Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2016

Homework 7 – Simulation of Grocery Store Checkout Line using Threads

Due: 06/05/2016, Friday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism will not be tolerated!

Introduction

In this homework, you are asked to write a **multithreaded** C++ program that simulates queues of customers waiting in a store to pay their groceries. There are two (2) cashiers and therefore, two separate queues in the store, one for each cashier. When arrived, customers need to go to the rear of one of the queues and wait in that queue until their turn comes to be served by a cashier. Whenever a cashier is available, the customer at the front of the cashier's queue begins the checkout process with that cashier. In this homework, you will simulate the customer's arrivals and their transactions with the cashiers via a static queue data structure and multithreading techniques.

The time between arrivals of two customers is probabilistic. Moreover, the time for a cashier to finish a checkout process is also probabilistic. Thus, the inter-arrival time between two customers and the checkout durations of cashiers are *random* values. The parameters of these random values will be input (see Section "Details of Simulation" for details).

In the scope of this homework, simulation means to employ two queues which customers arrive at random intervals and cashiers get the next customer to process. Simulation starts after taking the inputs from the keyboard, and continues until all of the customers arrive and complete their payments. During the simulation, you are going to display some verbose output about actions of customers and cashiers (see Section "Details of Simulation" for details).

Using Threads

There will be three threads (other than main thread) in your program. One thread is for the arrival of the customers. Other two threads are for the cashiers; one for each. In the customer thread, the customers will arrive one by one at random intervals and choose to be enqueued to one of the two queues using a predefined rule. The details on how a customer chooses a queue will be revealed in the next section. In each cashier thread, the corresponding cashier dequeues a customer from its queue, if the queue is not empty. Then the cashier starts the checkout process, which also takes a random time period.

Please use the `HW7IntQueue` class given in this homework package as the queue data structure. Moreover, do not forget that each cashier and its thread have a separate `HW7IntQueue` object for dequeuing. However enqueuing customers to those queues will be done in a single customer thread. Such a structure may cause some special cases to deal with in a multithreaded application. For example, there might be a synchronization conflict during enqueuing and dequeuing customers to/from the queues. Some other special cases and conflicts may occur; therefore, you have to take some precautions in the threads to avoid such situations. Please refer to lecture and lab materials which include methods for dealing with these cases using *mutex*.

At the end, your threads must be joined properly and your program must terminate without any complications.

Details of Simulation

Before the simulation begins, some inputs (total 5 of them) are entered via keyboard. First, *total_customer_count* is entered. *total_customer_count* is the total number of customers who are going to arrive to the store and served during the simulation. In other words, the simulation will finish when *total_customer_count* customers complete their transactions. Moreover, you should not allow more than *total_customer_count* customers to arrive. Thus, at the end of simulation, all customers should be processed by the cashiers and the queues must be empty.

After the first input, you need to create your queues. `HW7IntQueue` class is a static one, meaning that its capacity does not change. In order to avoid dealing with full queues, you are required to set the capacity of your queues to exactly half of *total_customer_count* (you need to round up if *total_customer_count* is an odd number). This way, you do not have to check whether a queue is full in this homework.

Then, the parameters of the random customer inter-arrival time and random cashier checkout duration must be entered via keyboard. The time period between two customer arrivals is a random value in seconds. This random value is between two integer parameters: *min_arrival* and *max_arrival* (*min* and *max* values are included in the range). The cashier checkout time is also determined similarly; it is a random integer value between and including *min_checkout_time* and *max_checkout_time*. Of course, the random checkout time will be picked for each customer separately. You do not need to make any input checks for these parameters. You can assume that *min* values are always less than or equal to the corresponding *max* values and *min* values are greater than or equal to 1 second. You can use the `RandGen` class' `RandInt(`*`low, max`*`)` method from CS201 for random integer generation.

Simulation starts when the user enters all the inputs through keyboard. During the simulation, customers will arrive and cashiers will process customers in corresponding threads as explained before. Each new customer must be associated with a consecutive ID number starting with 1. You may simulate inter-arrival time of customers and checkout time of a customer by a cashier by sleeping the threads using the `this_thread::sleep_for(chrono::seconds(`*time_in_seconds*`))` command of `thread` (for `this_thread::sleep_for`) and `chrono` (for `chrono::seconds`) libraries. Moreover, you have to sleep the customer thread before the arrival of the first customer.

As mentioned before, there are two queues that a newly arriving customer can choose to be enqueued. Let us explain how a customer should decide on which queue to be enqueued. Assume two cashiers are called *A* and *B*. When a customer arrives, he/she first will check the sizes of the queues of the cashiers (there is a member function in `HW7IntQueue` class to return the size of the queue). If the queues are of different length, naturally the shorter queue will be chosen by the customer to be enqueued. Otherwise, the customer should check whether the cashiers are busy. If one of the cashiers is idle and the other one is busy, then the customer will be enqueued to the queue of the idle cashier.

Otherwise (i.e. if both of the cashiers are busy or both of them are idle), then the customer will choose Cashier *A*'s queue by default.

At the beginning of the program (after getting inputs and just before the beginning of the simulation), you have to display a message saying that the simulation is starting at the current time. During the simulation, customer thread should display the details of each customer's arrival. These details contain the ID of the customer, which queue the customer is enqueued, the queue size after enqueue operation and the time of arrival. Similarly, each cashier thread should display the details of the checkouts. At the beginning and at the end of a transaction, the tag of the cashier (*A* or *B*), the ID of the customer and the time of the operation should be displayed. Additionally, for the beginning of transaction, the size of the queue (after dequeuing) should be displayed as well.

At the end of the simulation, a message saying that the simulation is ended must be displayed. Please see the sample run section for some examples. Here please remark that a single line of output from a particular thread may be interleaved by the output of another thread if you do not take appropriate precautions. If this happens, the outputs mix up and become messy. You have to code appropriately, with the help of a dedicated mutex, in order not to end up with such an undesirable situation.

`HW7IntQueue` class and use of global variables

We provide a header file (HW7IntQueue.h) and its implementation (HW7IntQueue.cpp) together with this homework. **You have to use this queue class without any change**. In order to use it, include both the header and the .cpp to your project. Of course, you have to copy these files to appropriate folders in your project.

Finally a good news; you may use global variables in this homework. Actually, it would be miserable not to use them in a program that has several threads. However, we kindly request you not to exaggerate the global usage since after a certain point you may lose control over your program (as the famous Turkish proverb says "azı karar, coğu zarar").

Sample Runs

Some sample runs are given below, but these are not comprehensive, therefore you have to consider all cases, to get full mark.

Due to the probabilistic nature of the homework and due to scheduling of threads, same inputs may yield different outputs for your code. However, the order of the events must be consistent with the homework requirements and the given inputs. Nevertheless, occasional (i.e. rare) inconsistencies in the display order of the events occurred at the same time are acceptable.

The inputs from the keyboard are written in ***`boldface and italic`***.

Sample Run 1:

```
Please enter the total number of customers to be served: 10

Please enter the inter-arrival time range between two customers
min: 1
max: 1

Please enter the checkout time range of cashiers
min: 2
max: 2

Simulation begins at 23:36:04
```

```
Customer 1 has arrived at Cashier A's line (queue size is 1): 23:36:05
Cashier A started transaction with Customer 1 (queue size is 0): 23:36:05
Customer 2 has arrived at Cashier B's line (queue size is 1): 23:36:06
Cashier B started transaction with Customer 2 (queue size is 0): 23:36:06
Cashier A finished transaction with Customer 1: 23:36:07
Customer 3 has arrived at Cashier A's line (queue size is 1): 23:36:07
Cashier A started transaction with Customer 3 (queue size is 0): 23:36:07
Cashier B finished transaction with Customer 2: 23:36:08
Customer 4 has arrived at Cashier B's line (queue size is 1): 23:36:08
Cashier B started transaction with Customer 4 (queue size is 0): 23:36:08
Customer 5 has arrived at Cashier A's line (queue size is 1): 23:36:09
Cashier A finished transaction with Customer 3: 23:36:09
Cashier A started transaction with Customer 5 (queue size is 0): 23:36:09
Cashier B finished transaction with Customer 4: 23:36:10
Customer 6 has arrived at Cashier B's line (queue size is 1): 23:36:10
Cashier B started transaction with Customer 6 (queue size is 0): 23:36:10
Cashier A finished transaction with Customer 5: 23:36:11
Customer 7 has arrived at Cashier A's line (queue size is 1): 23:36:11
Cashier A started transaction with Customer 7 (queue size is 0): 23:36:11
Cashier B finished transaction with Customer 6: 23:36:12
Customer 8 has arrived at Cashier B's line (queue size is 1): 23:36:12
Cashier B started transaction with Customer 8 (queue size is 0): 23:36:12
Cashier A finished transaction with Customer 7: 23:36:13
Customer 9 has arrived at Cashier A's line (queue size is 1): 23:36:13
Cashier A started transaction with Customer 9 (queue size is 0): 23:36:13
Cashier B finished transaction with Customer 8: 23:36:14
Customer 10 has arrived at Cashier B's line (queue size is 1): 23:36:14
Cashier B started transaction with Customer 10 (queue size is 0): 23:36:14
Cashier A finished transaction with Customer 9: 23:36:15
Cashier B finished transaction with Customer 10: 23:36:16

Simulation ends
Press any key to continue . . .
```

Sample Run 2:

```
Please enter the total number of customers to be served: 8

Please enter the inter-arrival time range between two customers
min: 1
max: 2

Please enter the checkout time range of cashiers
min: 12
max: 15

Simulation begins at 11:20:08

Customer 1 has arrived at Cashier A's line (queue size is 1): 11:20:09
Cashier A started transaction with Customer 1 (queue size is 0): 11:20:09
Customer 2 has arrived at Cashier B's line (queue size is 1): 11:20:11
Cashier B started transaction with Customer 2 (queue size is 0): 11:20:11
Customer 3 has arrived at Cashier A's line (queue size is 1): 11:20:12
Customer 4 has arrived at Cashier B's line (queue size is 1): 11:20:14
Customer 5 has arrived at Cashier A's line (queue size is 2): 11:20:16
Customer 6 has arrived at Cashier B's line (queue size is 2): 11:20:17
Customer 7 has arrived at Cashier A's line (queue size is 3): 11:20:18
Customer 8 has arrived at Cashier B's line (queue size is 3): 11:20:20
Cashier A finished transaction with Customer 1: 11:20:21
Cashier A started transaction with Customer 3 (queue size is 2): 11:20:21
Cashier B finished transaction with Customer 2: 11:20:23
Cashier B started transaction with Customer 4 (queue size is 2): 11:20:23
Cashier A finished transaction with Customer 3: 11:20:35
Cashier A started transaction with Customer 5 (queue size is 1): 11:20:35
Cashier B finished transaction with Customer 4: 11:20:37
Cashier B started transaction with Customer 6 (queue size is 1): 11:20:37
```

```
Cashier A finished transaction with Customer 5: 11:20:47
Cashier A started transaction with Customer 7 (queue size is 0): 11:20:47
Cashier B finished transaction with Customer 6: 11:20:49
Cashier B started transaction with Customer 8 (queue size is 0): 11:20:49
Cashier A finished transaction with Customer 7: 11:21:02
Cashier B finished transaction with Customer 8: 11:21:04

Simulation ends
Press any key to continue . . .
```

Sample Run 3:

```
Please enter the total number of customers to be served: 12

Please enter the inter-arrival time range between two customers
min: 2
max: 3

Please enter the checkout time range of cashiers
min: 11
max: 13

Simulation begins at 11:26:00

Customer 1 has arrived at Cashier A's line (queue size is 1): 11:26:02
Cashier A started transaction with Customer 1 (queue size is 0): 11:26:02
Customer 2 has arrived at Cashier B's line (queue size is 1): 11:26:05
Cashier B started transaction with Customer 2 (queue size is 0): 11:26:05
Customer 3 has arrived at Cashier A's line (queue size is 1): 11:26:07
Customer 4 has arrived at Cashier B's line (queue size is 1): 11:26:10
Cashier A finished transaction with Customer 1: 11:26:13
Cashier A started transaction with Customer 3 (queue size is 0): 11:26:13
Customer 5 has arrived at Cashier A's line (queue size is 1): 11:26:13
Customer 6 has arrived at Cashier A's line (queue size is 2): 11:26:15
Cashier B finished transaction with Customer 2: 11:26:16
Cashier B started transaction with Customer 4 (queue size is 0): 11:26:16
Customer 7 has arrived at Cashier B's line (queue size is 1): 11:26:17
Customer 8 has arrived at Cashier B's line (queue size is 2): 11:26:20
Customer 9 has arrived at Cashier A's line (queue size is 3): 11:26:23
Cashier A finished transaction with Customer 3: 11:26:25
Cashier A started transaction with Customer 5 (queue size is 2): 11:26:25
Customer 10 has arrived at Cashier A's line (queue size is 3): 11:26:26
Cashier B finished transaction with Customer 4: 11:26:28
Cashier B started transaction with Customer 7 (queue size is 1): 11:26:28
Customer 11 has arrived at Cashier B's line (queue size is 2): 11:26:28
Customer 12 has arrived at Cashier B's line (queue size is 3): 11:26:31
Cashier A finished transaction with Customer 5: 11:26:36
Cashier A started transaction with Customer 6 (queue size is 2): 11:26:36
Cashier B finished transaction with Customer 7: 11:26:39
Cashier B started transaction with Customer 8 (queue size is 2): 11:26:39
Cashier A finished transaction with Customer 6: 11:26:49
Cashier A started transaction with Customer 9 (queue size is 1): 11:26:49
Cashier B finished transaction with Customer 8: 11:26:52
Cashier B started transaction with Customer 11 (queue size is 1): 11:26:52
Cashier A finished transaction with Customer 9: 11:27:01
Cashier A started transaction with Customer 10 (queue size is 0): 11:27:01
Cashier B finished transaction with Customer 11: 11:27:04
Cashier B started transaction with Customer 12 (queue size is 0): 11:27:04
Cashier A finished transaction with Customer 10: 11:27:13
Cashier B finished transaction with Customer 12: 11:27:16

Simulation ends
Press any key to continue . . .
```

Sample Run 4:

```
Please enter the total number of customers to be served: 11
```

```
Please enter the inter-arrival time range between two customers
min: 7
max: 8

Please enter the checkout time range of cashiers
min: 2
max: 4

Simulation begins at 11:28:56

Customer 1 has arrived at Cashier A's line (queue size is 1): 11:29:03
Cashier A started transaction with Customer 1 (queue size is 0): 11:29:03
Cashier A finished transaction with Customer 1: 11:29:05
Customer 2 has arrived at Cashier A's line (queue size is 1): 11:29:11
Cashier A started transaction with Customer 2 (queue size is 0): 11:29:11
Cashier A finished transaction with Customer 2: 11:29:14
Customer 3 has arrived at Cashier A's line (queue size is 1): 11:29:18
Cashier A started transaction with Customer 3 (queue size is 0): 11:29:18
Cashier A finished transaction with Customer 3: 11:29:20
Customer 4 has arrived at Cashier A's line (queue size is 1): 11:29:26
Cashier A started transaction with Customer 4 (queue size is 0): 11:29:26
Cashier A finished transaction with Customer 4: 11:29:30
Customer 5 has arrived at Cashier A's line (queue size is 1): 11:29:34
Cashier A started transaction with Customer 5 (queue size is 0): 11:29:34
Cashier A finished transaction with Customer 5: 11:29:37
Customer 6 has arrived at Cashier A's line (queue size is 1): 11:29:41
Cashier A started transaction with Customer 6 (queue size is 0): 11:29:41
Cashier A finished transaction with Customer 6: 11:29:44
Customer 7 has arrived at Cashier A's line (queue size is 1): 11:29:48
Cashier A started transaction with Customer 7 (queue size is 0): 11:29:48
Cashier A finished transaction with Customer 7: 11:29:51
Customer 8 has arrived at Cashier A's line (queue size is 1): 11:29:56
Cashier A started transaction with Customer 8 (queue size is 0): 11:29:56
Cashier A finished transaction with Customer 8: 11:30:00
Customer 9 has arrived at Cashier A's line (queue size is 1): 11:30:04
Cashier A started transaction with Customer 9 (queue size is 0): 11:30:04
Cashier A finished transaction with Customer 9: 11:30:08
Customer 10 has arrived at Cashier A's line (queue size is 1): 11:30:12
Cashier A started transaction with Customer 10 (queue size is 0): 11:30:12
Cashier A finished transaction with Customer 10: 11:30:16
Customer 11 has arrived at Cashier A's line (queue size is 1): 11:30:19
Cashier A started transaction with Customer 11 (queue size is 0): 11:30:19
Cashier A finished transaction with Customer 11: 11:30:21

Simulation ends
Press any key to continue . . .
```

Sample Run 5:

```
Please enter the total number of customers to be served: 9

Please enter the inter-arrival time range between two customers
min: 1
max: 8

Please enter the checkout time range of cashiers
min: 3
max: 4

Simulation begins at 11:35:22

Customer 1 has arrived at Cashier A's line (queue size is 1): 11:35:23
Cashier A started transaction with Customer 1 (queue size is 0): 11:35:23
Cashier A finished transaction with Customer 1: 11:35:26
Customer 2 has arrived at Cashier A's line (queue size is 1): 11:35:28
```

```
Cashier A started transaction with Customer 2 (queue size is 0): 11:35:28
Customer 3 has arrived at Cashier B's line (queue size is 1): 11:35:30
Cashier B started transaction with Customer 3 (queue size is 0): 11:35:30
Cashier A finished transaction with Customer 2: 11:35:32
Cashier B finished transaction with Customer 3: 11:35:33
Customer 4 has arrived at Cashier A's line (queue size is 1): 11:35:37
Cashier A started transaction with Customer 4 (queue size is 0): 11:35:37
Cashier A finished transaction with Customer 4: 11:35:40
Customer 5 has arrived at Cashier A's line (queue size is 1): 11:35:42
Cashier A started transaction with Customer 5 (queue size is 0): 11:35:42
Customer 6 has arrived at Cashier B's line (queue size is 1): 11:35:46
Cashier B started transaction with Customer 6 (queue size is 0): 11:35:46
Cashier A finished transaction with Customer 5: 11:35:46
Customer 7 has arrived at Cashier A's line (queue size is 1): 11:35:49
Cashier A started transaction with Customer 7 (queue size is 0): 11:35:49
Cashier B finished transaction with Customer 6: 11:35:50
Cashier A finished transaction with Customer 7: 11:35:53
Customer 8 has arrived at Cashier A's line (queue size is 1): 11:35:57
Cashier A started transaction with Customer 8 (queue size is 0): 11:35:57
Cashier A finished transaction with Customer 8: 11:36:00
Customer 9 has arrived at Cashier A's line (queue size is 1): 11:36:04
Cashier A started transaction with Customer 9 (queue size is 0): 11:36:04
Cashier A finished transaction with Customer 9: 11:36:07

Simulation ends
Press any key to continue . . .
```

Please see the previous homework specifications for the other important rules and the submission guidelines

Good Luck!
Albert Levi, Ömer Mert Candan