Lab 7 – CSCI 112

Due Date: Wednesday, Oct 14 at 11:59pm EST

Information

- This lab is intended to be completed **individually.**
- The files must be submitted with the exact file name provided in this file. If the file names do not match you will receive **zero** points for that file.
- Before you submit, make sure that your code runs. Any code which does not run without errors will receive **zero** points.
- Do not share your work with anyone other than Professor Khan or the TAs. You may discuss algorithms, approaches, ideas, but **NOT** exact code.
- If you submit work after a second past the due date **WILL** be locked out from submission.

Review

Queues

Queues are a linear, ordered collection. Insertion occurs at one end while removal happens at the other. This is also called LIFO (Last In First Out). Priority queues have a similar behavior only upon entering the queue it will move the item up according to its priority, usually by using a **Comparable** class.

linkedQueue.py and linkedPriorityQueue.py

The code for these structures is provided in the book. Open the files and familiarize yourself with the implementations. Test out the classes with **testQueue.py.**

Assignment

Task 1 – Array based Queues [9 points]

Open **`arrayQueue.py`** in the **utils** submodule and fill in the missing code to satisfy the **queueInterface**. You must use the wrap-around approach in this implementation. Make sure to have your array resize itself, both **growing** and **shrinking**, at the appropriate locations. Test your implementation with **`testQueue.py`**. Next open the file **`arrayPriorityQueue.py`** in the utils submodule and complete the **`add`** method for the class. Test your class implementation with **`testPriorityQueue.py`**.

Task 2 – Maze [6 points]

Open the **maze.py** code and examine it, it should be familiar from a while ago in class. Modify the **getOut** method to utilize a parameter, **choiceStorage**, which is either a stack or a queue class. Test out the same maze but with differing storage for your choices. What can you say about the efficiency of either of these approaches? When would a stack be better? When would a queue be better? Do they perform the same? Create a plain text document named **maze.txt** and summarize your theories

Extra Credit Task

Due: Wed Oct, 21 11:59 pm

Market Model [2% Extra Credit]

Open the **marketModel.py** and examine the code- this should look familiar as we covered this version in class. You can test the current **MarketModel** with **marketApp1.py** or **marketApp2.py**. Make the following modifications to the **MarketModel** class:

1. The current implementation has only one cashier. Change the model to allow for another parameter named **numCashiers** to be passed in to **__init__,** and keep a list of cashiers instead of a solitary one. Update **runSimulation** to tell each cashier to serve customers and update the **str** method to show each cashier's **str** on a new line. Add each new customer that shows up to a random cashier for now.
2. Typically, customers try to go to the **shortest** available line. Modify the model to account for this.
3. Customers are sometimes lazy, and only want to go to a checkout that is **close** to their initial location. Instead of picking the optimally shortest line, have the customer arrive at a random cashier's line, and pick the shortest line of locations no more than two lines away from this initial spot. Be careful of edge cases! Run your simulation with at least 4 cashiers.
4. Currently, the **average** number of minutes per customer entered in the interface is used as the processing time for all customers. In real life, processing times vary around the average. Modify the customer class to randomly choose a service time between **1** and **(average * 2 + 1)**.

What To Turn In

Create a zip file named **Lab7_<your W&L ID>.zip**. Inside this zip archive should submit all the files from both tasks. To submit the extra credit part of the lab, zip all the files in the **Extra Credit** folder and submit them to the **Extra Credit (Market Modeling)** assignment on Canvas.