University of Toronto

CSC270S—Data Structures and Techniques, Fall 1999

Assignment 3: Simulation

Due: Monday November 22, Worth: 12%.

Start NOW. This assignment will take longer than you expect. You may find instructive to familiarize yourself with the gas simulation, since the base code was based on it. The README file describes the similarities and differences between them.
Check the webpage of the course periodically for bug fixes. Bugs may appear in the code that we have provided. When we find one, we will post a fix in the webpage. It is very unlikely that these fixes will be in parts of the code that you are modifying (since that is restricted to a few functions), so it should be easy for you to incorporate the fixes that we post.

The problem

The purpose of this assignment is to teach you how to simulate a complex system. It will also help you to get comfortable with C++.
You are to simulate customers in a supermarket. We focus on the arrival of customers to be attended by a cashier. Each cashier has its own separate queue. The main objective of the simulation is to recommend to the supermarket manager how many cashiers should be express (that is, cashiers devoted to customers buying few items). Other studies are also done.
Customers arrive at the cashier stand (checkout counter). Each customer has a number of items to pay for. Depending on this number, the customer goes to either an express or a regular cashier. There are only two types of cashiers (regular and express). We define a threshold `maxExpressItems` (maxExpressItems is set to 10 for the simulation), such that if the number of items bought by a customer does not exceed `maxExpressItems`, s/he must go to an express cashier; otherwise, s/he must go to a regular one. We also assume that a customer will go to the shortest queue among the ones of the appropriate type (regular or express). If more than one queue has the same smallest size, it does not matter which one is chosen. For the sake of simplification, no customer will ever leave his/her queue to move to a shorter one. The cashiers attend the customers in the associated queue; the service time depends on the number of items bought.

Events and probabilities

The main events in our simulation are `arrivalEvent` and `departEvent` which correspond to the arrival of a customer to the cashier queues, and the departure of a customer from a cashier.

`arrivalEvent`: When a customer enters the simulation, the number of items associated with him/her must be generated. You should do this in the following way: We define the constant `proportExpressCust` to be what we anticipate would be the proportion of customers which use the express queue. For this simulation, proportExpressCust is always set to 0.3. When a customer arrives, we make a random decision, choosing that s/he will be an express customer with probability proportExpressCust, and a non-express customer otherwise. Next, we determine the number of

items for the customer by picking a uniform random value between 1 and `maxExpressItems` for an express customer, and between `maxExpressItems` +1 and `itemsBoughtMax` for a non express customer (itemsBoughtMax is set to 80 for this simulation).
After a customer is created, if a cashier of his/her type is free, the customer starts being served immediately. Otherwise, the customer is sent to the appropriate queue. Then, the arrival a new customer is scheduled. The interarrival time follows an exponential distribution.

`departEvent:` This event corresponds to the end of cashier service for a customer. It may also trigger the start of service of a new customer if one is in that cashier's queue. Statistics for the departing customer are collected at this moment. Then, the cashier is made available for new customers. If there is a customer in the cashier queue, the cashier starts attending that customer right away.
The method `startService` inside class `Cashier` takes care of determining the service time for the arriving customer. It also schedules the departure event corresponding to this customer. The service time is a linear function of the number of items bought by the customer. There is a constant amount of time independent of the number of items (e.g. time for receiving the payment), and there is an amount of time spent per item (e.g. time to scan an item). These constants are given by `serviceTimeBase` and `serviceTimePerItem`.

In addition to the previous events there are two "pseudo-events" `EndOfSimulationEvent` and `ReportEvent`. Initially, three events are scheduled: the end of the simulation, the first report, and the arrival of the first customer.

Programming

The starter code is in the webpage of the course.
The first part of the assignment consists in familiarizing yourself with the code. You have to write C++ code for: the generation of the exponential interarrival times of customers, the generation of the number of items bought by a customer, and the computation of the service times. Moreover, you have to do a report for each of 3 different mean interarrival times: 8, 9, and 10 seconds. For each of these rates, you have to determine which is the optimal number of express cashiers. In order to conclude this, you will have to collect statistics. The file `stats.C` shows which statistics you have to collect. The word `CODE` indicates the places that you have to change code (to find these places in the starter code type `grep CODE *.C` and `grep CODE *.h`).

The second part of the assignment consists in simulating coffee breaks for the cashiers. The rules for coffee breaks are as follows:
Every half-hour, on the half-hour, one of the cashiers gets a 5 minute coffee break. The cashiers get their breaks in turn, following a natural cyclic order. (Eg. if they are numbered 1 to 20 then 1 gets the first break, 2 gets the next break and so on. After 20 finishes a break, 1 gets the next break.) At each half-hour, the cashier who gets the break will finish the customer (if any) that s/he is serving and then begin a 5 minute break immediately. As soon as s/he begins the break, his/her entire queue moves to other queues. Preserving the order of the queue during the move, one by one the customers move to the back of the shortest queue of the same type (express/non-express). (This rule is not exactly realistic, but it simplifies the simulation.)
If there is only one cashier of a certain type (express/non-express) then that cashier can never take a coffee break. S/he is removed from the cyclic order.
In the unlikely case that on a half hour, the person who got the last break has not returned from that break yet (for example if his/her customer took more than 25 minutes to service), then the

next person in the order does not get to take his/her break. S/he must wait for the next half-hour to take a break, and the breaks then continue in the usual order.
If the store opens exactly on a half-hour, there is not an immediate coffee break. Eg. if the store opens at 7:00, then the first break occurs at 7:30, but if it opens at 6:45 (or even 6:59) then the first break occurs at 7:00.
Simulating coffee breaks will require the creation of 2 new events: `startCoffeeBreakEvent` and `endCoffeeBreakEvent`.

Tests and report

With your program, you must submit tests and a report. The report should have 3 parts:
Part A: You should include a brief explanation (at high level and in English) of the changes in the new events you have introduced.
Part B: For each of the cases (customers mean interarrival time equal to 8, 9, and 10) do the full variation of each possible number of express cashiers, 1 to 19. Do not include the full output for all those tests, but organize the last line of output (with the final times) into a table of results. So, for the case of mean interarrival time equal to 8 seconds, you will have 19 lines of statistics in a table. Based on these statistics you should recommend what is the best number of express cashiers; you must justify your answer by analyzing the data. This should take no more than 1 paragraph. Do the same for interarrival time of 9 and 10 seconds.
The file `stats.C` shows which statistics you have to collect; see also the file `sampleoutput` in the starter code. You have to decide for yourself what statistics are useful in order to provide the optimal number of express cashiers in each test case (customers mean interarrival time equal to 8, 9, and 10 seconds). Your report should include a discussion on the results.
Part C: Study the effect of adding coffee breaks in a similar way. An analysis similar to the one in part B should be done in order to recommend any changes in the number of express cashiers once coffee breaks are introduced.
Your report should describe your experimental results. It should be brief and should aid the marker in understanding your work and the meaning of the results.

What to hand in

Hand in a hardcopy printout of all your programs, a hardcopy printout of the output of your program, and the report. Highlight with a colored pen the changes or additions that you have made to the code.
You will also submit your code and test files electronically. More details regarding electronic submission will appear on the course web page.