Objective: Practice reading and modifying an existing program. Provides an exposure to the concept of discrete-event simulation using FIFO and priority queues.

To start the homework: Download and extract the file project3.zip. The project3.zip file contains:

- binheap.py - the BinHeap class previously used with a new `peekMin` method which returns the minimum item without removing it
- queue_text.py - the textbook's Python-list based FIFO queue implementation
- priority_queue.py - the PriorityQueue and PriorityQueueEntry class implementations. The PriorityQueue uses a BinHeap to order PriorityQueueEntry items which bundle together a Priority and its corresponding Value.
- project3.py - the working discrete-event simulation of multiple check-out lines (e.g., like at a grocery store). The simulation is based on user inputs of: # of check-out lines, simulation length (# of minutes), probability of a customer finishes shopping within each minute, and average customer check-out time.

Background:

For an introduction to discrete-event simulation (DES) see the Wikipedia page at:
https://en.wikipedia.org/wiki/Discrete_event_simulation

The given DES in project3.py models each cashier with:

- a FIFO queue for customers waiting to check-out (initially an empty queue)
- a cashier "idle" flag to indicate if the cashier is free and ready to check-out a customer (initially True)
- a count of how many customers the cashier checked out during the simulation (initially 0)

The simulation's main-loop uses a `clock` variable to track the simulated time. Each tick of the clock simulates a minute. In a simulated minute the following events can occur: a new customer could arrive and enter a check-out line, cashiers can finish checking out customers and become idle, and idle cashiers with non-empty queues can start their next customer. Since customers starting to be checked out can take several minutes, the future event when the cashier will become free is scheduled using an event queue (a priority queue) ordered by ascending clock time.

Project Description:

Your task for this project is to make **all three** of the following improvements for a more realistic grocery store simulation:

- currently when a customer arrives at the check-out lines, the customer goes into the cashier queue with the fewest number of customers. However, a real arriving customer would look not just at the number of customers in a line, but also how many items those customers have to gauge their check-out time. Modify the simulator to add an arriving customer to the shortest line based on the total check-out time for all customers currently in line. You should not need to modify the FIFO queue class, but just keep an accumulator for each line to track the total check-out minutes of customers in line.

- currently when the simulation ends, it outputs for each check-out lane the number of customers served and the length of the line at the end of the simulation. Additional information would be helpful since customers get frustrated if they have to wait along time in line. Modify the simulation to calculate and print the average waiting time in line across all customers checked-out. Hint: modify the information stored about each customer waiting in the FIFO queue to include the clock-time when they entered the line.

- currently when the simulation starts, the number of cashiers/check-out lines is entered and fixed for the duration of the simulation. However, a real grocery stores might have 20 check-out lanes with only 8 cashiers open. If the open lines all get too long, then a new check-out line is opened and customers at the rear of existing lines shift into it. Modify the simulator to allow the user to specify additional simulation parameters for maximum

number of check-out lines, initial number of open check-out lines, and a threshold for opening a new check-out line (i.e., length of shortest line in terms of total minutes across all customers in that line). Hint: use Deque (e.g., your `lab5/doubly_linked_deque.py`) instead of FIFO queue to model each waiting line so you can use the Deque's `removeRear` method when shifting customers into the newly opened check-out line. For our simulation we will not worry about closing check-out lanes when they get too small.

After completing your modifications, make sure that your simulation results seem reasonable. For example the original project3.py simulation gave the following simulation results:

```
Enter # of cashiers: 5
Enter simulation length (in minutes): 1000
Enter probability of customer arrival each minute (0 to 1): 0.5
Enter average # minutes for a customer: 10

Cashier 0 checked out 91 customers with 11 customers in their line at end of simulation.

Cashier 1 checked out 81 customers with 12 customers in their line at end of simulation.

Cashier 2 checked out 80 customers with 11 customers in their line at end of simulation.

Cashier 3 checked out 88 customers with 10 customers in their line at end of simulation.

Cashier 4 checked out 81 customers with 11 customers in their line at end of simulation.
```

Analyzing these results for reasonability, I make the following observations:

- With a customer arrival probability of 0.5 and 1000 minute simulation, I would expect about 500 total customers. If you add the customers checked-out (421), the customers still in lines (55), and the customers currently being checked-out when the simulation ends (5), then we get 481 which is reasonably close to 500.
- I would expect all the check-out lines at the end of the simulation to be about the same length since arriving customers always get added to the smallest line based on the number of customers in line.

For your simulation modifications, you also can compare your simulation results with a "similar" simulation of the original simulator.

Submit all necessary files for your modified project 3 (binheap.py, node.py, doubly_linked_deque.py, project3.py, etc. ...) as a single zipped file (called project3.zip) to the Programming Project 3 Submission link.