Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2020

Homework 8 – Simulation of Grocery Store Checkout Line using Threads

Due: 15/05/2020, 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 a queue of customers waiting in a store to pay their groceries. There are two (2) cashiers and a queue in the store, shared by these cashiers. One of the cashier always serves to the customers, while the other cashier serves when the current number of customers waiting in the queue passes a threshold value. When arrived, customers need to go to the rear of the queue 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 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 dynamic 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 a queue 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 checkout. During the simulation, you are going to display some verbose output about the actions of customers and cashiers (see Section "Details of Simulation" for details).

Using Threads

There will be three threads (other than the 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 enqueued to the queue. On the other hand, while one cashier always dequeues a customer from the queue, other cashier only dequeues when queue’s size passes a pre-determined threshold value, which is also an input. Then the cashier starts the checkout process, which also takes a random time period.

Please use the `HW8DynIntQueue` class given in this homework package as the queue data structure. Moreover, do not forget that each cashier and its thread uses same `HW8DynIntQueue` object for dequeuing. Also, enqueuing customers to this queue 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 queue. 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/crashes.

Details of Simulation

Before the simulation begins, some inputs (total 6 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 queue must be empty.

Second input is *cashier2_threshold* that determines when second cashier starts and stops to serve customers according to the current queue size. More specifically, as long as the current queue size is greater or equal to *cashier2_threshold*, second cashier serves by getting the next customer in the queue. When the queue size becomes less than *cashier2_threshold*, second cashier stops serving. Here remark that if queue size gets greater than or equal to the *cashier2_threshold* again at a later time, second cashier starts to serve again, and so on.

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 cashiers' 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. To generate random numbers for arrival and checkout time, **do not use the `RandGen` class from** CS201 for random integer generation in multithreaded applications because it is not thread-safe. Instead, **you must use the following thread-safe function, `random_range`, for generating random waiting times inside the threads**. This function returns a random number between (and including) `min` and `max` parameters. In the threads, you can call it by passing minimum and maximum waiting times of the corresponding thread as arguments.

```cpp
#include <random>
#include <time.h>

int random_range(const int & min, const int & max) {
    static mt19937 generator(time(0));
    uniform_int_distribution<int> distribution(min, max);
    return distribution(generator);
}
```

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.

Here we want to give a hint about second cashier's thread. Since the second cashier works in on/off manner due to the threshold logic explained above, you may think of starting and stopping its thread for the implementation of this logic. However, this would be too complicated and we are not expecting such a solution; it would be sufficient if you start the second cashier's thread at the beginning of the program (after getting the inputs) and implement the logic of threshold within the thread's entry point function.

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 cashier threads just after starting the thread functions as well.

At the beginning of the program (after getting the 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, 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 (1 or 2), 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 together with the current time. Please see the sample run section for some examples.

Two important rules about the simulation details and use of threads:

- 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 <u>dedicated</u> mutex, in order not to end up with such an undesirable situation. Here by "dedicated" we mean this; the mutex that you use for tidy output should not be the one that you use for accessing the shared queue object. These two critical sections are different concepts and must be handled via different mutexes.
- <u>Do not sleep a thread while a mutex is locked.</u> We use sleeping to simulate the arrival duration or checkout duration. We do not access shared resources while doing so. Thus, sleeping a thread while a mutex is locked is totally nonsense.
- <u>Any violation to these rules will be penalized severely (50 points or more)</u>.

`HW8DynIntQueue` class and use of global variables

We provide a header file (HW8DynIntQueue.h) and its implementation (HW8DynIntQueue.cpp) together with this homework. <u>**You have to use this queue class without any change**</u>. 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: 15
Please enter the number of customers waiting in the queue to open the second cashier:
2
Please enter the inter-arrival time range between two customers:
Min: 1
Max: 1
Please enter the checkout time range of cashiers:
Min: 1
Max: 1
Simulation starts 11:09:30
New customer with ID 1 has arrived (queue size is 1): 11:09:30
New customer with ID 2 has arrived (queue size is 2): 11:09:31
Cashier 1 started transaction with customer 1 (queue size is 1): 11:09:31
New customer with ID 3 has arrived (queue size is 2): 11:09:32
Cashier 2 started transaction with customer 2 (queue size is 1): 11:09:32
Cashier 1 finished transaction with customer 1 11:09:32
Cashier 1 started transaction with customer 3 (queue size is 0): 11:09:32
New customer with ID 4 has arrived (queue size is 1): 11:09:33
Cashier 2 finished transaction with customer 2 11:09:33
Cashier 1 finished transaction with customer 3 11:09:33
Cashier 1 started transaction with customer 4 (queue size is 0): 11:09:33
New customer with ID 5 has arrived (queue size is 1): 11:09:34
Cashier 1 finished transaction with customer 4 11:09:34
Cashier 1 started transaction with customer 5 (queue size is 0): 11:09:34
New customer with ID 6 has arrived (queue size is 1): 11:09:35
Cashier 1 finished transaction with customer 5 11:09:35
Cashier 1 started transaction with customer 6 (queue size is 0): 11:09:35
New customer with ID 7 has arrived (queue size is 1): 11:09:36
Cashier 1 finished transaction with customer 6 11:09:36
Cashier 1 started transaction with customer 7 (queue size is 0): 11:09:36
New customer with ID 8 has arrived (queue size is 1): 11:09:37
Cashier 1 finished transaction with customer 7 11:09:37
Cashier 1 started transaction with customer 8 (queue size is 0): 11:09:37
New customer with ID 9 has arrived (queue size is 1): 11:09:38
Cashier 1 finished transaction with customer 8 11:09:38
Cashier 1 started transaction with customer 9 (queue size is 0): 11:09:38
New customer with ID 10 has arrived (queue size is 1): 11:09:39
Cashier 1 finished transaction with customer 9 11:09:39
Cashier 1 started transaction with customer 10 (queue size is 0): 11:09:39
New customer with ID 11 has arrived (queue size is 1): 11:09:40
Cashier 1 finished transaction with customer 10 11:09:40
Cashier 1 started transaction with customer 11 (queue size is 0): 11:09:40
New customer with ID 12 has arrived (queue size is 1): 11:09:41
Cashier 1 finished transaction with customer 11 11:09:41
Cashier 1 started transaction with customer 12 (queue size is 0): 11:09:41
New customer with ID 13 has arrived (queue size is 1): 11:09:42
Cashier 1 finished transaction with customer 12 11:09:42
Cashier 1 started transaction with customer 13 (queue size is 0): 11:09:42
```

```
New customer with ID 14 has arrived (queue size is 1): 11:09:43
Cashier 1 finished transaction with customer 13 11:09:43
Cashier 1 started transaction with customer 14 (queue size is 0): 11:09:43
New customer with ID 15 has arrived (queue size is 1): 11:09:44
Cashier 1 finished transaction with customer 14 11:09:44
Cashier 1 started transaction with customer 15 (queue size is 0): 11:09:44
Cashier 1 finished transaction with customer 15 11:09:45
End of the simulation ends: 11:09:45
Press any key to continue . . .
```

Sample Run 2:

```
Please enter the total number of customers: 15
Please enter the number of customers waiting in the queue to open the second cashier:
5
Please enter the inter-arrival time range between two customers:
Min: 1
Max: 3
Please enter the checkout time range of cashiers:
Min: 1
Max: 8
Simulation starts 11:14:28
New customer with ID 1 has arrived (queue size is 1): 11:14:28
New customer with ID 2 has arrived (queue size is 2): 11:14:31
Cashier 1 started transaction with customer 1 (queue size is 1): 11:14:32
New customer with ID 3 has arrived (queue size is 2): 11:14:32
New customer with ID 4 has arrived (queue size is 3): 11:14:35
New customer with ID 5 has arrived (queue size is 4): 11:14:36
Cashier 1 finished transaction with customer 1 11:14:37
Cashier 1 started transaction with customer 2 (queue size is 3): 11:14:37
New customer with ID 6 has arrived (queue size is 4): 11:14:37
New customer with ID 7 has arrived (queue size is 5): 11:14:40
Cashier 2 started transaction with customer 3 (queue size is 4): 11:14:40
New customer with ID 8 has arrived (queue size is 5): 11:14:43
New customer with ID 9 has arrived (queue size is 6): 11:14:44
Cashier 1 finished transaction with customer 2 11:14:45
Cashier 1 started transaction with customer 4 (queue size is 5): 11:14:45
Cashier 2 finished transaction with customer 3 11:14:46
Cashier 2 started transaction with customer 5 (queue size is 4): 11:14:46
Cashier 1 finished transaction with customer 4 11:14:47
Cashier 1 started transaction with customer 6 (queue size is 3): 11:14:47
New customer with ID 10 has arrived (queue size is 4): 11:14:47
New customer with ID 11 has arrived (queue size is 5): 11:14:49
Cashier 1 finished transaction with customer 6 11:14:52
Cashier 1 started transaction with customer 7 (queue size is 4): 11:14:52
New customer with ID 12 has arrived (queue size is 5): 11:14:52
Cashier 1 finished transaction with customer 7 11:14:53
Cashier 1 started transaction with customer 8 (queue size is 4): 11:14:53
Cashier 2 finished transaction with customer 5 11:14:53
New customer with ID 13 has arrived (queue size is 5): 11:14:54
Cashier 2 started transaction with customer 9 (queue size is 4): 11:14:54
Cashier 1 finished transaction with customer 8 11:14:56
Cashier 1 started transaction with customer 10 (queue size is 3): 11:14:56
New customer with ID 14 has arrived (queue size is 4): 11:14:56
New customer with ID 15 has arrived (queue size is 5): 11:14:58
Cashier 2 finished transaction with customer 9 11:15:00
Cashier 2 started transaction with customer 11 (queue size is 4): 11:15:00
Cashier 2 finished transaction with customer 11 11:15:03
Cashier 1 finished transaction with customer 10 11:15:04
Cashier 1 started transaction with customer 12 (queue size is 3): 11:15:04
Cashier 1 finished transaction with customer 12 11:15:11
Cashier 1 started transaction with customer 13 (queue size is 2): 11:15:11
Cashier 1 finished transaction with customer 13 11:15:18
Cashier 1 started transaction with customer 14 (queue size is 1): 11:15:18
Cashier 1 finished transaction with customer 14 11:15:19
Cashier 1 started transaction with customer 15 (queue size is 0): 11:15:19
Cashier 1 finished transaction with customer 15 11:15:22
End of the simulation ends: 11:15:22
Press any key to continue . . .
```

Sample Run 3:

```
Please enter the total number of customers: 1
Please enter the number of customers waiting in the queue to open the second cashier:
1
Please enter the inter-arrival time range between two customers:
Min: 1
Max: 1
Please enter the checkout time range of cashiers:
Min: 1
Max: 1
Simulation starts 11:18:41
New customer with ID 1 has arrived (queue size is 1): 11:18:41
Cashier 1 started transaction with customer 1 (queue size is 0): 11:18:42
Cashier 1 finished transaction with customer 1 11:18:43
End of the simulation ends: 11:18:43
Press any key to continue . . .
```

Sample Run 4:

```
Please enter the total number of customers: 1
Please enter the number of customers waiting in the queue to open the second cashier:
1
Please enter the inter-arrival time range between two customers:
Min: 1
Max: 1
Please enter the checkout time range of cashiers:
Min: 1
Max: 1
Simulation starts 11:19:05
New customer with ID 1 has arrived (queue size is 1): 11:19:05
Cashier 2 started transaction with customer 1 (queue size is 0): 11:19:06
Cashier 2 finished transaction with customer 1 11:19:07
End of the simulation ends: 11:19:07
Press any key to continue . . .
```

Sample Run 5:

```
Please enter the total number of customers: 15
Please enter the number of customers waiting in the queue to open the second cashier:
1
Please enter the inter-arrival time range between two customers:
Min: 2
Max: 3
Please enter the checkout time range of cashiers:
Min: 1
Max: 1
Simulation starts 12:09:46
New customer with ID 1 has arrived (queue size is 1): 12:09:46
Cashier 2 started transaction with customer 1 (queue size is 0): 12:09:47
New customer with ID 2 has arrived (queue size is 1): 12:09:48
Cashier 1 started transaction with customer 2 (queue size is 0): 12:09:48
Cashier 2 finished transaction with customer 1 12:09:48
Cashier 1 finished transaction with customer 2 12:09:49
New customer with ID 3 has arrived (queue size is 1): 12:09:51
Cashier 1 started transaction with customer 3 (queue size is 0): 12:09:51
Cashier 1 finished transaction with customer 3 12:09:52
Cashier 2 started transaction with customer 4 (queue size is 0): 12:09:53
New customer with ID 4 has arrived (queue size is 1): 12:09:53
Cashier 2 finished transaction with customer 4 12:09:54
New customer with ID 5 has arrived (queue size is 1): 12:09:56
Cashier 2 started transaction with customer 5 (queue size is 0): 12:09:56
Cashier 2 finished transaction with customer 5 12:09:57
New customer with ID 6 has arrived (queue size is 1): 12:09:58
Cashier 2 started transaction with customer 6 (queue size is 0): 12:09:58
Cashier 2 finished transaction with customer 6 12:09:59
New customer with ID 7 has arrived (queue size is 1): 12:37:37
Cashier 1 started transaction with customer 7 (queue size is 0): 12:37:37
Cashier 1 finished transaction with customer 7 12:37:38
New customer with ID 8 has arrived (queue size is 1): 12:37:39
Cashier 1 started transaction with customer 8 (queue size is 0): 12:37:39
Cashier 1 finished transaction with customer 8 12:37:40
New customer with ID 9 has arrived (queue size is 1): 12:37:42
Cashier 1 started transaction with customer 9 (queue size is 0): 12:37:42
Cashier 1 finished transaction with customer 9 12:37:43
Cashier 1 started transaction with customer 10 (queue size is 0): 12:37:44
New customer with ID 10 has arrived (queue size is 1): 12:37:44
Cashier 1 finished transaction with customer 10 12:37:45
New customer with ID 11 has arrived (queue size is 1): 12:37:46
Cashier 1 started transaction with customer 11 (queue size is 0): 12:37:46
Cashier 1 finished transaction with customer 11 12:37:47
New customer with ID 12 has arrived (queue size is 1): 12:37:49
Cashier 2 started transaction with customer 12 (queue size is 0): 12:37:49
Cashier 2 finished transaction with customer 12 12:37:50
New customer with ID 13 has arrived (queue size is 1): 12:37:52
Cashier 1 started transaction with customer 13 (queue size is 0): 12:37:52
Cashier 1 finished transaction with customer 13 12:37:53
New customer with ID 14 has arrived (queue size is 1): 12:37:55
Cashier 2 started transaction with customer 14 (queue size is 0): 12:37:55
Cashier 2 finished transaction with customer 14 12:37:56
New customer with ID 15 has arrived (queue size is 1): 12:37:58
Cashier 2 started transaction with customer 15 (queue size is 0): 12:37:58
Cashier 2 finished transaction with customer 15 12:37:59
End of the simulation ends: 12:38:00
Press any key to continue . . .
```

As you can see, there are inconsistencies here in the display order of the events occurred at the same time. As mentioned before, such occasional inconsistencies are acceptable.

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

Last, but not the least, you are highly recommended to use Windows for this homework. In multithreaded applications, there are several dependencies to the underlying operating system. Thus the behavior that you see while running in a Windows computer could be totally different than MacOS. Your code will be tested in Windows and if it does not work, even if it works using Mac, we may not re-evaluate your code due to hectic grading period at the end of the semester. To sum up, use Mac at your own risk.

Good Luck!
Albert Levi, Vedat Peran