CSCI203 ASSIGNMENT TWO

Due 23:55 Sunday 17/09/2023

Must be submitted via Moodle

Task:

You should write a program to simulate the queuing and service of bank customers. The simulation should be discrete event driven.

Your program should:

1. Read in the number of servers/tellers for the simulation,
2. Read in the file name from standard input and then open the file,
3. Read data record by record from the named file,
4. Perform the simulation according to the input records, and
5. Output the statistics of the services as specified below.

The input text file (**as-sample.txt**) contains the following data:

1. A set of customers, one customer per line and each line consisting of a customer's arrival time, the corresponding service time, and the customer's priority (1 for low, 2 for normal, 3 for high).
2. The set of customers is sorted in the ascending order of their arrival times.
3. This set is terminated by a dummy record with the arrival time and the service time all equal to 0.

The simulation is to be a bank branch with **multiple tellers** (i.e., servers) and a **single queue**. The simulation should be run as follows:

1. Setup the total number of tellers;
2. Read the text file to start the simulation;
3. Allocate the new arrival customer to the next idle tellers (if applicable);
4. If all tellers are busy, add the new arrival customer to the queue;
5. The queue shall be updated based on two criteria of the customers, i.e., their arrival times and their priorities. For the customers with the same priority level, the customers who arrive earlier shall be put in front of the customers who arrive late. However, the customers with a higher priority shall be put in front of the customers with a lower priority regardless of their arrival times. For example, if we have three customers, i.e., c1 (2.8, 4.8, 2), c2 (1.5, 3.1, 2), and c3 (3, 5.6, 3), then their order in the queue should be 'c3, c2, c1', which means the service order for thes three customers should be c3,c2,c1.
6. If a busy teller finishes a service request, then the teller will serve the first customer in the queue (the customer will be removed from the queue). If the queue is empty, then the busy teller will become idle;
7. The simulation should be run until the queue is empty and the last customer has left the system.

The simulation will run multiple times to gather statistics on the efficiency of the service with a different number of tellers on duty. You are required to run the simulation three times with the number of servers being set to **1 teller**, **2 tellers** and **4 tellers**, respectively. Each simulation runs on the sample input file and outputs the statistics as specified below.

Output for each run will consist of the following data:

- Number of customers served by each teller;
- Total time of the simulation, i.e., the time difference between the simulation starts and the simulation ends;
- Average service time per customer (this excludes the time spent in the queue);
- Average waiting time per customer (this excludes the time spent in the service);
- The maximum length of the queue;
- The average length of the queue (this can be estimated as the ratio between the total queuing time and the time the last service request is completed);

- The idle rate of each teller (this can be calculated as the ratio between the total idle time of each teller and the total time of the simulation).

Implementation Requirements:

1. Part of the purpose of this subject is to gain an in-depth understanding of data structures and algorithms. As such, all programming tasks in this subject require you to choose appropriate data structures and algorithms and implement them yourself.
2. You may use any data structures or algorithms that have been presented in class. If you use other data structures or algorithms appropriate references must be provided.
3. The pseudocode should be provided first to explain the principle of your program.
4. Programs must be compiled and run. Programs which do not compile and run will lose marks.
5. Programs should be appropriately explained with comments.
6. All coding must be your own work.
7. **You are only allowed to include or import input, output, and file streams in the beginning of your program. That is, you should only use built-in data structures and functions to implement your program. For instance, if you want to use data structure heap and functions related with heap in your program, you should implement heap and the relevant functions by using built-in array instead of calling heap from existing library. Specifically, you should start your program like this:**

   7.1 If you are to use C++ to program, the head should be

   ```
   #include<iostream>
   #include<fstream>
   using namespace std;
   ```

   7.2 For Java,

   ```
   import java.io.File;
   import java.io.FileNotFoundException;
   import java.util.Scanner;
   ```

   7.3 If you use Python,

   ```
   import sys
   import numpy
   ```

8. Use of STL, Java Collection, collection framework and any third-party libraries of data structures and algorithms in your language choice is NOT allowed.
9. If you use any references other than the lecture notes, ensure you cite them. Otherwise, you may lose marks. A clear comment in your code is sufficient.
10. Code sourced from textbooks, the internet, etc may also not be used. Otherwise, you may lose marks.
11. If needed, you may assume the queue will never have more than 100 customers.

Report:

A pdf file describing your solution and program output should be produced. This file should contain:

1. A high-level description (in pseudo-code) of the overall solution strategy.
2. A complexity analysis of your solution with big-O notation and sufficient justification.
3. A list of all of the data structures used, and the reasons for using them.
4. A snapshot of the compilation and the execution of your program on the provided "`a2-sample.txt`" file.
5. The outputs (three runs in total) are produced by your program on the provided "`a2-sample.txt`" file.
6. A discussion based on the statistics that how the teller's number affect the efficiency of the services.
7. The report pdf file should be called `<your login>-a2.pdf`

Submission via Moodle:

- Please submit your source code and the pdf report as a zip file (named as `<your login>-a2.zip`) to CSCI203 Moodle site (Assignment 2 submission folder) before the deadline.
- Please note that the email submission will **NOT** be accepted.
- If an extension (maximally 1 week) is required, please submit an academic consideration via SOLS **before** the deadline.
- Late submission will receive 25% penalty per 24 hours and a zero mark after 4 days.

Marking Guide:

- Programs submitted must work (can be compiled and executed)! A program which fails to compile or run will lose marks.
- If your program produces different output from what is reported in the pdf file, a mark of zero will be graded.
- A program which produces the correct output, no matter how inefficient the code, will receive a minimum of 50% of the program component of the mark.
- Additional marks beyond this will be awarded for the appropriateness, i.e. efficiency for this problem, of the algorithms and data structures you use.
- Programs which lack clarity, both in code and comments, will lose marks. The total mark will be determined based on both your code and the accompanying design pdf document.