, 84, 107, 81, 81, 120, 92, 110, 104, 97, 58, 91, 86, 91, 90, 85, 96, 82, 86, 83, 108, 102, 98, 107, 93, 105, 99, 99, 90, 85, 100, 71, 101, 79, 93, 92, 102, 128, 100, 94, 95, 99, 93, 91, 103, 101, 98, 95, 86, 84, 103, 106, 88, 78, 63, 105, 92, 87, 109, 98, 94, 113, 105, 91, 89, 95, 124, 106, 96, 76, 82, 98, 110, 98, 103, 85, 99, 114, 117, 112, 100, 94, 100, 110, 87, 88, 100, 107, 79, 106, 91, 75, 93, 110, 86, 92, 84, 92, 103, 104, 112, 91, 85, 106, 71, 95, 83, 93, 89, 103, 70, 89, 78, 91, 91, 97, 77, 111, 93, 95, 96, 113, 110, 94, 103, 110, 77, 103, 101, 97, 120, 98, 86, 83, 119, 70, 89, 98, 103, 82, 90, 94, 98, 122, 110, 101, 101, 105, 98, 115, 83, 100, 68, 100, 120, 103, 92, 93, 91, 104, 94, 108, 109, 92, 124, 108, 101, 84, 76, 89, 101, 101, 104, 98, 112, 89, 111, 102, 102, 93, 98, 85, 105, 91, 125, 80, 96, 95, 76, 85, 83, 101, 101, 109, 95, 104, 92, 94, 88, 106, 86, 118, 116, 72, 96, 116, 101, 88, 87, 120, 89, 116, 80, 84, 90, 98, 149, 93, 109, 103, 99, 92, 76, 97, 75, 107, 96, 105, 103, 108, 97, 88, 93, 100, 78, 96, 80, 112, 116, 102, 84, 64, 94, 98, 101, 98, 91, 94, 88, 72, 101, 105, 96, 98, 102, 85, 98, 91, 79, 93, 88, 101, 76, 122, 121, 96, 104, 86, 86, 106, 100, 93, 96, 112, 100, 98, 93, 117, 76, 91, 112, 106, 74, 92, 121, 110, 94, 108, 92, 110, 78, 105, 101, 112, 94, 86, 90, 76, 83, 106, 117, 111, 77, 82, 93, 104, 94, 76, 101, 95, 102, 83, 84, 110, 87, 119, 93, 81, 107, 114, 96, 72, 100, 98, 119, 86, 93, 85, 105, 99, 100, 86, 108, 76, 127, 115, 98, 78, 107, 118, 85, 97, 99, 92, 96, 99, 88, 81, 101, 123, 96, 91, 97, 111, 117, 82, 105, 76, 82, 93, 90, 99, 108, 102, 73, 110, 101, 116, 79, 85, 119, 89, 108, 107, 95, 106, 90, 89, 98, 103, 93, 92, 84, 89, 98, 108, 79, 82, 92, 104, 114, 80, 124, 90, 87, 118, 120, 97, 110, 104, 82, 99, 103, 74, 90, 99, 99, 85, 100, 100, 100, 99, 90, 97, 86, 106, 80, 97, 86, 100, 81, 91, 85, 89, 106, 103, 91, 99, 77, 87, 106, 96, 118, 66, 104, 123, 102, 92, 101, 89, 94, 88, 100, 83, 117, 80, 84, 86, 95, 99, 93, 103, 83, 105, 90, 105, 97, 104, 86, 92, 100, 97, 109, 73, 92, 108, 90, 99, 118, 95, 113, 92, 106, 120, 101, 95, 113, 109, 126, 124, 86, 98, 104, 85, 82, 85, 89, 96, 110, 107, 87, 98, 95, 111, 116, 98, 83, 98, 96, 86, 98, 76, 102, 75, 93, 93, 121, 76, 122, 82, 103, 96, 91, 121, 106, 100, 97, 108, 87, 96, 100, 112, 86, 100, 95, 96, 127, 120, 115, 90, 91, 100, 106, 98, 105, 88, 74, 80, 79, 93, 94, 89, 89, 98, 99, 92, 92, 98, 73, 81, 118, 88, 101, 102, 110, 91, 80, 71, 100, 79, 101, 92, 89, 112, 105, 117, 105, 95, 85, 100, 76, 70, 81, 92, 99, 94, 88, 93, 95, 112, 110, 88, 95, 90, 103, 88, 100, 85, 100, 82, 84, 97, 104, 100, 105, 93, 101, 81, 81, 103, 103, 98, 89, 105, 120, 95, 87, 102, 91, 94, 93, 107, 96, 81, 91, 99, 98, 97, 75, 90, 90, 96, 85, 82, 90, 102, 115, 94, 83, 92, 103, 105, 81, 93, 81, 103, 96, 84, 123, 106, 80, 106, 90, 89, 96, 69, 118, 95, 108, 107, 88, 98, 95, 114, 113, 90, 91, 92, 116, 110, 115, 125, 82, 89, 93, 84, 84, 100, 110, 97, 90, 96, 82, 99, 104, 95, 86, 99, 94, 94, 108, 110, 90, 106, 75, 87, 106, 99, 91, 113, 101, 77, 92, 91, 100, 114, 95, 112, 96, 80, 85, 118, 97, 117, 92, 89, 97, 111, 86, 109, 95, 92, 88, 89, 117, 84, 90, 81, 112, 93, 93, 85, 93, 106, 85, 99, 109, 75, 95, 94, 84, 90, 99, 100, 110, 81, 120, 98, 91, 96, 81, 99, 75, 92, 103, 81, 118, 98, 98, 96, 77, 79, 81, 79, 106, 96, 96, 77, 95, 80, 84, 73, 80, 97, 96, 105, 96, 92, 115, 84, 107, 83, 91, 78, 101, 89]

09:00:

[144, 150, 151, 128, 161, 139, 124, 133, 153, 153, 147, 130, 139, 149, 124, 133, 133, 136, 135, 142, 152, 129, 127, 95, 116, 121, 134, 128, 118, 147, 137, 141, 123, 121, 150, 150, 115, 145, 136, 116, 137, 136, 102, 127, 124, 122, 145, 154, 120, 121, 125, 153, 163, 125, 136, 140, 132, 118, 120, 166, 148, 131, 133, 147, 150, 116, 138, 128, 126, 137, 119, 145, 143, 131, 127, 133, 159, 134, 156, 129, 129, 120, 127, 126, 115, 135, 112, 136, 149, 137, 153, 124, 134, 131, 147, 129, 120, 166, 122, 147, 151, 126, 125, 132, 128, 129, 145, 132, 135, 150, 141, 136, 124, 120, 143, 143, 138, 138, 117, 146, 145, 132, 133, 112, 147, 147, 117, 139, 93, 129, 119, 136, 144, 147, 115, 143, 142, 140, 141, 119, 130, 142, 118, 133, 143, 144, 140, 148, 143, 138, 135, 153, 135, 148, 131, 129, 136, 125, 132, 144, 127, 173, 138, 122, 113, 157, 155, 123, 139, 122, 135, 108, 181, 162, 127, 168, 131, 159, 126, 118, 142, 130, 132, 134, 139, 117, 126, 160, 129, 158, 127, 98, 132, 126, 95, 135, 131, 135, 139, 147, 120, 114, 114, 161, 144, 133, 111, 144, 129, 125, 150, 135, 131, 146, 139, 143, 144, 106, 162, 170, 155, 125, 139, 119, 128, 146, 173, 118, 139, 112, 115, 117, 126, 155, 120, 134, 131, 127, 116, 117, 155, 109, 131, 126, 116, 140, 115, 120, 132, 143, 122, 147, 131, 112, 125, 123, 147, 126, 118, 133, 164, 169, 141, 118, 162, 158, 141, 158, 104, 139, 140, 99, 149, 147, 125, 147, 148, 137, 156, 100, 141, 183, 139, 129, 125, 132, 128, 116, 131, 140, 137, 135, 147, 139, 140, 133, 113, 133, 129, 124, 145, 138, 126, 135, 141, 128, 150, 134, 136, 124, 174, 159, 122, 123, 143, 139, 138, 115, 160, 127, 134, 137, 162, 144, 135, 115, 138, 145, 113, 118, 147, 158, 179, 143, 144, 130, 168, 123, 102, 131, 153, 155, 142, 113, 122, 157, 129, 126, 161, 119, 139, 127, 117, 140, 145, 128, 144, 146, 136, 152, 133, 155, 124, 123, 128, 130, 123, 130, 133, 118, 141, 120, 122, 142, 147, 144, 136, 117, 153, 120, 133, 129, 93, 165, 147, 128, 157, 128, 124, 134, 124, 133, 116, 149, 168, 145, 135, 139, 126, 112, 116, 162, 116, 172, 145, 156, 117, 141, 137, 149, 134, 130, 98, 148, 130, 133, 128, 138, 126, 140, 146, 149, 162, 140, 128, 120, 116, 134, 103, 152, 126, 145, 135, 150, 157, 132, 136, 123, 117, 132, 134, 120, 131, 140, 125, 146, 140, 123, 145, 126, 119, 128, 112, 121, 155, 143, 141, 141, 141, 135, 139, 150, 117, 132, 129, 130, 144, 132, 149, 131, 123, 129, 112, 147, 159, 132, 116, 126, 127, 124, 123, 119, 121, 123, 140, 151, 126, 133, 105, 127, 114, 125, 175, 148, 108, 104, 124, 159, 155, 130, 110, 149, 134, 134, 146, 121, 135, 119, 132, 157, 132, 149, 139, 121, 104, 121, 116, 136, 122, 122, 139, 132, 132, 115, 139, 125, 130, 155, 132, 144, 129, 128, 139, 116, 136, 102, 137, 119, 117, 124, 133, 160, 138, 131, 135, 139, 129, 140, 153, 151, 137, 118, 128, 125, 141, 158, 120, 106, 100, 160, 134, 132, 147, 140, 135, 147, 152, 136, 127, 128, 161, 133, 123, 106, 116, 146, 139, 128, 145, 120, 156, 140, 137, 150, 131, 125, 145, 150, 126, 124, 132, 134, 111, 133, 126, 97, 128, 132, 124, 137, 131, 141, 140, 152, 151, 133, 109, 136, 117, 132, 116, 132, 121, 133, 112, 124, 109, 141, 122, 124, 115, 153, 129, 132, 127, 135, 157, 141, 154, 147, 121, 125, 136, 121, 172, 137, 126, 129, 156, 116, 123, 132, 139, 125, 129, 129, 161, 167, 139, 138, 125, 150, 139, 156, 109, 159, 98, 142, 161, 141, 149, 128, 142, 151, 120, 144, 158, 132, 160, 148, 133, 126, 109, 135, 132, 141, 130, 142, 149, 130, 140, 143, 122, 128, 144, 125, 128, 121, 157, 115, 125, 125, 123, 117, 125, 131, 153, 136, 124, 136, 119, 140, 119, 134, 132, 151, 161, 114, 129, 149, 137, 126, 142, 162, 114, 187, 120, 129, 122, 129, 166, 121, 161, 139, 141, 125, 120, 127, 124, 152, 143, 136, 151, 141, 147, 129, 137, 137, 114, 125, 113, 147, 156, 147, 118, 117, 149, 128, 154, 137, 126, 125, 123, 117, 141, 146, 142, 140, 139, 123, 138, 131, 127, 125, 140, 142, 105, 156, 170, 151, 155, 113, 126, 129, 155, 135, 131, 154, 153, 141, 121, 149, 113, 135, 145, 140, 119, 133, 149, 148, 122, 135, 120, 156, 123, 150, 147, 155, 128, 129, 137, 114, 124, 151, 167, 144, 118, 130, 121, 154, 138, 124, 152, 147, 152, 121, 123, 141, 140, 152, 126, 116, 141, 159, 128, 122, 158, 131, 155, 125, 133, 115, 152, 134, 142, 139, 142, 124, 161, 155, 145, 120, 137, 159, 117, 138, 136, 130, 128, 128, 133, 118, 153, 170, 132, 124, 127, 160, 150, 118, 162, 116, 119, 126, 113, 138, 156, 143, 109, 151, 142, 160, 125, 119, 166, 136, 161, 136, 116, 145, 134, 135, 127, 131, 125, 119, 123, 116, 142, 151, 115, 122, 127, 143, 157, 135, 161, 137, 123, 150, 175, 133, 157, 152, 132, 141, 142, 134, 132, 136, 143, 115, 157, 136, 147, 129, 130, 132, 113, 143, 111, 137, 125, 141, 114, 126, 130, 104, 133, 148, 111, 149, 106, 135, 139, 140, 165, 123, 123, 168, 137, 146, 144, 137, 137, 132, 140, 111, 163, 122, 122, 123, 120, 133, 131, 134, 147, 148, 128, 157, 131, 142, 115, 128, 151, 134, 143, 126, 127, 147, 128, 141, 161, 134, 134, 129, 141, 166, 136, 125, 168, 133, 147, 155, 123, 137, 130, 118, 117, 120, 131, 139, 147, 148, 126, 131, 144, 142, 148, 131, 111, 143, 144, 108, 134, 116, 125, 115, 120, 133, 175, 123, 144, 128, 129, 142, 135, 144, 126, 135, 125, 149, 127, 127, 144, 149, 128, 129, 128, 132, 184, 172, 157, 121, 140, 156, 136, 139, 137, 142, 125, 109, 112, 132, 117, 130, 121, 130, 140, 132, 135, 124, 111, 119, 165, 141, 126, 145, 142, 128, 112, 111, 140, 118, 130, 130, 135, 145, 143, 161, 132, 137, 130, 122, 115, 104, 103, 138, 125, 149, 118, 129, 128, 151, 139, 129, 126, 122, 151, 135, 125, 136, 138, 135, 117, 133, 154, 158, 130, 134, 142, 114, 109, 137, 128, 137, 132, 153, 171, 120, 137, 139, 118, 124, 134, 132, 136, 131, 123, 130, 133, 137, 120, 127, 120, 122, 118, 124, 126, 133, 177, 133, 120, 139, 150, 139, 115, 130, 121, 156, 125, 108, 143, 153, 127, 138, 130, 128, 140, 108, 164, 121, 140, 143, 123, 132, 117, 131, 146, 119, 124, 142, 147, 158, 157, 160, 111, 120, 118, 111, 121, 131, 148, 141, 136, 137, 112, 158, 139, 116, 129, 136, 141, 124, 139, 149, 116, 145, 123, 115, 139, 130, 115, 139, 117, 117, 125, 123, 126, 154, 161, 123, 114, 125, 144, 145, 159, 144, 130, 141, 158, 116, 153, 145, 138, 129, 121, 157, 125, 122, 122, 149, 134, 129, 133, 128, 141, 116, 133, 143, 111, 139, 145, 128, 126, 139, 134, 137, 117, 160, 131, 127, 126, 117, 144, 107, 119, 139, 115, 163, 133, 131, 144, 125, 139, 120, 129, 144, 138, 126, 120, 143, 119, 131, 108, 115, 136, 126, 138, 142, 115, 142, 139, 147, 126, 143, 118, 145, 130]

It can be seen from the above data that the general trend in the number of people is upward as the times increase from 08:30 to 09:00. This shows a sign that the queue gets longer and longer as time goes on and hints at needs for another (or a faster) elevator to lessen the severity of the queue size.

Screenshot of Program

The program itself is all run and simulated on the command line. The results of the simulation can be viewed through the command line and through an output text file and the user must simply uncomment and comment out different lines of code in order to visualize the distributions of different metrics for the simulation. Example output for one simulation is:

```
[Person #1, Arrival Time: 0.33078835783279587, Departure Time: 0.33078849490619094,
Wait Time: 1.370733950789571e-07, Time Boarded Lift: 0.33078849490619094, Walker:
False]
[Person #2, Arrival Time: 0.7108963439594704, Departure Time: 2.3307884949061908,
Wait Time: 1.6198921509467203, Time Boarded Lift: 2.3307884949061908, Walker: False]
[Person #3, Arrival Time: 1.3972405190782042, Departure Time: 2.3307884949061908,
Wait Time: 0.9335479758279865, Time Boarded Lift: 2.3307884949061908, Walker: False]
…
[Person #0, Arrival Time: 11.324609533440597, Departure Time: 0, Wait Time: 0, Time
Boarded Lift: 0, Walker: True]
[Person #0, Arrival Time: 15.003097873387361, Departure Time: 0, Wait Time: 0, Time
Boarded Lift: 0, Walker: True]
[Person #0, Arrival Time: 15.098161737679582, Departure Time: 0, Wait Time: 0, Time
Boarded Lift: 0, Walker: True]
[Person #0, Arrival Time: 15.62587256312548, Departure Time: 0, Wait Time: 0, Time
Boarded Lift: 0, Walker: True]
…
Total Persons walked/took lift: 388
Time last person boarded elevator: 134.8307884949062
There were 61 people in line at 8:30
There were 97 people in line at 8:45
There were 144 people in line at 9:00
A total of 63 people walked.
People that walked to the second floor: 34
People that walked to the third floor: 22
People that walked to the fourth floor: 7
```

These results detail all the data required to answer the problems. The data is then all collected and organized in specific files to keep track of data for the specific metrics required by the prompt (e.g. all the data after running the 1280 simulations for average wait times is stored in a file called average_wait_times.txt.)

A screenshot of all the code and program can be found in the Appendix.

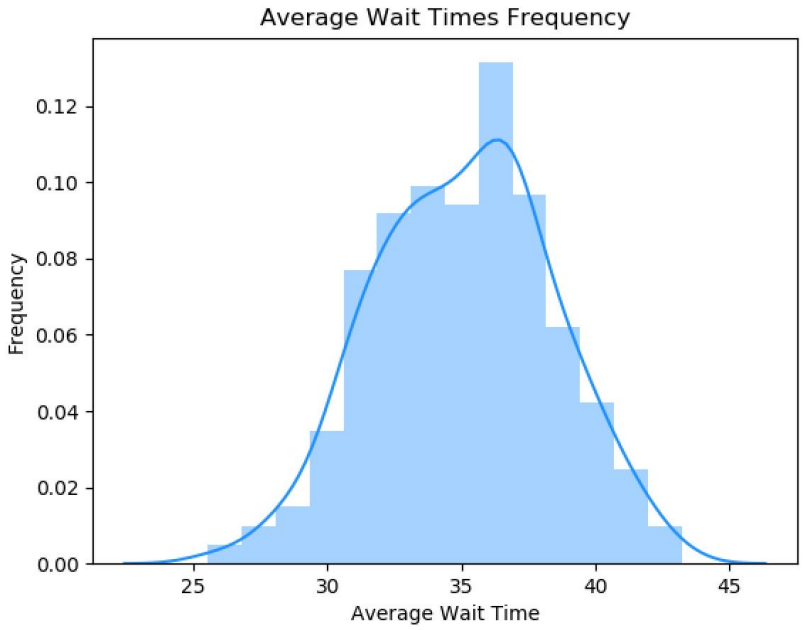
Results Summary

Metrics Summarized in Graphic Form

After the proper number of simulations were run to ensure a 99% accuracy in the data, the metrics for each of the four categories discussed above were visualized using the Matplotlib and Seaborn libraries. The results can be seen below.

Average Wait Times:

The average wait time's conservative sample of 320 simulations appears has a distribution that looks like:



This is approximately a normal distribution minus the skewed dent in the top of the data, so a normality test was run. After running the Kolmogorov-Smirnov test for normality on the data, the data was determined to have A K-S test statistic of 0.0617 and, more importantly, a p-value of .36735. This means that the above data does not differ significantly from that which is normally distributed [2].

Average Number of People that walk to 2nd, 3rd, and 4th Floors:

After running 1280, the distributions of the numbers of people that walked to each floor can be seen below:

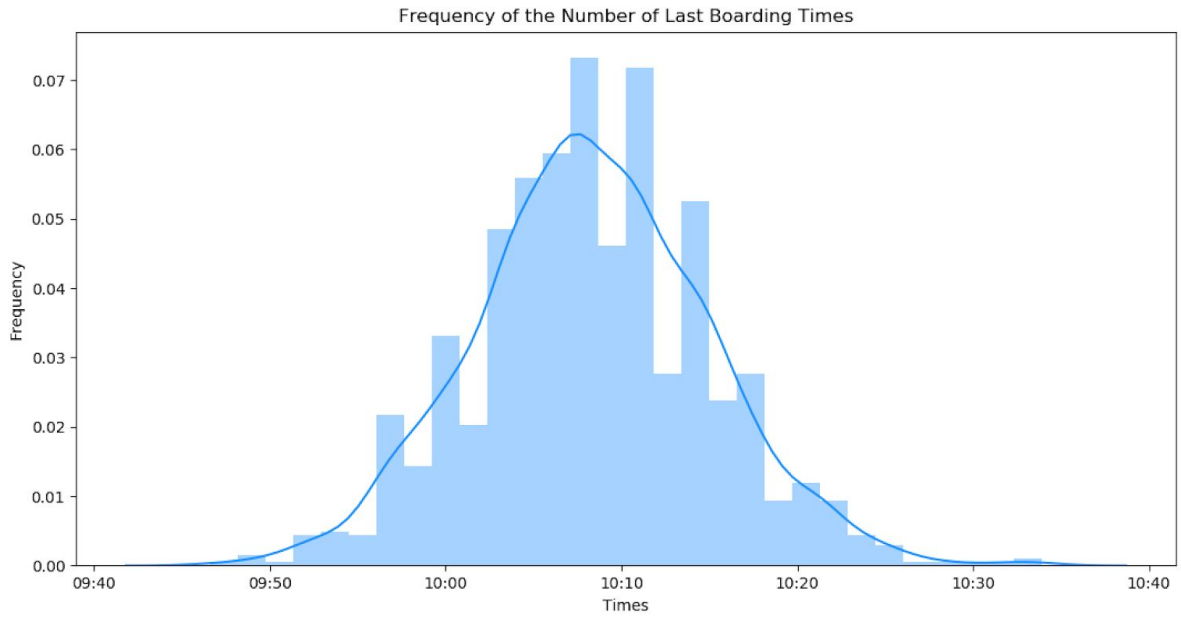
2nd Floor: This is most likely a normal distribution, evident after a normality test was run. After running the Kolmogorov-Smirnov test for normality on the data, the data was determined to have A K-S test statistic of 0.07506 and, more importantly, a p-value of 0.16478. This means that the above data for the 2nd floor does not differ significantly from data that is normally distributed [2].

3rd Floor: This is most likely a normal distribution, evident after a normality test was run. After running the Kolmogorov-Smirnov test for normality on the data, the data was determined to have A K-S test statistic of 0.07679 and, more importantly, a p-value of 0.07065. This means that the above data for the 3rd floor does not differ significantly from data that is normally distributed [2].

4th Floor: This is surprisingly not a normal distribution, evident after a normality test was run. After running the Kolmogorov-Smirnov test for normality on the data, the data was determined to have A K-S test statistic of .09967 and, more importantly, a p-value of 0.09967. This means that the above data for the 4th floor offers strong evidence that the data is not normally distributed [2]. Rather, this distribution appears to resemble a binomial distribution in nature.

Time of Last Boarding of Elevator:

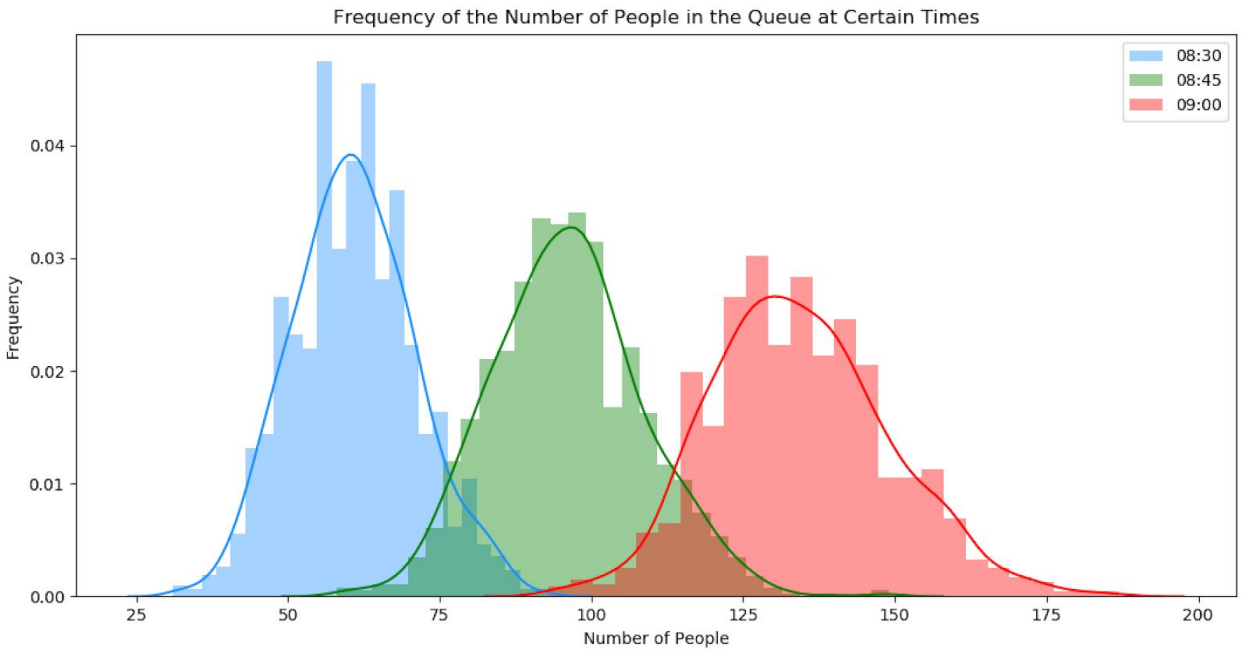
The density curve and histogram of the distribution of the 1280 simulations resulting in the times of the last boarding of the elevator looks like:



This is approximately a normal distribution minus the skewed dent in the top of the data, so a normality test was run. After running the Kolmogorov-Smirnov test for normality on the data, the data was determined to have A K-S test statistic of 0.04688 and, more importantly, a p-value of 0.55591. This means that the above data does not differ significantly from that which is normally distributed [2].

Average Numbers of Workers in Queue at 08:30, 08:45, and 09:00:

After running 1280 simulations, the distributions of the numbers of people that were waiting in the queue at each time can be seen below:



08:30: This is most likely a normal distribution, evident after a normality test was run. After running the Kolmogorov-Smirnov test for normality on the data, the data was determined to have A K-S test statistic of 0.04436 and, more importantly, a p-value of.69644. This means that the above data for the 2nd floor does not differ significantly from data that is normally distributed [2].

08:45: This is most likely a normal distribution, evident after a normality test was run. After running the Kolmogorov-Smirnov test for normality on the data, the data was determined to have A K-S test statistic of 0 .04529 and, more importantly, a p-value of 0.7227. This means that the above data for the 3rd floor does not differ significantly from data that is normally distributed [2].

09:00: This is most likely a normal distribution, evident after a normality test was run. After running the Kolmogorov-Smirnov test for normality on the data, the data was determined to have A K-S test statistic of 0.04672 and, more importantly, a p-value of .75716. This means that the above data for the 2nd floor does not differ significantly from data that is normally distributed [2].

Results Interpreted from a Worker's Perspective

From a worker's point of view, one thing is very clear: if I don't want to wait in an extremely long queue line to board the elevator, I must get to work early, preferably before 08:30. It is clear that, as time goes on, the queue gets longer and longer, and the distribution titled "Frequency of the Number of People in the Queue at Certain Times" shows the trend very clearly. As time

moves on, there is a much higher distribution of people in the queue for the elevator, with the distribution of the people in the elevator queue at 09:00 being effectively double that of the distribution of people in the elevator queue at 08:30.

Additionally, getting to work early is bolstered by the “Frequency of the Number of Last Boarding Times” distribution graph above, for it shows how those who arrive late, especially the last person to arrive, doesn’t even get to board the elevator until much, much later, with the peak of the normal distribution being at being around 10:10. That would be over an hour and ten minutes of waiting time if they arrived at or just before 09:00. That is an outrageous amount of time for workers to wait, so they must arrive earlier for work if they can.

The frequency of the number of people who actually walk is starkly low, especially for those going to the third and fourth floors, as the distributions found in the “Frequency of the Number of People Who Walked to Different Floors” graph illustrate. The workers should keep in mind how the stairs are not very crowded and that they could likely save anywhere from a half hour to an hour and a half of their day if they simply would choose to take the stairs.

So, the workers have effectively two options if they want to not waste lots of minutes from their day:

1. Arrive to work early and take the elevator.
2. Arrive to work at any time but take the stairs.

If the workers do not follow one of these two options, they are shaving valuable hours of their life every week and should reevaluate their life decisions.

Results Interpreted from a Building Owner’s Perspective

From the building owner’s perspective, one thing is clear: the elevator system clearly needs a major revamping. As the graph titled “Average Wait Times Frequency” illustrates, the *average* wait times often lie around 36 whole minutes. And, as the graph titled “Frequency of the Number of Last Boarding Times” shows, the last people to show up in the queue around 09:00 often have to wait over an hour to board the elevator. This 36 minutes of time waiting in the elevator queue is simply untolerable for any company concerned about productivity and not wasting man-hours and effort. The workers are almost certainly less productive and happy due to their long waiting times, and this will certainly hurt company morale and productivity.

The building owner could rectify these problems in several ways, including, but not limited to:

- Increasing the elevator capacity to be more than 12
- Adding additional elevators
- Decreasing the time it takes the elevator to reach other floors

All of the above options should help to rectify the long queue times and thus increase the productivity, happiness, and morale of all the workers by saving them precious hours every day. The line gets intolerably long as time continues, illustrated in the “Frequency of the Number of People in the Queue at Certain Times” graph. This shows a strong need to speed up the overall speed of the elevators since the overall length of the queue should stay consistent and not grow almost linearly longer as a function of time as the minutes increase, and the bullet point suggestions above should help with this.

The owner could evaluate the effectiveness of the speeding up of the elevator by seeing if his/her changes to the building transportation structure help provide effectively the same length of queue lines at all times of the day rather than allowing the workers to “clog” up the queue line and get frustrated waiting in longer lines as time goes on. The building owner could also ask workers to stagger the times they come in so that they do not “clog” the queue line up as they currently do by arriving at an exponential rate over the hour.

The owner of the building could also ask more people to take the stairs, as the chart titled “Frequency of the Number of People Who Walked to Different Floors” shows how often only around 20% of the building occupants elect to walk. The owner could incentivize people to walk more by making the stairwells nicer such as renovating and painting the stairs and adding air conditioning to the stairwells if it is not already present. The owner could also put motivational and encouraging signs around the building encouraging those who can walk to be more active and walk. This would help reduce the queues. The owner could then run the simulation and change the effects if the workers are more likely to walk (for example, if the owner could increase the number of people who walk to the fourth floor from 10% to 25%, the queue lines would be alleviated greatly.)

Critical Thinking Summary

Most Difficult Part of the Project

For me, the most difficult part of the project was dealing with the continuity of time. I had issues with how to advance time in a non-discrete manner, and I especially struggled in worrying about a stasis in time occurring if there was no one on the elevator or anyone waiting in the queue, for time at first would only advance if a worker stopped at a floor, advancing the time 0.5 minutes for the door to open and X amount of minutes depending on what floor the elevator was on and what floor it came from.

At first, time would not move forward if no one was to get in the elevator queue or elevator at a certain time, and the simulation would come to a halt. I eventually realized that I could set a boolean called `has_time_moved_forward` and set that to `True` if someone got off the elevator during

that current time. If someone did not get off the elevator however, `has_time_moved_forward` remained `False` and a simple if check that read

```
# Ensure time always moves forward to avoid stasis
if has_time_moved_forward == False:
    # Since -ln(0.999999) / 6 = 0.000001675, it is very, very, very unlikely
    # (re: practically impossible) that a person's initial term would skip over their time
    # at the very same time they arrived
    current_time = current_time + 0.000000166667
```

This if check ensured that time would always move forward at the smallest increment reasonably possible. Because the exponential rate of the times that workers arrive is equal to `-ln(u) / 6`, it was decided that the time would increment by just 0.000000166667 when the time had not naturally been forced to move forward, since a random uniform value of 0.999999 was yielded that small number. Because it is practically impossible that both a random uniform number bigger than this number is chosen *and* that someone just barely met the cutoff time of 0.000000166667 at the exact same time, this number was deemed sufficient for ensuring that no worker's boarding times of the elevator or of the queue were extended by an unreasonable amount of time. Even in the off chance that this *did* happen, the timing of their data would only be off by less than 0.000000166667, which is effectively trivial in the metrics and can be ignored.

This problem was thus more difficult than the discrete-time oriented project, where it was much easier to navigate the field of time and gather metrics. This project was thus enjoyable to learn more about dealing with the tricky issue of continuous variables, and this was a properly challenging obstacle to tackle.

Additionally, one frustratingly difficult part of the project for me was dealing with the waiting times for running 1280 simulations over and over again so that the correct metrics could be gathered that were sufficient enough per Tchebycheff's Theorem. I had to run the simulations on my laptop, which, lacking advanced processors and an abundance of RAM, required that I be very patient and wait out the results. I am also at fault for the longer run times, for I could have optimized the code so that it ran much quicker, but the hour or so wait times required to run the 1280 and gather all the data into separate files was a convenient way to gather all the needed data. After doing a brief cost-benefit analysis I came to realize that the time it would have taken to optimize the speed of the program would have defeated the purpose of not wanting to wait out the program run times in the first place -no overall time would have been saved, and it would have required more work on my part.

What I Would Do Differently To Improve this Project

If I were to redo this project again, there is one main thing I would definitely change: I would gather the data from the thousands of simulations I ran all at once, rather than running and rerunning the tests and only gathering one data metric each time.

To elaborate, I would run Tchebycheff's Theorem to on the 100 samples of the number of people who walked to different floors, find that I needed to run 1280 simulations, then run those 1280 simulations, wait a long time, and only gather data for the number of people who walked to different floors. Then I would run Tchebycheff's Theorem again, find that I needed to run 1280 simulations to gather enough data for the last boarding times data, and wait a long time again to gather all that data for that one very metric.

In hindsight, this was very inefficient and made my project literally take over four times as long as it otherwise should have. If I would have taken a more intelligent approach and gathered all the metrics at once in just 1280 simulations, I would have finished the project much faster and much more efficiently.

I thus learned an important lesson about planning out and strategizing my code and experimentation methods. I will more carefully carry out future experiments to avoid this time obstacle in future classes.

Another aspect I may consider changing in the future is the language. Python is an excellent language to quickly prototype on a few samples, but the nature of the slow interpreted language and inefficiencies of the language come out as I have to run more and more iterations. If I had to actually run thousands more of simulations, which many other projects may require, I would have needed a more nitty-gritty language that allowed me to manipulate the bits and efficiency of my whole program. A language such as C++ would have helped to alleviate this inefficiency and slowness and may have helped speed my overall progress more swiftly. I am not sure if the trade-off between the longer programming times and the faster runtimes would be worth it, however. But this is certainly something I may explore and try to change in the future.

What I Learned and How I Can Apply It to Academia/Industry

The most important lesson that I learned from this project is the importance of planning out and strategizing what I am going to code and what approach I am going to take before I go ahead and carry out all the simulations and work. Because I ended up running thousands of simulations to gather metrics when I only needed to run the simulation 1280 times, I ended up wasting a lot of my time and productivity.

This would be disastrous and could possibly get me fired in the industry if my boss found out how much time and productivity I was wasting by using such an inefficient approach to collecting all the needed data and requirements. In academia, were I to waste so much time by

collecting data so inefficiently, I may never finish carrying out the important research and work needed to find important discoveries and innovations.

I also learned a lot about scaling up programs. My program ran efficiently and quickly for one simulation and the necessary metrics were gathered. But I realized that running a program and gathering all the data for one simulation is very different than running a simulation thousands of times. I thus had to come up with a more efficient way to store all the collected data in a safe and reusable manner and had to optimize the simulation so that it could run faster and save time.

This is an important lesson for industry because industry applications often start small and then, when successful, the scale of the applications blossoms and augments very quickly. If the applications and methods of carrying out the programs does not scale along with the popularity of the application, then the application is no longer useful. Similarly, in academia, an experiment and program always needs to be able to be replicated and scaled so that the research and innovations could be carried out. This was an important lesson of scale and planning out the efficiency of the programs I write, whether those programs are for industry or for academia.

Similarly, I learned of the power of exponentiality and data growth. Although the workers gathered at an exponential distribution of only 6 people per minute, the queue for the elevator quickly became clogged up and long wait times immediately took effect. This is an important lesson in design and planning out how people will react and work with systems in place, and it is very relevant to industry and academia in the need to account for exponential and wild growth of variables.

Through using the Matplotlib and Seaborn visualization libraries, I learned the importance of visualizing and collecting data in a useful and visual manner. The distribution graphs that Seaborn was able to create were very efficient in organizing and summarizing all of the data and program collected into four solid and easy-to-read pictures. Being able to summarize data effectively and summarily into sleek and easy-to-understand charts is very important for academia in research papers and conveying my findings from my experiments and in industry in, for example, summarizing presentations to higher-ups about the benefits and work that I've been doing for their company.

This project was very creative, challenging, and fun. I learned many useful aspects of programming, simulation, and modeling through this project and this course, and I am very thankful to Dr. Elmaghraby for enabling this amazing learning experience to happen.

Appendix

[1] Calculator.net, "Standard Deviation Calculator," *Calculator.net: Free Online Calculators - Math, Health, Financial, Science*. [Online]. Available: https://www.calculator.net/standard-deviation-calculator.html. [Accessed: 17-Apr-2020].

[2] “The Kolmogorov-Smirnov Test of Normality,” *Kolmogorov-Smirnov Calculator (Test of Normality)*. [Online]. Available: https://www.socscistatistics.com/tests/kolmogorov/default.aspx. [Accessed: 17-Apr-2020].

[3] W3 Schools, *Python Data Types*. [Online]. Available: https://www.w3schools.com/python/python_datatypes.asp. [Accessed: 17-Apr-2020].

Source Code:

`Fox_Final_Project_Elevator.py`:

```
# Aaron Fox
# CECS 622-01
# Dr. Elmaghraby
# Final Project - Elevator Project

import math # for natural log
import random # For uniform random selection
import matplotlib.pyplot as plt # For plotting distributions of results
import seaborn as sns # For plotting desnity curve on plot

class Person:
    def __init__(self, arrival_time):
        self.floor = 1
        # random.choice uniformly selects one of the choices, which is fitting since each floor
        # to fo to had a 33% chance of being selected
        self.floor_to_go_to = random.choice([2, 3, 4])

        # This is the time the person had to wait to board and the time it takes to leave the elevator
        self.wait_time = 0

        # arrival_time is the time the person initially arrives at the ground elevator
        self.arrival_time = 0

        # departure_time is the time the person gets off the elevator
        self.departure_time = 0
        self.set_arrival_time(arrival_time)

        # Identify each person in order of who got off first
        self.id = 0

        self.time_boarded_elevator = 0

        self.attempted_to_walk = False

        self.walked = False

    def set_walked_true(self):
        self.walked = True

    def get_walked(self):
        return self.walked

    def set_attempted_to_walk_true(self):
        self.attempted_to_walk = True

    # Sets the arrival time of the person to the ground floor elevator (either line or no line)
    def set_arrival_time(self, arrival_time):
        self.arrival_time = arrival_time

    # Sets departure time of the person from the elevator (and also wait time since wait = departure - arrival)
    def set_departure_time(self, departure_time):
        self.departure_time = departure_time
        self.wait_time = self.departure_time - self.arrival_time
```

```
    def get_time_boarded_elevator(self):
        return self.time_boarded_elevator

    def set_boarding_time(self, boarding_time):
        self.time_boarded_elevator = boarding_time

    def get_floor(self):
        return self.floor_to_go_to

    def set_id(self, id):
        self.id = id

    def get_wait_time(self):
        return self.wait_time

    def __str__(self):
        return "[Person #" + str(self.id) + ", Arrival Time: " + str(self.arrival_time) + ", Departure Time: " + str(self.departure_time) + ", Wait Time: " + str(self.wait_time) + ", Time
                Boarded Lift: " + str(self.time_boarded_elevator) + ", Walker: " + str(self.walked) + "]\n"


    def __repr__(self):
        return "[Person #" + str(self.id) + ", Arrival Time: " + str(self.arrival_time) + ", Departure Time: " + str(self.departure_time) + ", Wait Time: " + str(self.wait_time) + "]"

def run_elevator_simulation(mean_interarrival_time):
    # Generate times for people to arrive at elevator until time reaches 61 (which is 9:00)
    times_to_arrive = []
    current_time = 0
    # lambda = mean interarrival rate = 1 / meant interarrival time
    # See: https://www.kellogg.northwestern.edu/faculty/weber/decs-430/Notes%20on%20the%20Poisson%20and%20exponential%20distributions.pdf
    mean_interarrival_rate = round(1 / mean_interarrival_time)
    while current_time <= 63: # Have this go to 61 to ensure everyone is accounted for in case last random var is very small (like 0.001, which would yield a high interarrival time)
        interarrival_time = -1 * math.log(random.random()) / mean_interarrival_rate
        current_time = current_time + interarrival_time
        times_to_arrive.append(current_time)

    # print(times_to_arrive)
    print("There are " + str(len(times_to_arrive)) + " people that can come to the elevator.")

    # Keeping track of the number of people that walk to each floor
    number_of_people_walked_to_2nd_floor = 0
    number_of_people_walked_to_3rd_floor = 0
    number_of_people_walked_to_4th_floor = 0

    # Keep track of last time a person boarded the elevator
    last_time_boarded_elevator = 0

    # Keep track of number of workers in line at specific times
    number_of_workers_in_line_at_8_30 = 0
    number_of_workers_in_line_at_8_45 = 0
    number_of_workers_in_line_at_9_00 = 0

    # Queue for elevator
    elevator_occupants = []
    queue = []
    elevator_floor = 1 # Ground floor of G == 1 here
    i = 0
    current_time = 0
    next_time = times_to_arrive[i]
    i = i + 1
    all_persons = []
    id_num = 1
    has_checked_830 = False
    has_checked_845 = False
    has_checked_900 = False
    # Run loop to iterate over needed hours 08:00-09:00+ (0 = 08:00, ..., 59 = 08:59, 60 = 9:00, etc.)
    # Continue running loop until all people have joined, the queue is empty, and all elevator_occupants have departed the lift
    while i < len(times_to_arrive) - 1 or len(queue) != 0 or len(elevator_occupants) != 0:
        # Use the variable has_time_moved_forward to check and make sure that time has moved forward. If not, increment by an amount
        has_time_moved_forward = False
        # print("Length of queue == " + str(len(queue)))
        # print("Length of elevator_occupants == " + str(len(elevator_occupants)))
        # print("Current Time: " + str(current_time))

        # Always start on ground floor (floor 0)
        elevator_floor = 1

        error_tolerance = 2
```

```
# Get number of people in queue at 8:30, 8:45, and 9:00
if not has_checked_830 and (current_time <= 30 + error_tolerance and current_time >= 30 - error_tolerance):
    has_checked_830 = True
    number_of_workers_in_line_at_8_30 = len(queue)
if not has_checked_845 and (current_time <= 45 + error_tolerance and current_time >= 45 - error_tolerance):
    has_checked_845 = True
    number_of_workers_in_line_at_8_45 = len(queue)
if not has_checked_900 and (current_time <= 60 + error_tolerance and current_time >= 60 - error_tolerance):
    has_checked_900 = True
    number_of_workers_in_line_at_9_00 = len(queue)


# Only add people to queue/elevator if their time is less than or equal to the current time
while i < len(times_to_arrive) - 1:
    # If time below 60, can still add to queue
    if next_time <= current_time:

        # If elevator full, append to queue
        if len(elevator_occupants) >= 12:
            queue.append(Person(next_time))
        elif elevator_floor == 1:
            # Else, append to elevator queue
            next_person = Person(next_time)
            next_person.set_boarding_time(current_time)
            elevator_occupants.append(next_person)
        else:
            # elevator is not full, but it is not on ground (shouldn't happen)
            queue.append(Person(next_time))

        # Get next time if in index
        i = i + 1
        next_time = times_to_arrive[i]
    else:
        # Otherwise next time hasn't come yet so break out of while loop
        break

# If elevator on ground floor (floor 1) which it should always be to start, then go up
if elevator_floor == 1:
    # Append necessary amount of persons from queue since on ground floor
    if len(elevator_occupants) < 12:
        # Add persons from queue to elevator
        while len(elevator_occupants) < 12 and len(queue) >= 1:
            person_to_add = queue[0]
            person_to_add.set_boarding_time(current_time)
            elevator_occupants.append(person_to_add)
            del queue[0]

    # People still left in queue here may choose to take the stairs. Check.
    for person in queue:
        # Make sure a person only contemplates walking once. Otherwise a 50% chance of walking twice
        # becomes a 75% chance and so on while a person remains in the queue
        if person.attempted_to_walk == False:
            if person.get_floor() == 2:
                if random.random() < 0.5:
                    number_of_people_walked_to_2nd_floor = number_of_people_walked_to_2nd_floor + 1
                    person.set_walked_true()
                    all_persons.append(person)
                    queue.remove(person)
                else:
                    person.set_attempted_to_walk_true()
            elif person.get_floor() == 3:
                if random.random() < 0.33:
                    number_of_people_walked_to_3rd_floor = number_of_people_walked_to_3rd_floor + 1
                    person.set_walked_true()
                    all_persons.append(person)
                    queue.remove(person)
                else:
                    person.set_attempted_to_walk_true()
            elif person.get_floor() == 4:
                if random.random() < 0.10:
                    number_of_people_walked_to_4th_floor = number_of_people_walked_to_4th_floor + 1
                    person.set_walked_true()
                    all_persons.append(person)
                    queue.remove(person)
                else:
                    person.set_attempted_to_walk_true()

    # Check if any occupant is going to second floor
```

```
elevator_floor = 2
someone_got_off_2nd = False
for occupant in elevator_occupants:
    if occupant.get_floor() == 2:
        someone_got_off_2nd = True
        occupant.set_id(id_num)
        id_num = id_num + 1
        occupant.set_departure_time(current_time)
        all_persons.append(occupant)
        elevator_occupants.remove(occupant)

if someone_got_off_2nd:
    has_time_moved_forward = True
    # Only option is to come from ground floor
    # Going from ground floor to second floor takes 1 minute
    current_time = current_time + 1.0
    # Opening elevator door takes 0.5 minutes
    current_time = current_time + 0.5

# Check if any occupant is going to third floor
elevator_floor = 3
someone_got_off_3rd = False
for occupant in elevator_occupants:
    if occupant.get_floor() == 3:
        someone_got_off_3rd = True
        occupant.set_id(id_num)
        id_num = id_num + 1
        occupant.set_departure_time(current_time)
        all_persons.append(occupant)
        elevator_occupants.remove(occupant)

# See where elevator came from
# From second floor
if someone_got_off_3rd:
    has_time_moved_forward = True
    if someone_got_off_2nd:
        # Coming from second floor
        # Going from second floor to third floor takes 0.5 minutes
        current_time = current_time + 0.5
        # Opening elevator door takes 0.5 minutes
        current_time = current_time + 0.5
    else:
        # Coming from Ground floor
        # Going from ground floor to third floor takes 0.5 minutes
        current_time = current_time + 1.5
        # Opening elevator door takes 0.5 minutes
        current_time = current_time + 0.5

# Check if any occupant is going to fourth floor
elevator_floor = 4
someone_got_off_4th = False
for occupant in elevator_occupants:
    if occupant.get_floor() == 4:
        occupant.set_id(id_num)
        id_num = id_num + 1
        all_persons.append(occupant)
        occupant.set_departure_time(current_time)
        elevator_occupants.remove(occupant)
        someone_got_off_4th = True

if someone_got_off_4th:
    has_time_moved_forward = True
    if someone_got_off_3rd:
        # Came from third floor
        # Going from third floor to fourth floor takes 0.5 minutes
        current_time = current_time + 0.5
        # Opening elevator door takes 0.5 minutes
        current_time = current_time + 0.5
    elif someone_got_off_2nd:
        # Came from 2nd floor
        # Going from second floor to fourth floor takes 0.75 minutes
        current_time = current_time + 0.75
        # Opening elevator door takes 0.5 minutes
        current_time = current_time + 0.5
    else:
        # Came from ground floor
        # Going from ground floor to fourth floor takes 1.75 minutes
        current_time = current_time + 1.75
```

```
                # Opening elevator door takes 0.5 minutes
                current_time = current_time + 0.5

        # Ensure time always moves forward to avoid stasis
        if has_time_moved_forward == False:
            # Since -ln(0.999999) / 6 = 0.000001675, it is very, very, very unlikely
            # (re: practically impossible) that a person's initial term would skip over their time
            # at the very same time they arrived
            current_time = current_time + 0.000000166667


    # Find last time boarded elevator
    # print("All Persons: " + str(all_persons))
    # Write data to text file
    # In calculating the average wait time of workers, we ignore walkers which obviously have a wait time of 0
    wait_time_sum = 0
    num_took_elevator = 0
    f = open(r'C:\Users\aaron\Classes_11th_Semester\CECS 622\CECS-622-Simulation-and-Modeling\Final Project\data_results.txt', "w")
    for person in all_persons:
        f.write(str(person))
        if person.get_walked() == False:
            num_took_elevator = num_took_elevator + 1
            wait_time_sum = wait_time_sum + person.get_wait_time()
        if last_time_boarded_elevator < person.get_time_boarded_elevator():
            last_time_boarded_elevator = person.get_time_boarded_elevator()

    f.write("Total Persons walked/took lift: " + str(len(all_persons)) + "\n")
    f.write("Time last person boarded elevator: " + str(last_time_boarded_elevator) + "\n")
    f.write("There were " + str(number_of_workers_in_line_at_8_30) +" people in line at 8:30\n")
    f.write("There were " + str(number_of_workers_in_line_at_8_45) +" people in line at 8:45\n")
    f.write("There were " + str(number_of_workers_in_line_at_9_00) +" people in line at 9:00\n")
    f.write("A total of " + str(number_of_people_walked_to_2nd_floor + number_of_people_walked_to_3rd_floor + number_of_people_walked_to_4th_floor) + " people
                walked.\n")
    f.write("People that walked to the second floor: " + str(number_of_people_walked_to_2nd_floor) + "\n")
    f.write("People that walked to the third floor: " + str(number_of_people_walked_to_3rd_floor) + "\n")
    f.write("People that walked to the fourth floor: " + str(number_of_people_walked_to_4th_floor) + "\n")
    f.close()
    # Append wait times to file to get average number of wait times
    # wait_time_file = open(r"C:\Users\aaron\Classes_11th_Semester\CECS 622\CECS-622-Simulation-and-Modeling\Final Project\average_wait_times.txt", "a")
    # wait_time_file.write(str(round((wait_time_sum / num_took_elevator), 5)) + ", ")
    # wait_time_file.close()
    # print("Wait Time: " + str(wait_time_sum))
    # print("num_took_elevator: " + str(num_took_elevator))
    # print("Average Wait Time: " + str(round((wait_time_sum / num_took_elevator), 5)))

    # Append number of people that came to each floor
    # num_2nd_floor = open(r"C:\Users\aaron\Classes_11th_Semester\CECS 622\CECS-622-Simulation-and-Modeling\Final Project\number_of_2nd.txt", "a")
    # num_3rd_floor = open(r"C:\Users\aaron\Classes_11th_Semester\CECS 622\CECS-622-Simulation-and-Modeling\Final Project\number_of_3rd.txt", "a")
    # num_4th_floor = open(r"C:\Users\aaron\Classes_11th_Semester\CECS 622\CECS-622-Simulation-and-Modeling\Final Project\number_of_4th.txt", "a")
    # num_2nd_floor.write(str(number_of_people_walked_to_2nd_floor) + ", ")
    # num_3rd_floor.write(str(number_of_people_walked_to_3rd_floor) + ", ")
    # num_4th_floor.write(str(number_of_people_walked_to_4th_floor) + ", ")
    # num_2nd_floor.close()
    # num_3rd_floor.close()
    # num_4th_floor.close()

    # Append times of the last boarding of the elevator
    # last_boarded_lift_file = open(r"C:\Users\aaron\Classes_11th_Semester\CECS 622\CECS-622-Simulation-and-Modeling\Final Project\last_board_lift_times.txt", "a")
    # last_boarded_lift_file.write(str(round(last_time_boarded_elevator, 5)) + ", ")
    # last_boarded_lift_file.close()

    # Append number of workers in queue at 08:30, 08:45, and 09:00
    num_830 = open(r"C:\Users\aaron\Classes_11th_Semester\CECS 622\CECS-622-Simulation-and-Modeling\Final Project\number_at_830.txt", "a")
    num_845 = open(r"C:\Users\aaron\Classes_11th_Semester\CECS 622\CECS-622-Simulation-and-Modeling\Final Project\number_at_845.txt", "a")
    num_900 = open(r"C:\Users\aaron\Classes_11th_Semester\CECS 622\CECS-622-Simulation-and-Modeling\Final Project\number_at_900.txt", "a")
    num_830.write(str(number_of_workers_in_line_at_8_30) + ", ")
    num_845.write(str(number_of_workers_in_line_at_8_45) + ", ")
    num_900.write(str(number_of_workers_in_line_at_9_00) + ", ")
    num_830.close()
    num_845.close()
    num_900.close()

if __name__ == "__main__":
    mean_interarrival_time = 0.1667
    for i in range(1):
        if i % 100 == 0:
            print("At iteration " + str(i))
        run_elevator_simulation(mean_interarrival_time)
```

```
# To be conservative, Tchebycheff's Theorem was used and determined that 320 samples were needed for 99% accuracy for average wait times
average_wait_time_results = [37.69784, 32.55458, 32.70936, 38.18702, 32.32497, 36.67784, 30.98008, 35.06011, 37.59831, 37.99863, 34.85578, 35.56183, 35.33353,
        35.03031, 36.69987, 38.3433, 31.6302, 33.55805, 37.68226, 41.01992, 28.04954, 34.16085, 34.33275, 41.02225, 34.95344, 37.68081, 30.55433, 33.94046,
        35.35801, 36.88835, 30.78619, 33.25131, 33.41697, 39.81182, 33.2769, 41.71673, 36.30373, 31.44374, 39.57568, 34.17377, 30.74996, 32.97349, 30.55312,
        31.45556, 39.85002, 34.23515, 34.32149, 31.44043, 30.71019, 32.53993, 31.37469, 37.12973, 33.63051, 33.54653, 39.63797, 35.13612, 40.39652, 37.45827,
        34.1835, 30.04749, 34.17886, 38.08287, 33.72079, 39.5865, 27.79078, 38.28525, 37.72699, 28.44639, 26.29424, 36.54732, 38.15131, 34.24733, 39.32666,
        36.58709, 36.0721, 36.0165, 35.52537, 32.98934, 32.01651, 41.28739, 32.31866, 34.32411, 34.08506, 37.11857, 33.66496, 31.22872, 29.38683, 31.8012,
        31.17452, 33.0534, 32.68437, 33.49614, 30.60152, 28.69215, 32.75517, 33.06797, 30.38313, 36.65658, 36.75022, 35.48525, 40.89122, 32.20408, 38.81221,
        35.6545, 31.66875, 32.04669, 35.98412, 35.69371, 35.82078, 34.70716, 35.25641, 30.86362, 32.82198, 34.68656, 38.94752, 36.91502, 37.63518, 30.87216,
        36.26022, 30.55183, 29.85325, 37.27147, 38.65306, 31.81061, 37.98331, 32.69509, 25.5561, 32.05256, 33.51682, 38.75918, 39.17035, 33.26124, 36.47547,
        33.11978, 31.4456, 36.28063, 38.54513, 30.03664, 39.4103, 36.78709, 34.20809, 34.84916, 36.80889, 38.7479, 34.52008, 31.36796, 37.24959, 37.40714,
        36.84317, 36.27482, 38.12926, 34.46907, 39.62099, 40.32065, 36.81044, 27.8274, 29.23096, 40.07716, 38.23005,
            38.84036, 39.26085, 36.01429, 30.54256, 36.14818, 36.87762, 35.12435, 37.212, 36.30107, 34.75462, 30.81757, 32.25398, 37.22452, 38.10595,
        34.12186, 37.14772, 36.93653, 40.04509, 34.44492, 36.18636, 32.71812, 36.84178, 40.61598, 38.77731, 38.5483, 33.19771, 33.71686, 36.48339, 30.97285,
        32.67994, 30.96189, 41.72056, 34.43247, 39.28281, 33.55376, 38.43496, 32.35117, 38.79696, 34.29055, 32.62633, 33.26932, 32.13996, 40.33471, 34.99872,
        36.89889, 35.40789, 35.73184, 37.04301, 37.01385, 40.56488, 30.65669, 31.09672, 37.1861, 32.2259, 32.09812, 36.93167, 38.99909, 37.23839, 39.30417,
        32.13749, 32.9279, 35.89427, 34.03916, 35.29793, 38.08172, 37.9934, 33.49494, 30.34859, 34.96082, 32.42927, 37.6059, 35.50058, 33.72124, 40.69547,
        29.46907, 32.69377, 38.90541, 36.10786, 41.68938, 31.62526, 36.86235, 33.92978, 29.30247, 31.57245, 42.08096, 36.88378, 35.09191, 36.78924, 37.11551,
        29.86197, 35.17959, 40.54262, 34.884, 36.93443, 36.1238, 37.7952, 39.03348, 43.20849, 31.86124, 39.36549, 34.50207, 34.27875, 31.61318, 29.7386,
        37.72204, 39.94726, 31.74289, 33.08104, 35.44814, 36.12312, 31.0392, 36.46662, 35.62474, 32.3224, 33.41869, 36.16184, 41.80312, 37.56781, 41.19712,
        36.34142, 36.04591, 33.15978, 33.33856, 39.60246, 37.1032, 34.75124, 42.18724, 36.24174, 28.48527, 29.46314, 35.87113, 35.69562, 35.01932, 36.84336,
        31.53006, 32.00403, 27.25476, 36.49905, 37.66352, 33.66475, 35.99109, 36.59605, 34.70702, 39.69591, 28.46675, 35.66402, 35.70068, 34.91797, 36.56838,
        32.91488, 34.40905, 32.77355, 32.95283, 37.38664, 35.48522, 30.68582, 42.92839, 36.82212, 34.47093, 33.32292, 33.27738]

# Plotting histogram and density curve of average wait times
# sns.distplot(average_wait_time_results, hist=True, color="dodgerblue", label="Compact")
# plt.gca().set(title='Average Wait Times Frequency', ylabel='Frequency', xlabel='Average Wait Time')
# plt.show()

# To be conservative, Tchebycheff's Theorem was used and determined that 1280 samples were needed for 99% accuracy for the numbers of people that walked to the 2nd,
        3rd, and 4th floors
number_of_2nd_results = [36, 35, 38, 34, 48, 25, 31, 28, 51, 28, 32, 33, 45, 28, 38, 41, 34, 21, 46, 43, 29, 36, 39, 39, 38, 34, 39, 42, 42, 36, 43, 33, 31, 48, 28, 45, 36, 30, 43,
        38, 40, 45, 31, 37, 36, 33, 32, 32, 29, 42, 25, 42, 45, 23, 36, 40, 30, 42, 35, 46, 35, 34, 34, 42, 26, 34, 35, 29, 36, 30, 35, 35, 32, 37, 25, 36, 38, 46, 47, 29, 25, 38,
        29, 35, 36, 39, 43, 27, 39, 29, 34, 35, 33, 33, 36, 36, 34, 27, 33, 39, 29, 45, 49, 36, 28, 44, 45, 31, 38, 33, 34, 31, 48, 44, 33, 33, 32, 35, 27, 35, 39, 33, 35, 40, 32,
        43, 40, 36, 29, 36, 34, 35, 37, 35, 44, 36, 29, 26, 31, 34, 38, 38, 44, 32, 27, 42, 32, 39, 31, 27, 39, 36, 37, 34, 30, 17, 40, 38, 40, 32, 39, 40, 30, 33, 29, 43, 51, 38,
        28, 39, 52, 37, 35, 45, 25, 36, 21, 28, 37, 39, 41, 50, 34, 24, 36, 32, 30, 35, 34, 18, 33, 32, 37, 40, 50, 37, 37, 29, 38, 28, 30, 43, 39, 38, 31, 54, 34, 35, 33, 39, 32,
        44, 34, 35, 41, 46, 29, 41, 39, 34, 42, 37, 34, 59, 43, 42, 26, 47, 29, 30, 23, 32, 33, 47, 36, 39, 46, 33, 25, 38, 36, 30, 26, 37, 40, 24, 34, 32, 43, 39, 38, 26, 33, 35,
        44, 23, 36, 38, 32, 36, 27, 33, 44, 37, 47, 41, 35, 34, 41, 46, 29, 44, 38, 49, 33, 26, 34, 29, 44, 27, 32, 33, 32, 25, 35, 55, 39, 34, 33, 33, 41, 41, 41, 22, 54, 37, 26,
        35, 33, 36, 42, 30, 30, 35, 42, 31, 40, 31, 42, 34, 50, 44, 32, 39, 34, 38, 52, 26, 32, 41, 40, 29, 34, 36, 34, 36, 44, 46, 48, 40, 29, 31, 44, 38, 27, 32, 38, 42, 36, 32,
        30, 40, 36, 45, 37, 36, 36, 32, 26, 29, 42, 30, 31, 39, 33, 32, 30, 44, 43, 45, 41, 34, 38, 33, 33, 39, 30, 33, 42, 34, 45, 33, 43, 40, 33, 40, 44, 40, 34, 38, 37, 45, 36,
        26, 40, 34, 44, 45, 32, 37, 41, 37, 51, 29, 60, 25, 33, 32, 29, 52, 37, 36, 31, 36, 41, 34, 32, 28, 49, 43, 37, 37, 37, 32, 35, 39, 29, 39, 48, 40, 32, 36, 40, 37, 36, 32,
        41, 35, 33, 27, 38, 33, 44, 41, 23, 36, 35, 33, 37, 26, 31, 52, 30, 35, 38, 33, 36, 40, 37, 34, 42, 37, 41, 34, 39, 32, 43, 23, 38, 37, 40, 31, 43, 34, 28, 44, 31, 29, 35,
        29, 24, 42, 46, 31, 44, 41, 40, 29, 39, 33, 38, 28, 50, 38, 48, 41, 35, 30, 26, 34, 38, 45, 47, 30, 37, 43, 36, 30, 44, 35, 35, 25, 26, 37, 36, 35, 33, 30, 27, 30, 37, 34,
        44, 30, 35, 47, 31, 38, 39, 33, 49, 33, 40, 28, 40, 36, 49, 41, 42, 28, 39, 28, 35, 27, 37, 38, 39, 27, 23, 35, 28, 29, 40, 40, 41, 49, 31, 33, 43, 33, 32, 40, 47, 38, 33,
        36, 39, 31, 37, 32, 38, 37, 57, 34, 37, 30, 28, 23, 36, 30, 33, 28, 29, 36, 31, 23, 33, 37, 39, 41, 33, 36, 27, 29, 36, 27, 35, 38, 46, 37, 39, 35, 34, 32, 36, 41, 27, 29,
        31, 30, 29, 32, 49, 28, 41, 33, 23, 22, 38, 33, 29, 37, 39, 42, 36, 37, 28, 39, 28, 34, 31, 35, 41, 29, 30, 25, 39, 44, 37, 41, 20, 33, 32, 43, 47, 37, 45, 29, 36, 34, 30,
        41, 39, 34, 44, 41, 30, 43, 33, 50, 30, 37, 25, 33, 37, 30, 31, 52, 31, 32, 31, 37, 31, 37, 43, 35, 36, 40, 34, 28, 34, 34, 38, 32, 46, 46, 32, 27, 30, 37, 34, 36, 39, 34,
        34, 32, 31, 30, 20, 37, 40, 27, 36, 48, 34, 42, 40, 34, 34, 33, 29, 35, 38, 43, 31, 50, 44, 32, 33, 27, 33, 32, 44, 28, 24, 39, 23, 41, 30, 52, 42, 46, 42, 43, 46, 46, 26,
        38, 27, 46, 40, 41, 36, 36, 34, 38, 32, 41, 42, 35, 40, 37, 40, 33, 30, 43, 31, 36, 28, 47, 24, 35, 41, 45, 39, 35, 34, 32, 37, 42, 38, 41, 41, 29, 28, 33, 47, 28, 39, 28,
        42, 42, 36, 42, 40, 31, 34, 38, 34, 26, 33, 37, 32, 40, 32, 33, 37, 35, 36, 36, 39, 41, 29, 41, 38, 25, 45, 31, 33, 42, 30, 24, 43, 25, 29, 38, 30, 44, 36, 38, 37, 38, 31,
        43, 42, 35, 23, 42, 24, 32, 33, 43, 35, 40, 27, 34, 38, 41, 40, 35, 52, 33, 45, 30, 25, 36, 38, 41, 36, 29, 29, 38, 28, 37, 39, 30, 33, 32, 42, 33, 29, 27, 46, 33, 41, 32,
        36, 41, 32, 30, 27, 30, 34, 43, 32, 40, 33, 49, 34, 32, 34, 32, 46, 37, 33, 41, 29, 41, 45, 43, 42, 26, 42, 32, 35, 40, 37, 43, 33, 30, 29, 46, 36, 36, 35, 39, 32, 37, 19,
        33, 38, 35, 57, 37, 35, 40, 28, 25, 48, 23, 40, 31, 35, 29, 36, 30, 41, 36, 38, 35, 41, 43, 48, 38, 37, 38, 31, 36, 33, 40, 34, 43, 35, 39, 45, 41, 42, 23, 33, 43, 32, 36,
        40, 26, 22, 38, 24, 37, 32, 28, 36, 38, 41, 41, 39, 28, 25, 39, 34, 47, 35, 28, 37, 43, 37, 32, 32, 39, 30, 34, 31, 29, 42, 29, 38, 32, 41, 24, 41, 29, 39, 27, 44, 34, 31,
        30, 40, 30, 41, 39, 31, 51, 46, 27, 31, 31, 29, 34, 29, 36, 41, 32, 38, 47, 40, 46, 37, 46, 44, 40, 33, 39, 49, 48, 35, 24, 31, 42, 33, 30, 36, 25, 31, 54, 38, 41, 27, 41,
        31, 39, 30, 38, 38, 32, 44, 28, 32, 30, 34, 50, 21, 51, 40, 38, 37, 48, 44, 32, 41, 39, 49, 30, 35, 35, 27, 39, 39, 37, 24, 25, 26, 37, 30, 33, 33, 28, 34, 41, 33, 28, 38,
        41, 32, 44, 37, 38, 39, 29, 35, 24, 26, 41, 31, 42, 41, 28, 35, 40, 33, 40, 20, 44, 39, 32, 40, 35, 29, 46, 42, 39, 29, 38, 38, 42, 34, 29, 34, 28, 31, 45, 42, 40, 35, 34,
        33, 44, 43, 31, 37, 39, 31, 34, 37, 38, 36, 46, 30, 29, 34, 32, 24, 27, 41, 39, 38, 33, 47, 46, 34, 27, 33, 30, 28, 37, 44, 50, 27, 30, 57, 23, 30, 38, 21, 36, 34, 28, 50,
        43, 31, 37, 35, 37, 44, 28, 41, 30, 48, 24, 42, 37, 26, 28, 41, 31, 33, 33, 44, 29, 43, 32, 41, 42, 37, 37, 33, 38, 40, 40, 48, 34, 32, 26, 39, 38, 40, 38, 34, 41, 42, 35,
        32, 56, 38, 31, 29, 31, 42, 40, 30, 35, 36, 37, 42, 39, 43, 32, 34, 32, 36, 41, 41, 39, 36, 24, 38, 35, 18, 30, 39, 29, 24, 41, 21, 35, 36, 36, 44, 47, 39, 33, 27, 37, 31,
        37, 34, 34, 43, 36, 46, 42, 34, 39, 45, 36, 29, 30, 37, 31, 28, 37, 38, 23, 38, 37, 42, 43, 38, 42, 41, 32, 29, 34, 35, 30, 27, 32, 33, 29, 30, 29]
number_of_3rd_results = [16, 23, 31, 20, 34, 24, 16, 27, 32, 25, 30, 21, 28, 26, 26, 22, 17, 27, 23, 23, 14, 28, 15, 23, 29, 24, 22, 29, 20, 23, 27, 26, 20, 14, 18, 29, 26, 25, 24,
        20, 23, 29, 35, 27, 25, 24, 23, 20, 25, 22, 27, 18, 18, 23, 25, 24, 28, 29, 16, 29, 34, 18, 28, 24, 25, 23, 24, 30, 33, 32, 13, 19, 19, 28, 21, 17, 17, 34, 31, 31, 22, 24,
        16, 27, 21, 26, 24, 19, 28, 22, 24, 16, 32, 25, 17, 25, 26, 24, 22, 30, 25, 31, 22, 22, 19, 26, 17, 23, 41, 27, 20, 28, 19, 19, 28, 39, 22, 29, 21, 22, 21, 34, 31, 29, 21,
        26, 18, 21, 29, 30, 21, 19, 34, 29, 31, 20, 28, 22, 18, 25, 15, 23, 21, 25, 23, 22, 23, 26, 18, 24, 23, 23, 20, 29, 17, 28, 37, 21, 24, 25, 30, 24, 24, 23, 17, 30, 22, 27,
        27, 19, 36, 27, 25, 29, 22, 25, 23, 21, 20, 24, 25, 25, 28, 34, 21, 24, 20, 22, 30, 30, 31, 29, 17, 25, 19, 23, 31, 21, 26, 24, 20, 24, 25, 13, 26, 28, 17, 25, 21, 30, 19,
        34, 27, 29, 27, 21, 28, 20, 25, 28, 30, 25, 22, 24, 31, 24, 20, 21, 22, 25, 13, 22, 17, 22, 29, 22, 26, 23, 25, 26, 17, 22, 22, 17, 22, 21, 20, 20, 32, 30, 22, 27, 22, 16,
        24, 29, 25, 25, 27, 28, 23, 20, 19, 31, 34, 27, 26, 21, 25, 36, 23, 24, 26, 17, 24, 18, 18, 23, 21, 19, 29, 28, 23, 27, 23, 24, 26, 34, 33, 25, 33, 26, 32, 18, 13, 23, 19,
        32, 28, 17, 23, 34, 26, 30, 11, 27, 24, 30, 22, 25, 22, 25, 30, 24, 24, 21, 28, 31, 18, 21, 33, 16, 19, 21, 32, 31, 21, 23, 35, 18, 18, 25, 26, 31, 19, 30, 21, 28, 21, 28,
        22, 26, 25, 31, 28, 22, 25, 17, 21, 20, 21, 25, 18, 33, 24, 19, 27, 23, 22, 27, 22, 19, 24, 22, 26, 32, 30, 30, 35, 17, 24, 22, 22, 25, 12, 25, 29, 32, 24, 30, 20, 24, 18,
        28, 21, 21, 21, 19, 20, 24, 24, 33, 21, 24, 26, 12, 27, 16, 19, 17, 15, 23, 26, 32, 27, 25, 25, 20, 28, 18, 21, 17, 16, 24, 25, 23, 17, 24, 26, 23, 20, 24, 22, 27, 39, 25,
        24, 26, 35, 29, 18, 19, 30, 26, 15, 26, 24, 18, 28, 28, 30, 39, 23, 24, 29, 28, 23, 24, 22, 21, 19, 32, 23, 28, 13, 26, 20, 20, 15, 17, 25, 19, 26, 24, 28, 21, 25, 20, 20,
        18, 23, 22, 29, 23, 20, 25, 27, 31, 27, 25, 20, 15, 20, 27, 34, 24, 28, 19, 26, 30, 28, 24, 19, 24, 20, 19, 31, 28, 29, 23, 24, 28, 20, 20, 19, 20, 17, 15, 24, 27, 30, 23,
        26, 20, 22, 39, 23, 19, 17, 18, 35, 21, 30, 34, 29, 20, 28, 23, 18, 29, 24, 22, 16, 21, 29, 21, 23, 14, 21, 20, 14, 23, 23, 30, 13, 28, 22, 32, 27, 18, 23, 18, 24, 25, 25,
        27, 24, 22, 30, 22, 22, 24, 15, 13, 22, 26, 26, 20, 20, 16, 19, 24, 20, 22, 20, 27, 32, 16, 24, 22, 20, 28, 20, 26, 22, 22, 26, 15, 30, 20, 19, 23, 14, 30, 30, 26, 29, 37,
        27, 19, 22, 25, 22, 23, 16, 29, 21, 26, 25, 21, 20, 19, 16, 18, 23, 33, 21, 16, 20, 31, 31, 21, 25, 24, 21, 19, 28, 27, 21, 23, 20, 13, 26, 18, 19, 26, 31, 19, 23, 24, 20,
        26, 22, 24, 33, 21, 28, 24, 13, 26, 24, 19, 21, 23, 20, 32, 16, 25, 26, 21, 28, 26, 22, 18, 28, 25, 20, 27, 19, 19, 28, 21, 26, 17, 33, 28, 23, 16, 27, 28, 21, 19, 24, 28,
```

26, 25, 26, 18, 23, 27, 26, 21, 26, 25, 29, 32, 22, 26, 16, 26, 22, 12, 28, 19, 27, 27, 25, 21, 33, 15, 23, 26, 19, 29, 26, 30, 23, 27, 16, 23, 22, 26, 25, 17, 17, 24, 22, 23, 8, 18, 22, 22, 27, 22, 26, 31, 22, 27, 23, 30, 24, 35, 29, 28, 24, 17, 28, 21, 20, 25, 18, 30, 19, 16, 19, 16, 29, 26, 18, 19, 17, 27, 28, 28, 28, 24, 23, 21, 35, 12, 26, 22, 30, 27, 22, 22, 30, 29, 32, 20, 32, 27, 37, 23, 26, 24, 25, 28, 24, 26, 20, 21, 22, 25, 21, 21, 21, 25, 28, 21, 19, 22, 17, 26, 27, 32, 24, 29, 23, 21, 28, 20, 24, 25, 20, 22, 28, 22, 17, 23, 24, 19, 28, 27, 19, 18, 24, 17, 21, 25, 34, 22, 30, 18, 20, 20, 31, 15, 24, 21, 15, 27, 19, 20, 24, 26, 24, 23, 21, 26, 25, 29, 30, 23, 25, 21, 22, 25, 19, 25, 20, 27, 19, 27, 21, 22, 25, 23, 26, 37, 22, 26, 28, 29, 20, 27, 23, 32, 27, 28, 27, 24, 17, 21, 31, 20, 24, 38, 17, 26, 31, 21, 26, 22, 28, 15, 29, 23, 17, 22, 28, 21, 33, 20, 16, 20, 28, 30, 27, 18, 25, 25, 25, 30, 30, 25, 27, 22, 16, 20, 25, 17, 27, 32, 22, 22, 26, 24, 21, 24, 24, 26, 23, 29, 27, 21, 27, 27, 26, 27, 15, 34, 26, 22, 26, 20, 24, 28, 30, 16, 31, 28, 28, 23, 30, 27, 32, 15, 20, 32, 20, 23, 30, 34, 25, 17, 23, 25, 25, 24, 26, 16, 24, 21, 25, 18, 22, 20, 22, 30, 20, 26, 26, 27, 28, 21, 16, 21, 31, 22, 26, 18, 26, 20, 22, 22, 19, 17, 28, 16, 26, 20, 32, 26, 21, 24, 19, 17, 18, 15, 28, 22, 30, 19, 23, 22, 37, 28, 20, 15, 26, 24, 26, 25, 18, 22, 27, 14, 16, 28, 21, 29, 18, 22, 21, 15, 19, 30, 28, 25, 27, 21, 25, 16, 24, 25, 25, 18, 28, 25, 24, 24, 13, 26, 18, 22, 20, 20, 16, 22, 24, 22, 17, 26, 21, 16, 21, 18, 24, 35, 25, 26, 28, 24, 23, 31, 20, 25, 27, 21, 22, 22, 21, 24, 22, 35, 21, 30, 25, 24, 26, 18, 23, 17, 18, 21, 24, 21, 26, 23, 23, 28, 21, 19, 17, 24, 17, 26, 31, 36, 25, 27, 24, 21, 27, 28, 26, 21, 24, 19, 30, 15, 25, 26, 27, 28, 28, 27, 21, 17, 22, 24, 25, 21, 23, 30, 30, 17, 26, 18, 22, 28, 28, 33, 25, 25, 29, 19, 31, 25, 23, 21, 20, 28, 16, 26, 23, 19, 27, 23, 15, 30, 21, 20, 36, 12, 33, 24, 20, 24, 23, 15, 11, 21, 17, 21, 31, 32, 36, 20, 36, 32, 35, 29, 27, 23, 33, 28, 24, 26, 26, 27, 30, 30, 22, 24, 24, 18, 28, 16, 23, 28, 27, 20, 14, 27, 23, 28, 18, 15, 21, 17, 20, 28, 26, 26, 21, 26, 21, 35, 23, 17, 19, 26, 20, 16, 18, 17, 25, 21, 35, 11, 20, 29, 26, 23, 22, 19, 24, 28, 22, 18, 24, 29, 33, 29, 32, 17, 18, 16, 22, 22, 31, 23, 23, 31, 25, 15, 29, 32, 30, 17, 29, 28, 25, 25, 25, 28, 29, 29, 23, 16, 17, 18, 22, 20, 20, 30, 28, 20]

number_of_4th_results = [10, 13, 9, 11, 4, 10, 8, 2, 4, 9, 8, 6, 7, 4, 3, 6, 6, 4, 4, 5, 8, 8, 5, 6, 8, 7, 12, 11, 9, 12, 9, 3, 8, 3, 8, 7, 8, 6, 10, 7, 9, 7, 5, 5, 6, 7, 6, 9, 6, 5, 8, 10, 8, 6, 12, 6, 5, 8, 12, 8, 5, 7, 4, 10, 3, 8, 11, 5, 7, 6, 3, 12, 5, 8, 8, 8, 6, 10, 5, 3, 5, 5, 8, 6, 10, 8, 9, 6, 10, 9, 3, 7, 11, 7, 3, 9, 3, 4, 14, 10, 7, 3, 9, 8, 7, 4, 4, 6, 9, 11, 8, 11, 9, 8, 6, 7, 6, 7, 8, 9, 5, 8, 6, 8, 11, 9, 9, 10, 12, 9, 3, 7, 3, 7, 12, 2, 10, 9, 13, 10, 7, 9, 6, 11, 3, 9, 12, 9, 6, 8, 6, 7, 4, 4, 3, 9, 7, 9, 7, 7, 7, 7, 7, 9, 6, 8, 5, 8, 5, 9, 13, 5, 5, 14, 7, 11, 6, 11, 6, 3, 9, 7, 7, 5, 7, 9, 7, 4, 4, 10, 10, 4, 8, 6, 9, 7, 7, 5, 12, 7, 9, 4, 11, 8, 10, 5, 5, 4, 6, 9, 9, 7, 2, 8, 6, 4, 8, 4, 7, 11, 6, 6, 6, 6, 2, 9, 5, 6, 5, 14, 8, 9, 9, 4, 7, 4, 6, 5, 8, 5, 10, 2, 8, 4, 4, 5, 4, 6, 8, 7, 6, 6, 9, 6, 5, 8, 7, 10, 9, 5, 7, 10, 6, 8, 5, 5, 4, 14, 2, 8, 7, 10, 7, 14, 7, 4, 5, 8, 8, 9, 10, 6, 9, 5, 6, 8, 7, 6, 8, 10, 6, 4, 4, 7, 5, 5, 6, 4, 12, 6, 5, 3, 3, 12, 8, 9, 10, 11, 10, 7, 8, 4, 5, 7, 10, 5, 5, 5, 9, 11, 9, 8, 8, 14, 2, 7, 11, 10, 7, 4, 6, 8, 8, 7, 3, 9, 5, 7, 4, 6, 10, 11, 7, 6, 4, 6, 4, 4, 7, 8, 9, 10, 6, 9, 7, 7, 8, 9, 4, 6, 8, 7, 9, 7, 16, 7, 5, 4, 6, 7, 11, 5, 12, 8, 4, 2, 7, 6, 7, 9, 4, 6, 7, 9, 4, 8, 5, 8, 3, 12, 9, 8, 7, 5, 9, 7, 9, 7, 5, 6, 5, 13, 6, 8, 5, 5, 7, 4, 10, 10, 8, 7, 9, 7, 3, 3, 4, 5, 5, 9, 4, 5, 9, 10, 9, 10, 9, 7, 5, 5, 8, 8, 12, 8, 3, 4, 8, 10, 9, 9, 7, 5, 2, 8, 6, 9, 5, 7, 14, 5, 6, 9, 11, 8, 10, 6, 6, 9, 9, 2, 4, 5, 8, 11, 8, 11, 11, 5, 4, 8, 10, 9, 3, 8, 3, 7, 5, 11, 10, 3, 11, 10, 5, 10, 4, 7, 4, 4, 18, 9, 6, 9, 10, 5, 9, 11, 4, 14, 6, 9, 7, 7, 8, 6, 7, 2, 10, 9, 7, 7, 7, 10, 9, 5, 14, 12, 8, 9, 10, 11, 5, 11, 5, 8, 4, 9, 6, 6, 9, 9, 7, 6, 6, 4, 9, 4, 8, 11, 5, 9, 8, 13, 8, 5, 9, 8, 7, 8, 8, 7, 5, 3, 4, 6, 3, 7, 10, 11, 12, 12, 4, 7, 12, 5, 7, 11, 7, 4, 8, 11, 9, 9, 4, 6, 4, 11, 9, 7, 2, 10, 7, 4, 9, 4, 7, 2, 7, 9, 6, 8, 4, 8, 5, 7, 6, 4, 11, 6, 8, 9, 6, 6, 5, 4, 7, 8, 7, 12, 9, 8, 5, 6, 8, 6, 6, 6, 9, 5, 10, 5, 5, 6, 10, 7, 4, 5, 9, 8, 9, 14, 4, 3, 7, 4, 7, 4, 12, 4, 3, 7, 3, 7, 5, 5, 3, 4, 5, 4, 5, 4, 9, 9, 8, 7, 9, 5, 9, 7, 5, 10, 14, 5, 4, 4, 5, 14, 4, 7, 11, 9, 11, 8, 4, 7, 10, 5, 8, 7, 6, 9, 5, 6, 9, 5, 5, 7, 5, 11, 3, 5, 8, 7, 10, 6, 9, 8, 5, 6, 3, 7, 9, 3, 7, 10, 6, 7, 3, 13, 8, 9, 4, 6, 7, 3, 8, 7, 7, 7, 8, 6, 2, 5, 5, 7, 7, 11, 7, 10, 7, 8, 6, 11, 7, 6, 2, 4, 3, 5, 11, 8, 9, 6, 8, 4, 8, 8, 7, 8, 4, 4, 5, 12, 5, 6, 9, 11, 7, 9, 4, 10, 6, 12, 5, 7, 4, 3, 6, 8, 3, 6, 9, 8, 7, 7, 5, 8, 5, 7, 7, 10, 6, 8, 9, 6, 10, 6, 7, 8, 8, 10, 6, 6, 4, 9, 5, 9, 6, 6, 11, 4, 6, 2, 7, 7, 5, 3, 9, 7, 5, 2, 4, 10, 13, 7, 9, 10, 14, 7, 10, 7, 9, 6, 6, 10, 4, 6, 14, 9, 7, 6, 7, 12, 4, 8, 4, 10, 3, 1, 11, 6, 13, 5, 8, 8, 6, 11, 5, 5, 9, 16, 11, 8, 4, 14, 6, 11, 4, 9, 8, 9, 9, 10, 2, 4, 8, 3, 9, 3, 7, 6, 4, 8, 7, 6, 6, 4, 11, 10, 10, 9, 10, 8, 1, 9, 10, 5, 4, 5, 5, 3, 14, 4, 4, 7, 10, 6, 5, 13, 8, 7, 5, 10, 5, 10, 5, 5, 8, 11, 9, 9, 10, 3, 6, 10, 6, 4, 6, 4, 4, 9, 6, 9, 9, 7, 9, 3, 6, 9, 8, 6, 10, 7, 6, 7, 5, 5, 9, 6, 1, 9, 5, 3, 5, 11, 6, 10, 6, 9, 6, 5, 6, 6, 9, 9, 7, 8, 8, 7, 5, 5, 1, 6, 5, 7, 4, 4, 4, 8, 5, 5, 9, 6, 10, 5, 3, 4, 5, 7, 3, 8, 5, 4, 9, 3, 10, 5, 10, 7, 9, 7, 6, 7, 9, 6, 9, 7, 6, 12, 6, 13, 5, 11, 4, 4, 8, 5, 8, 10, 2, 10, 6, 12, 7, 7, 4, 8, 8, 10, 10, 9, 9, 2, 8, 5, 11, 8, 9, 3, 4, 7, 7, 10, 11, 4, 5, 3, 10, 9, 4, 6, 6, 8, 6, 5, 11, 3, 9, 4, 9, 5, 8, 6, 13, 5, 5, 11, 4, 6, 5, 12, 8, 3, 5, 7, 8, 3, 4, 9, 8, 16, 10, 6, 9, 5, 5, 6, 9, 5, 2, 6, 8, 4, 7, 4, 8, 6, 8, 9, 10, 9, 3, 11, 0, 9, 9, 5, 5, 11, 9, 11, 5, 6, 6, 6, 10, 8, 16, 10, 6, 4, 9, 5, 6, 10, 4, 8, 3, 3, 5, 3, 5, 7, 8, 4, 9, 4, 6, 8, 7, 8, 10, 7, 10, 3, 3, 5, 6, 6, 7, 5, 2, 4, 4, 8, 3, 4, 8, 8, 10, 9, 11, 7, 6, 11, 4, 4, 7, 6, 6, 7, 8, 6, 3, 7, 2, 6, 12, 2, 4, 8, 7, 2, 6, 9, 11, 9, 5, 8, 5, 5, 7, 7, 5, 8, 3, 6, 6, 8, 7, 4, 7, 10, 7, 7, 5, 9, 10, 6, 11, 5, 7, 2, 12, 5, 8, 4, 4, 3, 7, 10, 8, 4, 6, 9, 4, 10, 5, 7, 7, 9, 8, 7, 9, 6, 12, 11, 4, 9, 10, 9, 3, 8, 8, 6, 6, 7, 8, 9, 9, 4, 4, 8, 6, 6, 5, 4, 12, 8, 4, 8, 5, 6, 5, 7, 6, 5, 12, 8, 2, 3, 8, 6, 7, 9, 9, 8, 9, 7, 5, 4, 7, 8, 12, 7, 14, 3, 3]

 # sns.distplot(number_of_2nd_results, hist=True, color="dodgerblue", label="2nd Floor")
 # sns.distplot(number_of_3rd_results, hist=True, color="green", label="3rd Floor")
 # sns.distplot(number_of_4th_results, hist=True, color="red", label="4th Floor")
 # plt.gca().set(title='Frequency of the Number of People Who Walked to Different Floors', ylabel='Frequency', xlabel='Number of People')
 # plt.legend()
 # plt.show()

 # To be conservative, Tchebycheff's Theorem was used and determined that 1280 samples were needed for 99% accuracy for the last board times people boarded elevators each day

last_board_lift_times = [132.54228, 128.29161, 123.30639, 133.55279, 127.81515, 126.15239, 125.06504, 120.01562, 141.10004, 125.86946, 124.27028, 136.90406, 131.01636, 127.92517, 127.06215, 124.77991, 134.55492, 126.49463, 131.5517, 133.53128, 130.52626, 134.32688, 127.46329, 120.71053, 119.0069, 130.62538, 128.93192, 152.8637, 120.21749, 128.02238, 134.06373, 129.40594, 130.31713, 134.62541, 131.53913, 130.09627, 140.084, 119.88924, 130.09547, 131.12949, 127.56093, 123.57965, 122.04183, 127.90073, 120.56341, 130.25598, 131.02458, 129.01062, 130.7712, 133.12506, 127.58829, 127.75867, 131.04141, 120.26171, 128.29427, 116.49566, 120.52662, 136.08137, 123.40316, 134.4089, 129.44575, 128.29153, 126.54659, 129.26973, 127.85915, 127.16846, 123.28229, 137.50635, 134.05811, 127.53543, 127.0003, 122.75796, 132.96931, 134.04217, 117.98505, 134.024, 124.76491, 141.98539, 126.03929, 124.74277, 131.50802, 128.18338, 132.26133, 123.39765, 133.00733, 130.04546, 128.89301, 123.44763, 133.73922, 130.27032, 129.90225, 133.54365, 117.81143, 123.61008, 120.51662, 131.42467, 112.13671, 117.52848, 124.31347, 133.75133, 119.99358, 123.41418, 130.39878, 115.79752, 128.26512, 139.36701, 130.6615, 138.24978, 122.29207, 127.00264, 129.56874, 116.48072, 137.59444, 126.95273, 121.24142, 131.11818, 129.87113, 129.58762, 127.84133, 127.75215, 124.07297, 124.0335, 128.38408, 131.78042, 131.53411, 124.61924, 139.07853, 132.43409, 132.56446, 123.60887, 129.76368, 134.02263, 127.23443, 135.13792, 137.10221, 126.55052, 124.42386, 127.6758, 136.81876, 117.26821, 119.55381, 126.56195, 129.83805, 136.83413, 135.09972, 123.08911, 130.31177, 125.17366, 124.41264, 127.96056, 122.53333, 128.09441, 141.75816, 131.14629, 124.54649, 130.69666, 119.87349, 130.55906, 131.55232, 133.65364, 131.45396, 124.22762, 134.79829, 134.5836, 137.29683, 129.53594, 124.79488, 127.3298, 120.07358, 141.41614, 130.17165, 130.41498, 130.09533, 127.82501, 137.0347, 135.37872, 116.64619, 127.82368, 123.712, 137.74138, 128.29026, 127.27725, 129.78261, 137.01478, 117.52287, 130.1766, 126.97577, 131.25488, 116.01967, 128.09588, 130.27672, 117.50055, 125.36925, 116.06304, 135.04085, 117.16078, 126.5545, 131.89656, 117.13484, 123.77353, 133.47891, 140.10641, 127.31397, 136.79286, 122.80199, 116.34999, 127.80429, 124.68202, 124.00218, 134.54366, 136.36777, 133.13172, 120.90551, 136.68343, 112.00913, 136.0864, 143.63083, 134.36981, 128.0224, 120.56913, 123.42187, 126.2839, 131.84384, 141.33953, 128.83866, 131.99531, 127.22425, 124.07628, 134.47119, 130.55923, 124.51307, 141.04176, 131.61082, 124.76637, 125.51093, 127.51374, 130.84898, 115.62745, 134.16989, 127.10205, 133.82106, 131.56037, 127.65198, 126.00269, 120.60575, 131.44119, 127.85963, 131.44037, 118.54811, 127.50229, 124.50859, 138.01925, 121.64538, 111.69584, 130.06907, 125.52414, 137.72636, 127.78173, 133.76994, 123.3226, 149.19159, 124.14923, 123.14911, 117.28614, 134.0481, 126.35161, 127.75923, 135.29011, 140.60253, 124.76771, 119.41753, 120.37509, 123.52847, 129.03247, 126.18853, 120.69849, 127.07982, 133.37821, 130.57959, 139.44034, 127.54221, 131.22163, 124.26538, 119.70724, 132.51608, 131.53045, 127.28048, 136.05328, 123.06871, 124.00601, 124.58775, 130.08153, 127.30263, 136.066, 135.27249, 123.63856, 126.56971, 117.94283, 127.77087, 119.14066, 133.60624, 127.51748, 116.6753, 117.05475, 128.26077, 141.00762, 127.07321, 124.77487, 117.37579, 121.55914, 130.90838, 124.52882, 151.9717, 136.76509, 129.81595, 127.26132, 117.66999, 131.7738, 137.2323, 126.48097, 130.60323, 130.02946, 130.86039, 113.5932, 138.37456, 131.38407, 116.86161, 131.09932, 113.38962, 122.66595, 136.09967, 134.78234, 130.57925, 135.49261, 124.102, 130.55396, 129.32602, 119.36287, 126.29478, 127.8847, 130.58138, 120.01022, 136.73242, 125.14933, 127.9449, 122.2544, 124.02081, 132.65601, 134.31777, 120.25735, 130.18781, 126.3676, 133.01199, 141.46017, 145.52076, 140.84367, 127.32473, 112.39025, 134.02467, 116.57342, 134.85477, 119.76948, 127.52408, 120.52481, 126.01339, 144.26561, 120.33648, 134.56377, 132.11593, 131.77987, 123.95917, 131.86751, 134.29418, 133.38606, 121.76308, 130.69388, 123.69008, 131.1568, 127.04531, 127.05855, 115.85172, 127.93888, 120.68938, 133.9521, 122.54464, 127.52753, 137.51769, 128.79681, 135.30435, 130.412, 130.65597, 127.12822, 135.40175, 124.32519, 133.40866, 122.753, 137.38163, 131.34982, 135.75623, 119.88544, 127.08904, 128.13969, 136.96268, 127.6577, 134.92067, 119.52513, 125.04756, 116.52404, 124.4064, 127.74277, 123.28356, 131.8384, 126.40306, 121.84793, 127.46645, 122.914, 127.78879, 124.89897,

126.75908, 120.00967, 141.51021, 145.08583, 127.27798, 130.60044, 126.23051, 123.44418, 134.03107, 134.80867, 121.01534, 130.51468, 133.15912,
134.59267, 116.51225, 127.50338, 134.5081, 124.54732, 127.86206, 137.84611, 113.84772, 131.01375, 121.05881, 132.15011, 126.04386, 126.31455,
124.86425, 140.36682, 117.73588, 132.62136, 124.5066, 133.84961, 128.83795, 153.9199, 129.62739, 120.51546, 123.56806, 124.89648, 123.79107,
127.26165, 136.562, 123.34404, 117.80923, 132.0211, 122.09218, 124.68793, 123.54128, 128.30358, 120.82747, 127.79475, 122.77322, 123.76058,
122.53024, 121.31865, 125.26886, 127.62877, 119.57066, 116.58644, 127.03782, 124.98305, 135.25617, 124.70565, 130.52914, 126.69076, 130.6707,
130.71749, 130.87223, 127.73975, 114.91786, 143.56988, 120.64717, 136.57153, 132.50709, 120.08061, 119.83528, 123.01614, 121.27013, 129.57087,
123.39513, 130.06286, 113.16396, 128.36587, 140.39862, 127.05627, 139.31157, 123.01051, 130.67007, 123.27053, 134.52701, 114.07995, 122.51995,
130.37033, 137.38486, 124.28725, 134.11244, 123.01904, 128.78216, 129.57498, 134.07066, 134.2628, 132.63762, 111.53728, 128.20623, 129.82059,
132.76097, 124.09357, 118.90366, 126.08432, 120.35009, 127.52012, 140.32854, 124.30901, 129.54844, 120.34761, 117.3044, 124.52207, 119.79727,
123.91937, 130.22325, 134.76878, 129.58495, 126.78381, 128.02734, 131.80019, 134.30048, 129.83651, 128.07679, 129.05543, 127.50277, 124.63815,
138.50168, 129.18967, 127.70615, 130.54596, 126.52269, 120.10145, 137.09323, 127.99291, 126.27201, 127.5469, 126.82499, 120.33146, 131.02219,
138.38244, 127.73154, 124.62522, 124.09889, 127.02213, 129.13224, 123.0657, 130.25466, 117.88051, 135.18497, 123.0985, 134.7732, 124.10308,
125.59531, 124.06075, 131.07701, 124.86854, 124.24642, 137.03478, 134.87895, 134.64446, 133.54617, 139.83927, 117.18415, 129.72874, 133.82916,
123.02437, 123.86156, 128.31075, 127.53045, 123.5025, 124.79877, 131.16897, 120.54841, 128.57272, 123.59901, 131.81247, 134.65809, 125.30829,
127.81912, 140.63433, 130.02175, 118.71614, 124.22247, 126.17436, 137.55072, 135.04711, 127.03132, 127.78462, 135.0836, 123.4184, 127.22667,
144.04826, 123.60499, 123.04053, 128.6098, 141.7838, 124.69789, 143.43674, 131.09119, 134.27292, 127.06719, 134.48933, 134.0234, 132.55661,
120.76331, 126.61818, 120.20754, 138.53348, 119.453, 123.27878, 134.05581, 138.73624, 125.6481, 130.25993, 130.44337, 123.26283, 134.87355,
126.03127, 136.0215, 129.96215, 126.62932, 134.61448, 120.8666, 117.81271, 125.57089, 120.77463, 117.02822, 141.04993, 120.27299, 145.56715,
134.11136, 130.73347, 136.44074, 144.61448, 123.02018, 113.00428, 137.04335, 123.58303, 116.68515, 128.03353, 133.86004, 121.3673, 119.25353,
126.25889, 129.76321, 120.96691, 131.59935, 126.45226, 126.35645, 125.29416, 122.78265, 127.56362, 124.00412, 130.5598, 144.33894, 132.78916,
126.04931, 117.25792, 122.92363, 116.0763, 117.87049, 131.08369, 129.01981, 127.58361, 130.23261, 131.42817, 121.00855, 135.18106, 127.00086,
125.52852, 128.48502, 139.09752, 136.62882, 129.64377, 126.05672, 124.54414, 127.78249, 124.00606, 136.37203, 130.08499, 116.93941, 144.23668,
121.34782, 128.20779, 125.34428, 126.87811, 132.25155, 136.68052, 141.30085, 141.35957, 130.8037, 137.65235, 115.27942, 133.47762, 126.19577,
136.71064, 131.47162, 137.58851, 126.66025, 117.26894, 130.11221, 127.0637, 130.02992, 121.04548, 127.26077, 134.21576, 134.9865, 118.51286,
127.0754, 129.17427, 126.93902, 131.00287, 133.80102, 126.59677, 131.82613, 118.40186, 129.88259, 144.3438, 141.25153, 131.92665, 119.30012,
124.26355, 127.13019, 135.06353, 131.53975, 121.03938, 129.41421, 126.00338, 124.51132, 116.64657, 127.77093, 131.72434, 126.0457, 130.27766,
131.32192, 121.02749, 128.37534, 135.29711, 126.51955, 130.8323, 124.89573, 118.4087, 133.34147, 138.77813, 124.84385, 117.78328, 129.60972,
141.54465, 130.53456, 140.31167, 140.79329, 127.29975, 127.50541, 126.2913, 140.53109, 144.61845, 141.03165, 131.26029, 127.04613, 127.7877,
124.55632, 130.36999, 128.2999, 137.23257, 136.80078, 124.33146, 127.04504, 120.78634, 127.56419, 145.75575, 133.88259, 124.02258, 138.54982,
124.36512, 133.26488, 122.92111, 126.98033, 137.04979, 124.7338, 130.0102, 126.62447, 124.75025, 126.07511, 129.07738, 126.85594, 121.3827,
126.68087, 132.0489, 135.06198, 138.51652, 131.75455, 130.54067, 134.7258, 125.45985, 125.76778, 123.7708, 128.56753, 124.18434, 127.5317, 122.50208,
117.55538, 127.6428, 123.13954, 124.05207, 127.30703, 128.16874, 127.02918, 134.45949, 128.09241, 127.54816, 131.8194, 132.54338, 128.40157,
129.87578, 121.33301, 128.37619, 130.87369, 131.02306, 130.60589, 123.22248, 126.56095, 118.89924, 119.80151, 131.28868, 121.22026, 130.192,
134.67895, 123.89639, 124.35044, 131.34577, 134.18052, 123.80634, 131.79483, 114.00598, 141.80021, 118.81893, 133.40675, 136.17309, 130.87933,
123.29492, 116.54457, 121.09633, 128.09793, 127.12083, 131.26577, 136.82751, 141.7592, 137.79202, 129.84833, 122.09547, 124.61046, 130.02064,
124.02425, 137.57971, 133.03446, 127.17447, 129.45285, 137.68595, 126.10363, 132.00798, 130.79346, 130.8598, 124.29503, 126.00638, 119.59556,
116.83984, 116.88835, 135.15805, 111.57968, 116.34251, 121.56708, 118.60174, 123.71509, 127.33291, 123.58571, 131.66312, 134.64439, 120.96554,
131.3466, 130.51609, 134.02736, 127.05196, 131.59706, 119.26508, 131.0942, 133.51097, 127.86953, 117.13083, 121.54106, 147.7508, 141.51878,
133.35659, 118.82753, 141.78992, 124.03161, 129.5925, 126.09263, 137.57323, 129.28533, 131.02212, 137.19469, 130.63527, 128.39098, 136.25004,
129.21339, 130.37172, 109.06655, 134.94299, 127.53264, 126.81389, 111.11779, 134.39515, 142.26065, 127.95856, 131.30111, 120.56592, 120.96227,
137.11478, 130.24746, 134.50568, 120.26354, 131.35702, 124.79213, 125.05936, 138.94089, 120.81219, 131.63406, 119.79745, 131.80065, 120.76105,
130.59255, 135.56778, 128.05147, 134.34786, 134.0647, 120.34455, 127.19482, 133.37012, 122.95788, 119.14222, 134.38906, 138.47439, 120.71785,
128.30103, 121.02206, 135.08215, 138.04035, 136.88857, 134.09421, 131.07632, 128.02108, 135.00707, 123.79684, 127.80875, 125.50604, 124.69913,
147.02118, 125.3151, 132.75672, 126.73189, 129.40695, 128.43812, 134.03012, 138.04335, 138.25644, 130.74872, 123.07718, 133.84958, 127.50926,
118.66574, 122.03915, 136.85147, 136.78771, 126.76198, 133.94149, 126.82539, 120.38363, 127.28196, 140.64251, 143.8299, 137.76149, 137.27188,
123.60757, 129.52506, 140.5182, 124.44258, 126.55425, 124.25921, 141.88432, 140.001, 130.79373, 118.00177, 117.21197, 124.35018, 127.18146,
132.56134, 124.54708, 135.25543, 123.13611, 135.13358, 126.31279, 131.51449, 133.57487, 121.35731, 136.33667, 136.35373, 122.76037, 116.00891,
133.57517, 130.07687, 128.87936, 141.66213, 120.55324, 140.69773, 124.0432, 133.57789, 134.8146, 131.08452, 129.64251, 127.55552, 113.50344,
134.52814, 125.21227, 135.06259, 126.32606, 109.02639, 134.04049, 124.32424, 126.12348, 126.56041, 127.14898, 119.2936, 129.45339, 116.55405,
123.82824, 129.10982, 130.81851, 130.50226, 130.86148, 131.56177, 128.05247, 130.16476, 129.8669, 127.35786, 126.6075, 119.26813, 118.81221,
124.82788, 126.73877, 112.73046, 124.0143, 125.81203, 122.01281, 132.52919, 112.65373, 133.36933, 133.6861, 131.2475, 133.00492, 129.21284,
120.35134, 127.29648, 138.56797, 138.00107, 130.358, 125.66011, 123.32867, 129.3869, 134.03929, 126.20495, 123.08408, 127.02662, 124.34441,
127.01601, 122.61052, 131.34162, 128.53799, 135.85992, 124.75021, 135.79495, 130.74732, 129.50232, 122.8058, 125.82053, 113.39791, 129.64542,
123.02809, 130.59147, 117.58851, 133.27837, 122.84928, 138.25256, 127.26398, 129.80494, 134.54196, 124.0077, 126.36288, 127.53502, 128.80282,
132.70988, 124.65092, 134.38933, 129.28626, 129.27899, 133.14336, 130.45498, 126.68037, 134.61301, 125.75044, 123.67471, 133.93833, 118.29594,
123.84819, 141.65555, 134.7768, 126.2317, 133.72743, 130.37877, 123.52503, 123.37026, 125.03771, 129.68296, 131.8496, 121.25599, 117.53953,
125.82294, 135.25955, 131.88085, 126.59946, 115.343, 131.24785, 131.04909, 130.58041, 134.00046, 118.56553, 136.509, 133.10447, 133.78892, 120.06426,
113.85356, 130.75578, 127.55636, 136.86238, 131.75813, 127.12307, 138.10987, 140.83558, 125.77645, 126.04755, 124.25478, 131.27173, 130.27164,
141.55002, 124.65715, 133.11007, 135.94585, 116.87316, 120.5245, 131.04395, 126.50195, 112.45488, 127.00603, 126.51168, 124.31117, 131.10994,
122.51512, 120.8649, 127.51209, 126.8143, 124.4527, 131.50937, 126.55845, 126.5175, 131.847, 137.71795, 133.53581, 123.15272, 123.27195, 134.21074,
126.22031, 130.2353, 133.04557, 131.26809, 130.0596, 120.50978, 123.72579, 115.80979, 123.41496, 123.94313, 117.93255, 122.5058, 109.51607,
125.41465, 135.28421, 126.65902, 126.52843, 127.94963, 127.58117, 128.57825, 116.2558, 127.52633, 120.66247, 126.78895, 122.5964, 116.55627,
122.79104, 116.83557, 128.72539, 140.3931, 124.29173, 116.19002, 137.09056, 137.85125, 123.40483, 120.2723, 130.19845, 134.02898, 134.80122,
127.34357, 128.24331, 135.17953, 132.55108, 130.50267, 125.40266, 124.39623, 126.79377, 134.17754, 135.17837, 124.31295, 125.15372, 135.2549,
119.24605, 130.35819, 130.49506, 131.09349, 133.50289, 132.18759, 126.68795, 122.87746, 129.05472, 128.56058, 130.54841, 127.63626, 134.39745,
127.17522, 137.58554, 122.65159, 134.06718, 106.59345, 121.04184, 123.30464, 122.07208, 127.79598, 128.42616, 123.72017, 130.28629, 127.59549,
134.5055]

 # Convert times to actual times (e.g. 0 = 08:00)
 # plt.xticks([100, 110, 120, 130, 140, 150, 160], ['09:40', '09:50', '10:00', '10:10', '10:20', '10:30', '10:40'])
 # sns.distplot(last_board_lift_times, hist=True, color="dodgerblue", label="Times")
 # plt.gca().set(title='Frequency of the Number of Last Boarding Times', ylabel='Frequency', xlabel='Times')
 # plt.show()

 # To be conservative, Tchebycheff's Theorem was used and determined that 1280 samples were needed for 99% accuracy for the number of people in the queue at 08:30, 08:45, and 09:00

number_830 = [63, 62, 71, 59, 69, 57, 52, 48, 85, 67, 54, 50, 70, 66, 60, 62, 44, 60, 82, 76, 73, 50, 56, 52, 62, 58, 45, 62, 61, 70, 71, 73, 67, 56, 62, 67, 50, 71, 46, 43, 68, 66, 44, 55, 32, 57, 65, 63, 60, 61, 47, 69, 79, 63, 75, 53, 61, 56, 53, 74, 64, 55, 62, 85, 66, 42, 57, 67, 63, 62, 59, 79, 73, 63, 59, 61, 81, 70, 83, 56, 60, 44, 57, 52, 55, 59, 47, 64, 72, 65, 59, 55, 61, 51, 54, 48, 56, 80, 67, 75, 60, 51, 61, 51, 63, 57, 70, 77, 74, 66, 57, 58, 56, 62, 78, 60, 73, 65, 59, 61, 68, 68, 66, 52, 77, 54, 34, 68, 33, 44, 67, 70, 61, 65, 47, 73, 60, 63, 69, 48, 65, 58, 63, 61, 70, 52, 58, 67, 66, 53, 53, 75, 73, 60, 56, 42, 73, 47, 49, 57, 56, 87, 63, 56, 56, 66, 67, 51, 59, 54, 66, 55, 84, 70, 52, 77, 61, 65, 53, 56, 63, 60, 55, 55, 69, 62, 58, 71, 67, 65, 57, 37, 61, 50, 36, 44, 55, 66, 61, 59, 64, 51, 43, 67, 73, 66, 52, 72, 68, 58, 62, 60, 50, 82, 60, 67, 69, 39, 83, 83, 73, 68, 83, 51, 60, 63, 80, 59, 71, 40, 51, 53, 54, 70, 44, 62, 48, 64, 53, 49, 58, 55, 70, 51, 47, 63, 50, 46, 50, 62, 62, 68, 58, 46, 59, 58, 84, 47, 41, 56, 49, 63, 68, 61, 80, 62, 68, 67, 41, 71, 57, 45, 68, 64, 56, 57, 55, 76, 85, 51, 56, 61, 75, 64, 58, 67, 50, 51, 64, 66, 55, 49, 75, 62, 69, 47, 65, 71, 55, 57, 69, 79, 68, 62, 60, 64, 74, 63, 62, 63, 86, 63, 52, 57, 56, 71, 69, 46, 85, 64, 53, 55, 81, 73, 67, 50, 65, 52, 60, 55, 54, 67, 85, 48, 70, 66, 71, 43, 31, 52, 65, 82, 63, 47, 48, 65, 49, 71, 70, 58, 72, 47, 63, 68, 60, 58, 75, 64, 58, 55, 66, 69, 54, 62, 45, 61, 50, 65, 59, 53, 60, 50, 49, 66, 69, 57, 63, 46, 55, 64, 60, 51, 39, 68, 76, 65, 55, 55, 59, 69, 51, 55, 56, 55, 81, 69, 62, 49, 58, 43, 50, 87, 56, 70, 79, 76, 51, 65, 65, 66, 66, 64, 42, 61, 60, 57, 65, 53, 49, 58, 70, 75, 84, 68, 67, 49, 48, 66, 42, 70, 64, 68, 60, 65, 64, 52, 59, 63, 63, 61, 52, 59, 43, 58, 48, 70, 56, 66, 64, 63, 57, 53, 67, 46, 74, 63, 63, 66, 69, 58, 58, 68, 45, 62, 60, 61, 70, 51, 80, 49, 53, 51, 44, 56, 75, 52, 54, 65, 62, 59, 54, 52, 62, 52, 52, 61, 69, 50, 38, 57, 57, 45, 72, 71, 57, 46, 59, 73, 73, 49, 58, 65, 59, 56, 58, 55, 69, 40, 49, 80, 64, 75, 62, 66, 41, 53, 47, 62, 54, 51, 59, 49, 50, 52, 49, 64, 60, 68, 57, 61, 65, 63, 54, 45, 61, 48, 60, 49, 54, 60, 67, 92, 58, 56, 56, 57, 45, 50, 69, 74, 55, 44, 58, 56, 71, 81, 55, 44, 41, 72, 61, 57, 66, 65, 58, 75, 61, 61, 51, 59, 76, 69, 81, 46, 54, 63, 83, 65, 66, 49, 72, 61, 81, 76, 69, 63, 67, 71, 60, 56, 53, 67, 44, 71, 59, 47, 47, 69, 59, 65, 45, 51, 70, 65, 74, 58, 47, 71, 43, 59, 61, 60, 71, 57, 43, 61, 45, 56, 61, 49, 42, 62, 69, 63, 58, 72, 77, 59, 70, 76, 54, 72, 70, 64, 57, 70, 61, 58, 74, 40, 61, 62, 63, 48, 62, 65, 58, 69, 79, 71, 62, 69, 57, 77, 48, 65, 45, 55, 63, 58, 67, 61, 61, 63, 57, 67, 68, 58, 93, 69, 68, 56, 50, 55, 68, 62, 65, 71, 70, 56, 80, 70, 55, 65, 66, 52, 76, 59, 75, 41, 49, 72, 54, 47, 48, 63, 70, 71, 52, 70, 45, 60, 51, 58, 60, 75, 76, 52, 58, 81, 61, 63, 57, 84, 54, 58, 51, 54, 62, 57, 86, 63, 69, 60, 72, 58, 56, 60, 43, 68, 59, 59, 64, 66, 68, 53, 50, 59, 40, 50, 49, 73, 73, 69, 57, 46, 59, 48, 57, 64, 55, 73, 55, 39, 65, 63, 54, 60, 60, 46, 69, 62, 42, 60, 54, 69, 41, 81, 81, 68, 60, 48, 53, 55, 58, 58, 59, 68, 67, 69, 72, 77, 48, 53, 57, 69, 46, 66, 81, 78, 59, 62, 66, 61, 56, 61, 66, 83, 61, 63, 58, 46, 64, 73, 75, 79, 56, 54, 56, 75, 55, 49, 60, 63, 56, 51, 58, 57, 60, 76, 62, 48, 59, 67, 52, 43, 67, 77, 72, 49, 60, 60, 64, 60, 71, 54, 65, 46, 81, 68, 65, 58, 66, 75, 66, 51, 61, 73, 53, 68, 63, 49, 64, 86, 78, 63, 58, 66, 83, 56, 66, 49, 46, 61, 55, 56, 66, 57, 49, 74, 70, 81, 55, 44, 67, 56, 60, 71, 54, 78, 52, 63, 70, 71, 59, 61, 47, 55, 61, 64, 52, 50, 52, 68, 72, 51, 89, 64, 51, 84, 74, 55, 67, 64, 50, 59, 63, 55, 67, 56, 63, 61, 66, 63, 68, 58, 59, 63, 56, 71, 54, 53, 55, 57, 54, 66, 58, 64, 51, 72, 55, 61, 41, 64, 61, 74, 79, 38, 71, 86, 64, 57, 65, 52, 70, 46, 62, 57, 61, 47, 52, 54, 55, 61, 64, 67, 46, 70, 55, 67, 63, 71, 59, 55, 61, 61, 70, 42, 59, 72, 52, 48, 56, 58, 79, 60, 73, 83, 73, 51, 73, 68, 73, 89, 61, 66, 64, 52, 46, 53, 58, 53, 75, 65, 48, 48, 54, 79, 68, 58, 59, 76, 61, 53, 72, 57, 60, 45, 63, 55, 85, 54, 64, 53, 72, 70, 68, 77, 57, 65, 69, 65, 52, 60, 64, 77, 55, 73, 69, 51, 82, 70, 51, 53, 59, 65, 75, 66, 65, 54, 45, 52, 57, 51, 56, 67, 53, 51, 67, 60, 60, 62, 46, 50, 78, 63, 56, 64, 70, 62, 46, 44, 65, 53, 62, 60, 57, 56, 70, 69, 61, 63, 54, 65, 47, 46, 49, 57, 63, 70, 60, 44, 60, 72, 69, 60, 61, 56, 67, 55, 61, 48, 69, 38, 54, 65, 56, 71, 51, 64, 68, 49, 42, 63, 61, 67, 63, 64, 78, 54, 46, 66, 46, 67, 66, 65, 75, 50, 56, 67, 61, 64, 45, 55, 54, 66, 52, 61, 64, 66, 70, 59, 51, 54, 67, 66, 52, 56, 56, 70, 63, 49, 79, 69, 56, 71, 58, 56, 59, 50, 82, 61, 76, 74, 54, 63, 61, 61, 73, 61, 50, 52, 75, 71, 59, 88, 63, 53, 59, 56, 46, 68, 80, 58, 47, 64, 51, 61, 77, 70, 60, 58, 58, 62, 60, 74, 50, 70, 40, 47, 68, 76, 60, 74, 69, 50, 55, 56, 60, 78, 51, 64, 63, 49, 55, 83, 41, 72, 54, 56, 59, 77, 53, 75, 67, 60, 57, 48, 79, 63, 54, 52, 72, 60, 72, 60, 57, 66, 56, 61, 59, 42, 74, 44, 59, 60, 57, 74, 79, 45, 74, 66, 67, 62, 50, 70, 55, 51, 63, 56, 81, 48, 65, 54, 48, 47, 49, 53, 70, 65, 68, 35, 56, 44, 37, 53, 52, 64, 55, 77, 60, 45, 75, 46, 59, 49, 64, 50, 64, 52]

number_845 = [105, 94, 106, 92, 106, 104, 81, 89, 117, 108, 104, 84, 102, 104, 83, 95, 88, 89, 106, 109, 122, 98, 84, 70, 84, 91, 87, 100, 90, 106, 96, 112, 90, 86, 103, 97, 85, 110, 98, 77, 99, 105, 71, 92, 75, 92, 97, 107, 79, 99, 81, 106, 121, 88, 103, 92, 101, 84, 77, 119, 103, 79, 106, 113, 105, 75, 92, 96, 91, 97, 90, 100, 112, 92, 96, 94, 115, 100, 115, 105, 84, 84, 98, 80, 86, 99, 80, 94, 110, 102, 104, 86, 97, 93, 90, 87, 89, 115, 100, 108, 93, 98, 90, 91, 83, 92, 103, 109, 111, 105, 96, 91, 91, 89, 114, 101, 114, 94, 79, 93, 100, 106, 102, 87, 105, 94, 76, 101, 61, 81, 93, 110, 97, 97, 78, 105, 81, 100, 94, 83, 94, 96, 97, 88, 101, 100, 101, 107, 100, 95, 76, 104, 91, 105, 88, 83, 113, 87, 91, 87, 83, 133, 96, 92, 82, 109, 99, 80, 97, 87, 109, 78, 148, 111, 87, 131, 83, 107, 97, 84, 99, 90, 89, 93, 99, 94, 98, 116, 97, 127, 97, 64, 90, 93, 80, 95, 87, 99, 101, 108, 88, 73, 73, 113, 110, 97, 74, 107, 97, 82, 113, 94, 95, 116, 97, 109, 102, 72, 111, 124, 115, 96, 114, 79, 96, 102, 131, 88, 100, 84, 91, 84, 88, 102, 76, 103, 91, 92, 89, 82, 99, 87, 103, 85, 88, 103, 79, 91, 84, 115, 87, 112, 94, 81, 100, 94, 114, 80, 83, 96, 88, 113, 95, 80, 121, 116, 110, 102, 60, 103, 97, 68, 112, 97, 93, 106, 99, 109, 115, 66, 98, 115, 103, 96, 101, 82, 89, 102, 99, 94, 83, 110, 92, 104, 86, 87, 98, 94, 86, 112, 105, 94, 87, 101, 99, 115, 98, 94, 88, 140, 96, 93, 92, 89, 114, 95, 77, 126, 91, 108, 102, 110, 109, 109, 85, 92, 90, 83, 86, 102, 106, 129, 83, 97, 101, 115, 73, 60, 87, 107, 119, 91, 83, 81, 101, 86, 106, 117, 76, 115, 92, 85, 101, 102, 89, 113, 98, 100, 100, 99, 105, 99, 78, 77, 101, 82, 93, 87, 90, 96, 85, 84, 113, 103, 104, 93, 90, 118, 86, 96, 82, 73, 106, 104, 100, 97, 84, 93, 94, 89, 89, 80, 111, 123, 102, 106, 88, 97, 90, 78, 117, 88, 125, 116, 115, 78, 106, 103, 94, 102, 100, 70, 94, 99, 84, 102, 102, 96, 97, 100, 108, 127, 103, 96, 84, 78, 101, 77, 101, 96, 100, 96, 101, 113, 89, 104, 90, 83, 103, 77, 92, 78, 111, 94, 98, 103, 89, 107, 97, 83, 84, 87, 82, 120, 105, 103, 113, 116, 94, 96, 108, 84, 88, 87, 93, 96, 85, 101, 88, 88, 106, 92, 102, 112, 95, 81, 92, 94, 88, 87, 89, 88, 74, 87, 101, 97, 96, 68, 88, 74, 78, 125, 106, 79, 78, 97, 119, 103, 91, 81, 102, 103, 99, 97, 84, 107, 81, 81, 120, 92, 110, 104, 97, 58, 91, 86, 91, 90, 85, 96, 82, 86, 83, 108, 102, 98, 107, 93, 105, 99, 99, 90, 85, 100, 71, 101, 79, 93, 92, 102, 128, 100, 94, 95, 99, 93, 91, 103, 101, 98, 95, 86, 84, 103, 106, 88, 78, 63, 105, 92, 87, 109, 98, 94, 113, 105, 91, 89, 95, 124, 106, 96, 76, 82, 98, 110, 98, 103, 85, 99, 114, 117, 112, 100, 94, 100, 110, 87, 88, 100, 107, 79, 106, 91, 75, 93, 110, 86, 92, 84, 92, 103, 104, 112, 91, 85, 106, 71, 95, 83, 93, 89, 103, 70, 89, 78, 91, 91, 97, 77, 111, 93, 95, 96, 113, 110, 94, 103, 110, 77, 103, 101, 97, 120, 98, 86, 83, 119, 70, 89, 98, 103, 82, 90, 94, 98, 122, 110, 101, 101, 105, 98, 115, 83, 100, 68, 100, 120, 103, 92, 93, 91, 104, 94, 108, 109, 92, 124, 108, 101, 84, 76, 89, 101, 101, 104, 98, 112, 89, 111, 102, 102, 93, 98, 85, 105, 91, 125, 80, 96, 95, 76, 85, 83, 101, 101, 109, 95, 104, 92, 94, 88, 106, 86, 118, 116, 72, 96, 116, 101, 88, 87, 120, 89, 116, 80, 84, 90, 98, 149, 93, 109, 103, 99, 92, 76, 97, 75, 107, 96, 105, 103, 108, 97, 88, 93, 100, 78, 96, 80, 112, 116, 102, 84, 64, 94, 98, 101, 98, 91, 94, 88, 72, 101, 105, 96, 98, 102, 85, 98, 91, 79, 93, 88, 101, 76, 122, 121, 96, 104, 86, 86, 106, 100, 93, 96, 112, 100, 98, 93, 117, 76, 91, 112, 106, 74, 92, 121, 110, 94, 108, 92, 110, 78, 105, 101, 112, 94, 86, 90, 76, 83, 106, 117, 111, 77, 82, 93, 104, 94, 76, 101, 95, 102, 83, 84, 110, 87, 119, 93, 81, 107, 114, 96, 72, 100, 98, 119, 86, 93, 85, 105, 99, 100, 86, 108, 76, 127, 115, 98, 78, 107, 118, 85, 97, 99, 92, 96, 99, 88, 81, 101, 123, 96, 91, 97, 111, 117, 82, 105, 76, 82, 93, 90, 99, 108, 102, 73, 110, 101, 116, 79, 85, 119, 89, 108, 107, 95, 106, 90, 89, 98, 103, 93, 92, 84, 89, 98, 108, 79, 82, 92, 104, 114, 80, 124, 90, 87, 118, 120, 97, 110, 104, 82, 99, 103, 74, 90, 99, 99, 85, 100, 100, 100, 99, 90, 97, 86, 106, 80, 97, 86, 100, 81, 91, 85, 89, 106, 103, 91, 99, 77, 87, 106, 96, 118, 66, 104, 123, 102, 92, 101, 89, 94, 88, 100, 83, 117, 80, 84, 86, 95, 99, 93, 103, 83, 105, 90, 105, 97, 104, 86, 92, 100, 97, 109, 73, 92, 108, 90, 99, 118, 95, 113, 92, 106, 120, 101, 95, 113, 109, 126, 124, 86, 98, 104, 85, 82, 85, 89, 96, 110, 107, 87, 98, 95, 111, 116, 98, 83, 98, 96, 86, 98, 76, 102, 75, 93, 93, 121, 76, 122, 82, 103, 96, 91, 121, 106, 100, 97, 108, 87, 96, 100, 112, 86, 100, 95, 96, 127, 120, 115, 90, 91, 100, 106, 98, 105, 88, 74, 80, 79, 93, 94, 89, 89, 98, 99, 92, 92, 98, 73, 81, 118, 88, 101, 102, 110, 91, 80, 71, 100, 79, 101, 92, 89, 112, 105, 117, 105, 95, 85, 100, 76, 70, 81, 92, 99, 94, 88, 93, 95, 112, 110, 88, 95, 90, 103, 88, 100, 85, 100, 82, 84, 97, 104, 100, 105, 93, 101, 81, 81, 103, 103, 98, 89, 105, 120, 95, 87, 102, 91, 94, 93, 107, 96, 81, 91, 99, 98, 97, 75, 90, 90, 96, 85, 82, 90, 102, 115, 94, 83, 92, 103, 105, 81, 93, 81, 103, 96, 84, 123, 106, 80, 106, 90, 89, 96, 69, 118, 95, 108, 107, 88, 98, 95, 114, 113, 90, 91, 92, 116, 110, 115, 125, 82, 89, 93, 84, 84, 100, 110, 97, 90, 96, 82, 99, 104, 95, 86, 99, 94, 94, 108, 110, 90, 106, 75, 87, 106, 99, 91, 113, 101, 77, 92, 91, 100, 114, 95, 112, 96, 80, 85, 118, 97, 117, 92, 89, 97, 111, 86, 109, 95, 92, 88, 89, 117, 84, 90, 81, 112, 93, 93, 85, 93, 106, 85, 99, 109, 75, 95, 94, 84, 90, 99, 100, 110, 81, 120, 98, 91, 96, 81, 99, 75, 92, 103, 81, 118, 98, 98, 96, 77, 79, 81, 79, 106, 96, 96, 77, 95, 80, 84, 73, 80, 97, 96, 105, 96, 92, 115, 84, 107, 83, 91, 78, 101, 89]

number_900 = [144, 150, 151, 128, 161, 139, 124, 133, 153, 153, 147, 130, 139, 149, 124, 133, 133, 136, 135, 142, 152, 129, 127, 95, 116, 121, 134, 128, 118, 147, 137, 141, 123, 121, 150, 150, 115, 145, 136, 116, 137, 136, 102, 127, 124, 122, 145, 154, 120, 121, 125, 153, 163, 125, 136, 140, 132, 118, 120, 166, 148, 131, 133, 147, 150, 116, 138, 128, 126, 137, 119, 145, 143, 131, 127, 133, 159, 134, 156, 129, 129, 120, 127, 126, 115, 135, 112, 136, 149, 137, 153, 124, 134, 131, 147, 129, 120, 166, 122, 147, 151, 126, 125, 132, 128, 129, 145, 132, 135, 150, 141, 136, 124, 120, 143, 143, 138, 138, 117, 146, 145, 132, 133, 112, 147, 147, 117, 139, 93, 129, 119, 136, 144, 147, 115, 143, 142, 140, 141, 119, 130, 142, 118, 133, 143, 144, 140, 148, 143, 138, 135, 153, 135, 148, 131, 129, 136, 125, 132, 144, 127, 173, 138, 122, 113, 157, 155, 123, 139, 122, 135, 108, 181, 162, 127, 168, 131, 159, 126, 118, 142, 130, 132, 134, 139, 117, 126, 160, 129, 158, 127, 98, 132, 126, 95, 135, 131, 135, 139, 147, 120, 114, 114, 161, 144, 133, 111, 144, 129, 125, 150, 135, 131, 146, 139, 143, 144, 106, 162, 170, 155, 125, 139, 119, 128, 146, 173, 118, 139, 112, 115, 117, 126, 155, 120, 134, 131, 127, 116, 117, 155, 109, 131, 126, 116, 140, 115, 120, 132, 143, 122, 147, 131, 112, 125, 123, 147, 126, 118, 133, 124, 169, 141, 118, 162, 158, 141, 158, 104, 139, 140, 99, 149, 147, 125, 147, 148, 137, 156, 100, 141, 183, 139, 129, 125, 132, 128, 116, 131, 140, 137, 135, 147, 139, 140, 133, 113, 133, 129, 124, 145, 138, 126, 135, 141, 128, 150, 134, 136, 124, 174, 159, 122, 123, 143, 139, 138, 115, 160, 127, 134, 137, 162, 144, 135, 115, 138, 145, 113, 118, 147, 158, 179, 143, 144, 130, 168, 123, 102, 131, 153, 155, 142, 113, 122, 157, 129, 126, 161, 119, 139, 127, 117, 140, 145, 128, 144, 146, 136, 152, 133, 155, 124, 123, 128, 130, 123, 130, 133, 118, 141, 120, 122, 142, 147, 144, 136, 117, 153, 120, 133, 129, 93, 165, 147, 128, 157, 128, 124, 134, 124, 133, 116, 149, 168, 145, 135, 139, 126, 112, 116, 162, 116, 172, 145, 156, 117, 141, 137, 149, 134, 130, 98, 148, 130, 133, 128, 138, 126, 140, 146, 149, 162, 140, 128, 120, 116, 134, 103, 152, 126, 145, 135, 150, 157, 132, 136, 123, 117, 132, 134, 120, 131, 140, 125, 146, 140,

123, 145, 126, 119, 128, 112, 121, 155, 143, 141, 141, 141, 135, 139, 150, 117, 132, 129, 130, 144, 132, 149, 131, 123, 129, 112, 147, 159, 132, 116, 126, 127, 124, 123, 119, 121, 123, 140, 151, 126, 133, 105, 127, 114, 125, 175, 148, 108, 104, 124, 159, 155, 130, 110, 149, 134, 134, 146, 121, 135, 119, 132, 157, 132, 149, 139, 121, 104, 121, 116, 136, 122, 122, 139, 132, 132, 115, 139, 125, 130, 155, 132, 144, 129, 128, 139, 116, 136, 102, 137, 119, 117, 124, 133, 160, 138, 131, 135, 139, 129, 140, 153, 151, 137, 118, 128, 125, 141, 158, 120, 106, 100, 160, 134, 132, 147, 140, 135, 147, 152, 136, 127, 128, 161, 133, 123, 106, 116, 146, 139, 128, 145, 120, 156, 140, 137, 150, 131, 125, 145, 150, 126, 124, 132, 134, 111, 133, 126, 97, 128, 132, 124, 137, 131, 141, 140, 152, 151, 133, 109, 136, 117, 132, 116, 132, 121, 133, 112, 124, 109, 141, 122, 124, 115, 153, 129, 132, 127, 135, 157, 141, 154, 147, 121, 125, 136, 121, 172, 137, 126, 129, 156, 116, 123, 132, 139, 125, 129, 129, 161, 167, 139, 138, 125, 150, 139, 156, 109, 159, 98, 142, 161, 141, 149, 128, 142, 151, 120, 144, 158, 132, 160, 148, 133, 126, 109, 135, 132, 141, 130, 142, 149, 130, 140, 143, 122, 128, 144, 125, 128, 121, 157, 115, 125, 125, 123, 117, 125, 131, 153, 136, 124, 136, 119, 140, 119, 134, 132, 151, 161, 114, 129, 149, 137, 126, 142, 162, 114, 187, 120, 129, 122, 129, 166, 121, 161, 139, 141, 125, 120, 127, 124, 152, 143, 136, 151, 141, 147, 129, 137, 137, 114, 125, 113, 147, 156, 147, 118, 117, 149, 128, 154, 137, 126, 125, 123, 117, 141, 146, 142, 140, 139, 123, 138, 131, 127, 125, 140, 142, 105, 156, 170, 151, 155, 113, 126, 129, 155, 135, 131, 154, 153, 141, 121, 149, 113, 135, 145, 140, 119, 133, 149, 148, 122, 135, 120, 156, 123, 150, 147, 155, 128, 129, 137, 114, 124, 151, 167, 144, 118, 130, 121, 154, 138, 124, 152, 147, 152, 121, 123, 141, 140, 152, 126, 116, 141, 159, 128, 122, 158, 131, 155, 125, 133, 115, 152, 134, 142, 139, 142, 124, 161, 155, 145, 120, 137, 159, 117, 138, 136, 130, 128, 128, 133, 118, 153, 170, 132, 124, 127, 160, 150, 118, 162, 116, 119, 126, 113, 138, 156, 143, 109, 151, 142, 160, 125, 119, 166, 136, 161, 136, 116, 145, 134, 135, 127, 131, 125, 119, 123, 116, 142, 151, 115, 122, 127, 143, 157, 135, 161, 137, 123, 150, 175, 133, 157, 152, 132, 141, 142, 134, 132, 136, 143, 115, 157, 136, 147, 129, 130, 132, 113, 143, 111, 137, 125, 141, 114, 126, 130, 104, 133, 148, 111, 149, 106, 135, 139, 140, 165, 123, 123, 168, 137, 146, 144, 137, 137, 132, 140, 111, 163, 122, 122, 123, 120, 133, 131, 134, 147, 148, 128, 157, 131, 142, 115, 128, 151, 134, 143, 126, 127, 147, 128, 141, 161, 134, 134, 129, 141, 166, 136, 125, 168, 133, 147, 155, 123, 137, 130, 118, 117, 120, 131, 139, 147, 148, 126, 131, 144, 142, 148, 131, 111, 143, 144, 108, 134, 116, 125, 115, 120, 133, 175, 123, 144, 128, 129, 142, 135, 144, 126, 135, 125, 149, 127, 127, 144, 149, 128, 129, 128, 132, 184, 172, 157, 121, 140, 156, 136, 139, 137, 142, 125, 109, 112, 132, 117, 130, 121, 130, 140, 132, 135, 124, 111, 119, 165, 141, 126, 145, 142, 128, 112, 111, 140, 118, 130, 130, 135, 145, 143, 161, 132, 137, 130, 122, 115, 104, 103, 138, 125, 149, 118, 129, 128, 151, 139, 129, 126, 122, 151, 135, 125, 136, 138, 135, 117, 133, 154, 158, 130, 134, 142, 114, 109, 137, 128, 137, 132, 153, 171, 120, 137, 139, 118, 124, 134, 132, 136, 131, 123, 130, 133, 137, 120, 127, 120, 122, 118, 124, 126, 133, 177, 133, 120, 139, 150, 139, 115, 130, 121, 156, 125, 108, 143, 153, 127, 138, 130, 128, 140, 108, 164, 121, 140, 143, 123, 132, 117, 131, 146, 119, 124, 142, 147, 158, 157, 160, 111, 120, 118, 111, 121, 151, 138, 141, 136, 137, 112, 158, 139, 116, 129, 136, 141, 124, 139, 149, 116, 145, 123, 115, 139, 130, 115, 139, 117, 117, 125, 123, 126, 154, 135, 161, 123, 114, 125, 144, 145, 159, 144, 130, 141, 158, 116, 153, 145, 138, 129, 121, 157, 125, 122, 122, 149, 134, 129, 133, 128, 141, 116, 133, 143, 111, 139, 145, 128, 126, 139, 134, 137, 117, 160, 131, 127, 126, 117, 144, 107, 119, 139, 115, 163, 133, 131, 144, 125, 139, 120, 129, 144, 138, 126, 120, 143, 119, 131, 108, 115, 136, 126, 138, 142, 115, 142, 139, 147, 126, 143, 118, 145, 130]

```
sns.distplot(number_830, hist=True, color="dodgerblue", label="08:30")
sns.distplot(number_845, hist=True, color="green", label="08:45")
sns.distplot(number_900, hist=True, color="red", label="09:00")
plt.gca().set(title='Frequency of the Number of People in the Queue at Certain Times', ylabel='Frequency', xlabel='Number of People')
plt.legend()
plt.show()
```