CSC 2110, Spring 2013
Programming Assignment 3: Simulation of a Queuing System
Assign date: March 20, 2013, Due: April 3, 2013 at 9:00pm

Overview

Simulation is the use of one system to imitate the behavior of another system. Simulations are often used when it would be too expensive or dangerous to experiment with real systems. There are physical simulations, such as wind tunnels that are used to experiment with design of car bodies or airplanes, flight simulators that used to train pilots. Mathematical simulations are systems of equations used to describe some system, and computer simulations use computer programs to imitate the behavior of the system under study.

Let's look at a very useful type of simulation that uses queues as the basic data structure for simulations. In fact, the real world system is called queuing system. A queuing system is made up of servers and queues of objects to be served. We deal with queuing systems all the time in our daily lives. When you stand in line to checkout at a supermarket or to cash a check at the bank, you are dealing with queuing system. When you make a phone call to complain about a problem with your computer that you have recently purchased and get a recording that says: **"Thank you for calling Cheap Computers. The next available operator will answer your call. Please wait."** – you are dealing with a queuing system.

Please wait. Waiting is the critical element. The objective of a queuing system is to utilize the servers (checkout clerk, bank teller, operators etc.) as much as possible, while keeping the wait time within a reasonable limit. These goals usually require compromise between cost and customer satisfaction. No one likes to stand in line, if there were one checkout counter for each customer in a supermarket, the customers would be delighted. The supermarket, however, would not be in business for long. How does a company determine the optimal compromise between the number of servers and the wait time? One way is by experience; the company tries out different numbers of servers and sees how things work out. There are two problems with this approach: it takes too long and it is too expensive. Another way of examining this problem is by using a computer simulation. To simulate a queuing system, we must know several things:

1. The number of events and how they affect the system.
2. The number of servers
3. The distribution of arrival times
4. The expected service time for each object.

The simulation program uses these parameters to predict the average wait time. The interactions between these parameters are the rules of the model. By changing these parameters, we change the rules. The average wait times are then examined to determine what a reasonable compromise would be.

Consider a bank with one teller. How long does a customer have to wait? If business gets better and customers start to arrive more frequently, what would be the effect on the average wait time? When would the bank need to add a second teller? This problem has the characteristics of a queuing system. We have a server (the teller), objects being served (customers), and the average wait time is what we are interested in observing.

The events in this system are the arrivals and departures of customers. Suppose that the number of tellers is 1, a transaction takes 5 minutes, and a new customer arrives about every 3

minutes. Let's look at how we can solve this problem as a time driven simulation. The time driven simulation is one in which the program has a counter that represents a clock. To simulate the passing of a unit time (a minute for example) we increment the clock. We run the simulation for a predetermined amount of time, say 100 minutes.

From a software point of view, the simulation is big loop that executes a set of rules for each value of the clock, from 1 to 100, in this example. Here are the rules that are processed in the loop body.

1. If a customer arrives, he or she gets in line.
2. If a teller is free and if there is anyone waiting, the first customer in line leaves the line and advances to the teller window.
3. If a customer is at the teller window, the time remaining for that customer to be serviced is decremented.
4. If there are customers in the line, the additional minute that they have remained in the queue is recorded.

The output from the simulation is the average wait time. We calculate this value using the following formula:

Average wait time = total wait time for all customers / number of customers

The average wait time is only the time that customers spend waiting in line. Time spent processing at the teller does not contribute to the average wait time. Also, only the wait times for customers that have made it through the line should contribute to the total wait time for all customers. The wait times of customers remaining in the line when the simulation finishes should not contribute to the total wait time of all customers. It follows that the number of customers in this formula is the number of customers that have finished waiting in line and have made it to the teller window.

Given this output, the bank can see whether their customers have an unreasonable wait in a one-teller system. If so, the bank can repeat the simulation with two tellers.

We have described this example in terms of a single teller and a specific arrival rate and transaction time. In fact, these simulation parameters should be varied to see what effect the changes have on average wait time. Therefore these values are read as input to the program. We refer to the bank tellers as servers, and the customers as jobs. This program simulates a multiple-server/single-queue system. In contrast, grocery stores are usually multiple-server/multiple-queue systems.

Program Specifications

For this project you will develop a program to simulate a queuing system.

Function

The program simulates a multi-server/single-queue system, using the following simulation parameters: length of simulation, average time between arrivals, number of servers, and average transaction time.

Input

The simulation parameters are all integers, are entered interactively, and include:

1. Length of simulation
2. The average transaction time
3. The number of servers
4. The average time between arrivals.

The program must prompt the user for the inputs (positive integers), which are each entered on separate lines. At the end of each simulation, the program asks if another simulation is desired. If the user responds positively, the program prompts the user to input a new set of simulation parameters.

Output

The outputs are printed to the screen. The output consists of the set of simulation parameters, the resulting average wait time, and the number of jobs that are still waiting when the simulation stops. Average wait time should be specified to two decimal places.

Processing Requirements

The program must use a queue to hold the incoming jobs and must use input validation to guard against range errors in the user's input. The maximum values allowed for each parameter must be described as constants, so that they can be **"tuned"** if necessary. Use the following initial values:

a. Maximum Servers 10
b. Maximum Simulation Length 36000
c. Minimum Transaction Time 2
d. Maximum Transaction Time 600
e. Maximum Time Between Arrival 100

Assumptions

1. No more than one job arrives per unit time
2. User inputs can be assumed to be numeric.

Sample Execution

```
--------------------------------------------------
- Please enter values for the following parameters -
--------------------------------------------------

Simulation length: 100
Average transaction time: 5
Number of servers: 1
Average time between arrivals: 3

--------------------------------------------------
- Length of simulation: 100
- Average transaction time: 5
- Number of servers: 1
- Average time between arrivals: 3
-
- Average wait time: 24.00 minutes
- Number of customers still waiting: 16
--------------------------------------------------

Would you like to perform another simulation? (Y/N): Y

--------------------------------------------------
- Please enter values for the following parameters -
--------------------------------------------------

Simulation length: 100
Average transaction time: 5
Number of servers: 1
Average time between arrivals: 2

--------------------------------------------------
- Length of simulation: 100
- Average transaction time: 5
- Number of servers: 1
- Average time between arrivals: 2
-
- Average wait time: 32.00 minutes
- Number of customers still waiting: 33
--------------------------------------------------

Would you like to perform another simulation? (Y/N): Y

--------------------------------------------------
- Please enter values for the following parameters -
--------------------------------------------------

Simulation length: 100
Average transaction time: 5
Number of servers: 2
Average time between arrivals: 2
```

```
--------------------------------------------------
- Length of simulation: 100
- Average transaction time: 5
- Number of servers: 2
- Average time between arrivals: 2
-
- Average wait time: 16.00 minutes
- Number of customers still waiting: 16
--------------------------------------------------

Would you like to perform another simulation? (Y/N): Y

--------------------------------------------------
- Please enter values for the following parameters -
--------------------------------------------------

Simulation length: 36001
Simulation length: 100
Average transaction time: 1
Average transaction time: 601
Average transaction time: 7
Number of servers: 11
Number of servers: 1
Average time between arrivals: 101
Average time between arrivals: -3
Average time between arrivals: 3

--------------------------------------------------
- Length of simulation: 100
- Average transaction time: 7
- Number of servers: 1
- Average time between arrivals: 3
-
- Average wait time: 30.00 minutes
- Number of customers still waiting: 20
--------------------------------------------------

Would you like to perform another simulation? (Y/N): p
Invalid input.  Please try again.
Would you like to perform another simulation? (Y/N): asdf
Invalid input.  Please try again.
Would you like to perform another simulation? (Y/N): N
```

Deliverables:

- I expect you to fully design and implement this project yourself. You will need to use C++ classes to be able to create the objects that interact in the simulation program. You are **required** to use objects to complete this program and each object **must** be written in 2 files (a .h and a .cpp file).
- You must also include a README file describing what your program does and how to compile and run your program.
- You need to zip all your source code (all header and cpp files, no executable). The name of your zip file should be lastname_prog3_2110.zip (for example my filename would be mcdaniel_prog3_2110.zip.
- Remember to **#include <cstdlib>** so that we will have no problems compiling your code.

Your program will not be graded if it does not compile.

Grading:

This program is worth 100 points, distributed as follows:

Functionality (program works correctly, and meets the specifications)	**85 pts**
Program Style Meaningful variable names Proper indentation Correct program header Comments throughout the program	**10 pts**
README file	**05 pts**

Start working on your assignment early; and don't hesitate to contact the TA or me if you need help.