

Program 5: Queues and Simulation

(due Tuesday April 30 at 11:00 AM)

This assignment is inspired by the game *Roller Coaster Tycoon*. You will simulate the events during a day at an amusement park. We want to know the amount of time customers spend in line, and how much money the park will make in a day.

Objectives (this is how the program will be graded)

- Implement linked list
- Implement queue
- Apply object-oriented design and encapsulation principles
- Good documentation
- Style (modularity, proper naming, etc.)

The Simulation

The park is described in a file, which contains a description of the rides, the probability that a customer will arrive at a particular time, the probability that the customer will leave, and the amount of time the park is to be kept open. The simulation is very simplified.

Time. Time is discrete, in integer units. The simulation runs from time zero until the specified closing time, at which point no more new customers are let into the park, but all customers waiting in line are given rides before they leave.

Rides. The capacity (number of people) that may go on a ride is given in the input file. The line (queue) waiting to ride has no size limit. All rides are simulated as all-on, all-off, such as a Merry-Go-Round or Bumper Cars, instead of something like a Ferris Wheel where a few people get off and a few on every cycle. The cycle time of a ride is the amount of time that people are on. When that time is over, the ride is emptied and the first people in line are loaded, up to the capacity of the ride. Loading takes no time.

Rides cost a given amount of money to run, regardless of how many people are on board. They will not run if no one is waiting to ride. The price paid by the customers is per person.

People. When people get off a ride, they decide where to go next. With a probability specified in the file, they may decide to leave. If they stay, they will choose among all available rides with even probability. In other words, the chance they will be on the same ride they just left is the same as that of any other ride. So if there are five rides, there's a 20% chance that a customer will go to each.

We are not going to count the amount of time it takes customers to go from one ride to another. They travel instantaneously.

Arrival. An arrival probability is given in the input file. With this probability, a single person will arrive at each time. No more than one person arrives per clock tick. It's very unrealistic, but the arrival probability is the same throughout the day.

Algorithm. Each cycle of the simulation runs as follows.

1. Empty a customer at a time from any rides that are finished.

2. Decide whether each customer leaves the park. If not, decide to which ride customer goes.
3. Check for a new arrival to the park, and send to a ride.
4. Start any rides that are ready to go.

Don't forget to empty the park at the end of the day.

Input Description File

Here's an example description file (Park1.txt).

```
#
# Lines that begin with '#' are comment lines
#   all other lines have data.
#
# Fields are separated by colon
#
# ride
#   Rides begin with the word "ride", then
#       Name of ride
#       Time it takes to cycle the ride
#       Capacity - total number of people it holds.
#       Cost - pennies it costs to run one cycle
#       Price - price per person to ride
#
# arrival
#   Probability that a person will enter park at any given time
#
# departure
#   Probability that a person will leave park when exiting ride
#
# closes
#   Time when park closes (0 is when park opens)
#
ride:Ferris Wheel:4:2:12:25
ride:Merry-Go-Round:6:10:8:20
arrival:0.7
closes:100
departure:0.2
```

Output

Print at least the average and maximum wait times for each of the rides, as well as the cost to run each ride and amount of money made for each ride. Finally, print the profit for the day.

In my implementation, I also printed the events as they occurred (for debugging). Don't hand that in on paper, but you may wish to save in files and add to Zip archive to show your work.

Product

Submit the program using the Blackboard system. Hand in listing of the program, and the output described above for both of the two test cases (Park1 and Park2) on the web site. I have posted the results of my implementation. Please let me know if you find any errors in my results.

You'll need a way to specify input file. Use command line or GUI, *not a variable in program*.

Notice that you'll have to follow the steps *exactly* as I did in order to get the same results. For example, I examine each ride in the same order listed in the input file, and empty the customers from a ride in the same order they entered the ride (I kept a queue for the riders). Make sure to seed your random number generator with a zero so you get the same answers that I got.