**CS 101** **Project 3** **Summer 2017**

Write a C++ program project3.cpp that operates as described below. The program will provide a simulation of customers who are waiting for service at a business such as a supermarket, bank, restaurant, airport, or doctor's office.

A "server" is an employee who waits on a customer, such as a checker at a supermarket, or a teller at a bank. First your program will read two command line arguments: the number of servers, and an integer from 0 to 9 that denotes a simulation option. Simulation options are discussed below.

Next the program will read data from standard input. Each line of input contains a customer id number, the customer's arrival time, and a duration that indicates how much time the customer needs (which may be proportional to a number of items or transactions). Time units are arbitrary, but it might be helpful to think in terms of minutes. You may assume that the arrival times are in ascending order. Here is an example:

| Customer | Arrival | Duration |
|---|---|---|
| 0 | 0 | 3 |
| 1 | 2 | 5 |
| 2 | 4 | 6 |
| 3 | 5 | 4 |
| 4 | 5 | 1 |
| 5 | 6 | 2 |
| 6 | 6 | 5 |
| 7 | 6 | 3 |

Next the program will perform a simulation for each option 0 to 9, and write the results to standard output. The output will consist of a schedule of customers in the order they begin to receive service. The output values will vary depending on the option being simulated. Display at least the information shown in the following example, which uses 3 servers and the input data given above.

```
Option 0:

Server Customer Start Finish Wait
   0       0      0      3     0
   1       1      2      7     0
   0       2      4     10     0
   2       3      5      9     0
   1       4      7      8     2
   1       5      8     10     2
   2       6      9     14     3
   0       7     10     13     4

Total Wait = 11
Average Wait = 1.375000
Longest Wait = 4
Latest Finish = 14
```

This project is open-ended, so you can choose any reasonable simulation options to implement. Here are some possible factors to consider:

- The next customer to receive service may be whoever has been waiting the longest time, or whoever has been waiting the shortest time, or whoever needs the longest duration of service, or whoever needs the shortest duration. Or perhaps implement some combination of these criteria, or just select a customer at random.
- All customers may be placed into a single waiting line (such as at most banks), or there may be a separate waiting line corresponding to each server (such as at most supermarkets). Also there may be special requirements based on durations for entering certain lines (such as supermarket express lanes). Also customers might be permitted to switch from one line to another to maintain balanced lengths.
- Some options should correspond to standard practices that businesses use in the real world. Also, try to be creative and invent some other non-standard options that might work well. You can also include some options that do not work well.

The goals include minimizing the total wait, average wait, longest wait, and latest finish times. (You may also include other desirable goals if they are measurable.) However, sometimes achieving a decrease in one of these values may cause an increase in another value. The best strategy is not clear, so your job is to design and implement some reasonable options, and perform experiments to reach some conclusions and make some recommendations.

Submit your C++ program, some example input files and output files for each simulation option, and a written report in PDF format. The report should describe your chosen simulation options 0 to 9, summarize the results of your experiments, and also provide any conclusions or recommendations regarding which options are preferable.

Your project will be evaluated based on factors that include originality, correctness, and presentation. Because the project is open-ended, grading will necessarily include a subjective component.

Please carefully read the following requirements:

- You must do your own work. You must not borrow any code from any other person, book, website, or any other source. You also must not share your code with any other person, or post it on any website. We run plagiarism detection software on every project. So if you violate these rules, you may receive an invitation to the dean's office to discuss the penalties for academic misconduct.
- Because a purpose of this course is to learn how to properly implement data structures and algorithms, it is not permitted in general to include STL libraries such as <list>, <vector>, <stack>, <queue>, <deque>, <algorithm>, <numeric>, <utility>, … in your program. You are always permitted to use <iostream>, <string>, and any C libraries such as <cstdlib> and <cmath>. But do not include any other libraries unless explicitly allowed on the assignment.
- Make sure your program runs properly on the cs-intro.ua.edu server, because this is where your program will be graded. In particular, make sure your program initializes the values of all

variables when they are declared or allocated. Otherwise it might behave differently on Linux than it does on a PC or Mac.

- Your program will be compiled using this command: g++ project3.cpp –Wall –lm –std=c++11. Alternatively, if you split your program into multiple files, then it will instead be compiled using this command: g++ *.cpp –Wall –lm –std=c++11.
- Verify that all the necessary .h and .cpp files which are needed to compile your program are included in the same directory before you submit your program. There should be no extra subdirectories and no extra .h or .cpp files, otherwise your program might not compile the way you intended.
- Compress your project into a zip file that contains your C++ program source file. Right-click (or secondary click) on your project directory, and then (depending on your operating system) select either the Compress option or Send To → Compress from the popup menu. Finally upload your .zip file that contains your .cpp file for this project to Blackboard.
- If you violate the above requirements such that it breaks our grading script, your project will be assessed a significant point deduction, and extreme or multiple violations may cause the project to be considered ungradable.
- Every semester many students lose some points because they don't follow all the instructions. So please read and follow all the project specifications precisely to prevent losing points unnecessarily. If anything is unclear, please ask for clarification well before the project is due. Please pay particular attention to input and output formats.
- Submit your project on Blackboard by the due date (11:59pm Friday). There is a grace period of 24 hours (until 11:59pm Saturday). Projects submitted on Sunday will be assessed a late penalty of 5% per hour. No projects will be accepted after Sunday. Once it is graded, your project score will be posted on Blackboard and the results of the grading script will be sent to your Crimson email account.
- Double-check and triple-check your submission when you submit it. Errors discovered later cannot be fixed and resubmitted after the project is graded. Projects will not be re-graded unless an error is found in the grading script or in the input/output files that are used during grading.