AP Computer Science Name_____
Bank Simulation Lab

Lucky you! You've been hired as an efficiency expert by your local bank (known hereafter as the client). Since customers are likely to be dissatisfied if they are forced to wait in line too long, the bank wants to hire a sufficient number of tellers. However, the bank will lose money if it employs too many tellers. Thus, it's important to analyze how long customers must wait, and how often tellers are idle. A balance is needed between customers waiting and the number of clerks hired. Your job will be to answer some questions concerning the waiting line.

In this problem, we are given the probability of a person coming into the bank and the probability of a transaction being completed during a minute. Thus, while it is easy to imagine a person entering a wait line or being served, we do not know how long particular individuals might have to wait.

Our approach to the problem will involve the use of simulation. We use a queue to keep track of individuals as they enter and leave the waiting line, and we record how many customers were served and how long they waited (including serving time). We also would like to know how long the queue is, so that it doesn't fill up the lobby.

More precisely, we divide time into short intervals, and we use the averages given to determine the probability that a person in line will finish at the teller during a time interval. We also determine the probability that a person will arrive and enter the line. We record when a person enters the end of the line. Then, when the person has finished being served, we compute and record how much time the person was in the bank, waiting and completing the transaction.

At the end of the simulation time, we close the doors, allowing no new people to enter the bank, but continue to process those waiting.

Part I

A. Many days the client gets along with one teller on duty. During any given minute, suppose the probability that a person arrives is 1 out of 9 and the probability of a transaction being completed is 1 out of 5.

B. The client wishes to know
- how many people were served during an 8 hour day (480 minutes)?
- what was the longest and average waiting time for a bank customer?
- what was the average length and longest length of the queue?

C. For each time interval of one minute
1. If anyone leaves the queue,
   - dequeue and calculate their waiting time,
   - update total waiting time,
   - and update longest waiting time.
2. If anyone arrives at the bank,
   - enqueue them, recording the time of arrival,
   - and update number of people served.
3. Calculate the length of the queue
   - display length of the queue or show with a simple bar chart,
   - update total lengths of queue,
   - and update longest queue.

D. At the end of the 8 hour day, close the doors to the bank and
- process the remaining people,

- calculate the required statistics for this day.

E. Run the simulation for 10 days.

F. Average and report the results of the 10 days' simulation.

Part 2

The client is pleased with your report and wonders if you could help with another problem. The bank has several types of customers, and each is served with a different priority. There is a VIP business client that must be accommodated first, immediately going to the front of the line, a VIP individual client who requires second priority service, an average customer, and finally a new customer. The new customers always get added to the back of the line.

This problem requires you to use a **priority queue,** where elements enter in a random order but are removed according to their priority. For example:

- in the lunch line, all seniors get served first, juniors next, sophomores next, and freshman last.
- in a hospital, patients who are more likely to die, but are not dead yet, get the organ transplant first.
- in a print queue, shorter print runs get processed before the longer print runs.

"According to their priority" means something different for each situation. Accordingly, each element in the queue must be a *Comparable*, with a `compareTo`() method that defines what "priority" means in the given situation.

Create a customer class, which will implement Comparable, to model the 4 kinds of customers namely, VIPBusiness, VIPIndividual, AverageJoe, and Newbie. By implementing the Comparable interface you must provide a `compareTo` method that defines a relation of order, or a way to compare your class's objects. Output the numbers served, the average wait time, and the longest wait time for each kind of customer.

A Random word

Suppose, in a given minute, the probability of a transaction being completed is 1 out of 5. To determine if someone (at the head of the queue) has finished their transaction, generate the random numbers (0, 1, 2, 3, and 4). Thus, each minute there is a 1 out of 5 chance of someone arriving. If the number returned by the function is 0, then someone is dequeued from the queue. Likewise if the probability of someone arriving each minute is 1 out of 9, Math.random() to generate the numbers (0, 1, 2, ... 8).