

Programming Assignment 3

Submission Date : 25.11.2022

Due Date : 09.12.2022 (23:59)

Programming Languages: C++

Subject : Implementing HUCoffeeShop by using Queue

Advisors : [REDACTED]

1 Introduction

Heraclitus once said “The only constant in life is change”. Heraclitus was a wise man, indeed, but he missed an exception. If you take BBM 203, you solve a Discrete Event Simulation (DES) project!

Well, but what is DES?

Enter, Wikipedia: “A discrete-event simulation (DES) models the operation of a system as a (discrete) sequence of events in time. Each event occurs at a particular instant in time and marks a change of state in the system. Between consecutive events, no change in the system is assumed to occur; thus the simulation time can directly jump to the occurrence time of the next event”. In other words, DES is the representation of a certain process by simulating certain events which occur at certain times. Hence, time is not continuous in DES and progress according to the events. As a demonstrative example, a process of airplane boarding can be simulated as follows:

- TIME: 09:00 Go into the luggage queue.
- TIME: 09:15 Give your luggage and take your boarding pass.
- TIME: 09:20 Go into the security queue.
- TIME: 09:30 Wait for your boarding time and enjoy your free time.
- TIME: 10:00 Go into the boarding queue.
- TIME: 10.30 Have a nice flight!

Note that in DES the only things that count are events and everything else is ignored. In the case of the passenger, until 10:31 only 6 things happened. Additionally, the time progressed in a discrete fashion (i.e. jumped from one event to the next one).

In this experiment, you are expected to simulate HUCoffeeShop, which splits and executes tasks simultaneously in a way (by using queue data structure) that will not affect the outcome.

2 EXPERIMENT

Imagine a coffee shop with only one employee, a cashier who is supposed to do all the work alone. Gets an order, makes the coffee, delivers it to the customer, and repeats.. which may

take forever without concurrency. But having one more employee as a barista would really help, right? The cashier would gather orders and pass them to the barista, so once the barista was preparing an order, the cashier could get other orders, and the customers would not have to wait there all the time. The coffee shop would sell more coffee eventually.

You are a businessman who wants to build a new HUCoffeeShop in Hacettepe. There are two types of employees in HUCoffeeShop, one of them is cashier who takes the order of the customer and the other one is barista who prepares the coffee for the customer.

The HUCoffeeShop is available in two different models, each model with N cashiers and $N/3$ baristas. First model is shown in Figure 1. As shown in the Figure, there is one queue for both Cashier and Barista.

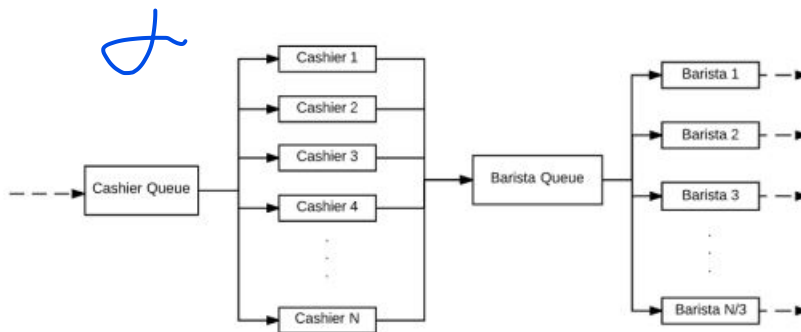


Figure 1: First Model in HUCoffeeShop

For the second model, there is one queue for cashiers and $N/3$ queues for baristas as shown in Figure 2. First 3 cashiers send the orders to the first barista and next 3 cashiers send the orders to the second barista and goes on like that.

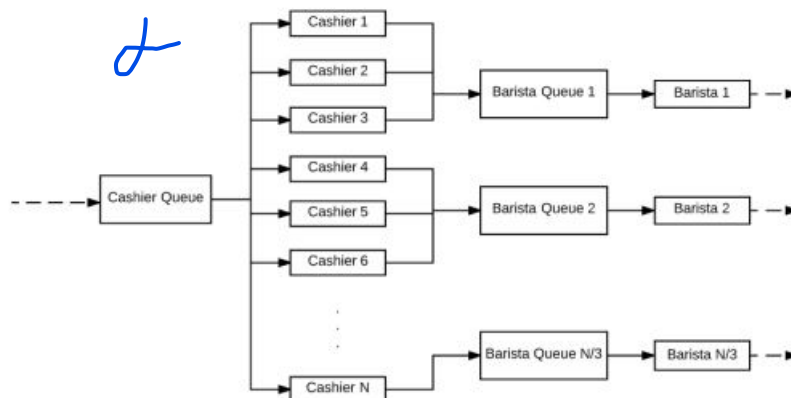


Figure 2: Second Model in HUCoffeeShop

In the cashier queue it is first come first serve, while in the barista queue the most expensive coffee is served first. This design is valid for both models.

In this experiment, we expect you simulating these two models shown in Figure 1 and Figure 2 and get statistics given below:

1. Total running time (for all system)
2. Utilization (for both cashier and barista)
3. Turnaround time (for each order)
4. Maximum length of each queue

3 IMPLEMENTATION DETAILS

1. Whenever a cashier or barista finishes his/her job, he/she immediately fetches an order waiting in corresponding queue. If the queue is empty, he/she goes idle.
2. If more than one cashier is available, the cashier with smallest ID takes the order. For example, if cashier1 and cashier3 are available at the same time, the next order will be taken by cashier1. This principle is same with baristas, too.
3. If any statistics have more than 2 decimal places, you need to write the number to the output file with 2 decimal values with rounding. You will use even rounding. (You can use `cout("%.2lf", variable name)` to print double variables.)
4. If you can't implement a certain statistic value, you should write "-1" in place of that statistics to the output file. This way you can implement other statistics and get points.
5. Unit Utilization = (Busy Time of the Unit)/(Total Running Time of The Coffee Shop)
6. Turnaround Time = (Time when the coffee is done)-(Arrival time of the customer)
7. Total Running Time = Time when all coffee orders are done
8. Your program will be compiled with "make". So, do not forget to create your own Makefile.
9. I will execute your program with `./Assignment3 inputFile outputFile` command. So, use command line arguments in your main function accordingly.

4 INPUT/OUTPUT FILE FORMAT

The input file has the following format:

1. First line is number of cashiers (N) which is always divisible by 3.
2. Second line is the number of orders (X).
3. Following X lines contain information about orders. There are 4 variables separated with space. They are:
 - Arrival Time: The time when customer enters to the coffee shop (in seconds)
 - Order Time: The time required to give coffee order (in seconds)
 - Brew Time: The time required to make coffee (in seconds)

- Price of Order: Price of the coffee.

+

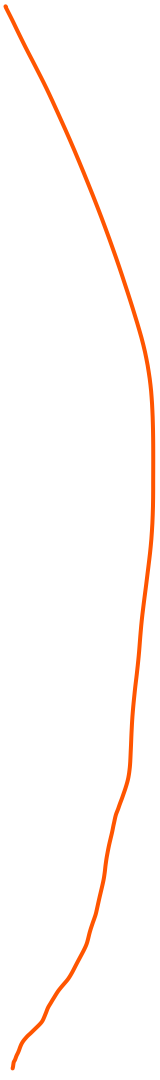

```
6
10
0 20.75 24.89 10.30
3.21 22.47 18.12 7.57
5.17 19.83 32.35 13.93
8.16 21.10 11.32 25.95
9.54 26.05 17.14 8.15
10.32 11.10 14.08 3.74
16.47 34.68 41.41 16.98
23.13 17.33 16.22 9.86
27.52 40.68 23.43 20.97
28.08 16.80 30.47 11.32
```

Figure 3: Sample input for HUCoffeeShop Simulation

The output file has the following format:

1. First line is the total running time of the coffee shop (in seconds).
2. Next line is the maximum length of the cashier queue.
3.
 - If the coffee shop has one barista queue, next line is maximum length of this queue.
 - If the coffee shop has $N/3$ barista queues, next $N/3$ lines are maximum lengths of each barista queue.
4. Next N lines are the utilization of the cashiers.
5. Next $N/3$ lines are the utilization of the baristas.
6. Next X lines are turnaround times of orders.

You will print these values for first and second models, respectively. You will put an empty line between models. Example of input and output files can be seen in the Figure 3 and Figure 4.



```
137.52  
1  
5  
0.40  
0.46  
0.27  
0.15  
0.19  
0.21  
0.85  
0.82  
45.64  
134.31  
72.82  
38.66  
124.54  
25.18  
102.93  
93.81  
73.20  
49.21  
  
191.42  
1  
4  
2  
0.29  
0.33  
0.19  
0.11  
0.14  
0.15  
0.89  
0.31  
45.64  
188.21  
72.82  
38.66  
70.64  
25.18  
126.36  
39.91  
73.90  
145.22
```

total running time

Figure 4: Sample output for HUCoffeeShop Simulation

4.1 Execution

The name of the compiled executable program should be “Assignment3”. Your program should read input/output file names from the command line, so it will be executed as follows:

Assignment3 [input file name] [output file name] e.g. ./Assignment3 input1.txt output1.txt

You can see sample input and output in piazza page. The program must run on DEV (dev.cs.hacettepe.edu.tr) UNIX machines. So make sure that it compiles and runs on dev server. If we are unable to compile or run, the project risks getting zero point. It is recommended that you test the program using the same mechanism on the sample files (provided) and your own inputs. You must compare your own output and sample output. If your output

is different from the sample, the project risks getting zero point, too.

4.2 Design Expectations

Your program will be graded based the correctness of your output and the clarity of the source code. Correctness of your output will be tested automatically so make sure you stick with the format described in the pdf. **You MUST implement your own queue classes, don't use other data structures such as array, string, arraylist, linkedlist etc in your implementation. If you use other data structures, you will not get any point (your grade will be 0).**

4.3 Required Files and Submit Format

Do not submit any file via e-mail. You should upload your files via “Online Experiment Submission System” which is at <http://submit.cs.hacettepe.edu.tr>.

You should create and submit a ZIP archive in the following structure for evaluation. An invalid structured archive will cause you partial or full score loss. You are required to submit a Makefile, which will be used to compile your program to generate the Assignment3 executable.

- student id.zip < *.cpp, *.h, Makefile >

5 Grading Policy

4

Task	Point
Compiled	10
Design	10
Output	80
Total	100

P.S. The Compiled part represents the note that will break if there is a compilation error and your code works after the small fix.

6 Notes

- The assignment must be original, individual work. Downloaded or modified source codes will be considered as cheating. Also the students who share their works will be punished in the same way.
- We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism. Our department takes the act of plagiarism very seriously. Those caught plagiarizing (both originators and copiers) will be sanctioned.
- You can ask your questions through course's piazza group and you are supposed to be aware of everything discussed in the piazza group. General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes and reports .

- Questions asked over the Piazza will be answered within working hours on weekdays. On weekends, we will try to answer as much as possible, but in case of possible unavailability, please do not insist on getting an answer.
- Ignore the cases which are not stated in this assignment and do not ask questions on Piazza for such extreme cases.
- Don't forget to write comments of your codes when necessary.
- Do not submit your project without first compiling it on dev machine. Make sure that the source code you submitted is compiled with gcc compiler under Linux OS. Applications that produce compilation or runtime errors cannot be graded.
- Save all work until the assignment is graded.
- Do not miss the deadline. Submission will be end at 09/12/2022 23:59, the system will be open until 23:59:59. The problem about submission after 23:59 will not be considered.
- Your grades will be announced within 15 days at the latest. Objections about your grades can be made within specified days given by TAs. Objections made outside of the specified days will not be accepted.