A. Bari
New York University
Computer Science Department
Data Structures
Dr. Anasse Bari

Homework Four: Serving Customers

Deadline: *See NYUclasses for the deadline, 15% off per day after the deadline (4 days maximum).*

Learning Objectives:

- Implementing Queues as a LinkedList

Read the guidelines bellow carefully to avoid receiving a zero grade on the HW:

- Important: **MAKE SURE YOUR CODE COMPILES. FAILURE OF COMPILATION WILL RESULT IN AN IMMIDIATE ZERO.**
- **As specified in the homework description, we will compile your program using command line. Make sure you compile it using command line before you submit it**
- Attach the Java source files and include them into HW's zip file. The file name should be **YourLastName_HW4.zip**
- Make an archive (zip file or compressed file) with all the **java files (the .java files NOT the .class files)** and post it on NYU Classes
- You must comment you code (basic comments explaining the role of a class, a method or variables used in your submission)
- Compile and run the program before you submit.
- It is your responsibility to make sure if the Zip files has your actual latest files. You may send the file to yourself by email to double check that is the actual file before you upload on NYU classes.
- **If the graders cannot open the file, you will receive a grade of zero.**
- **If you send the .class files instead of the .java files (source files) you will receive a zero.**
- An act of cheating will be severely addressed with an immediate zero on the homework and a report to the academic advisor and the administration.
- You will automatically lose 50% of the points for an exercise if the program does not compile and run correctly.

- **Plagiarized assignments will get a ZERO grade**. You cannot change the variable names of other student's solution and submit it as yours. The program structure of other students must not match yours. Every student must come up with his/her own solution. Any cheating (e.g. copying from internet without citing sources) is a serious violation of the University student code.
- **Homeworks sent by email to the instructor or to the graders will NOT be reviewed and will not be accepted.**

Problem Statement:

This assignment involves modeling a group of customers and a service-counter on a typical day from 9am to 5pm, using queues (*implemented as LinkedList using Generic Types, you must have attended the recitations before this assignment was assigned)*

Different customers come to the service counter at different times to get some service (get their gifts wrapped, say), and assume for simplicity that the service time per customer is some constant **T**.

The service is first-come first-serve.

Given as input a group of customers along with an arrival time for each customer, the kinds of queries that your program should be able to answer are:

- How long will a given Person wait?
- How many customers in that group got served on that typical day?
- What is the total idle time of the employee (the person behind the service counter)?
- How long is the longest break that the employee has?
- What is the longest that the waiting line got, measured in the number of people waiting, not including the person being served? (Note that if a new person arrives right at the exact time when the person at the front of the line is called for service, both the new person and the person being called for service are included.)

Definition: A *break* is the period of time between the end of serving one person and the beginning of serving the next person. If the next person is right there, waiting in the queue, the break length is 0. Note that if the last person in the group is done before 5pm, then the employee will have a last break extending from the time the last person is done until 5pm. Over the span of the day, the employee may end up having several breaks, or no break at all. The employee does not take any break while serving a customer of if there is a person waiting to be served.

Definition: The *total idle time* of the employee is the sum of the lengths of the breaks that the employee ends up having.

The precise formatting of the queries and answers for your program is detailed below.

The Input:

Your program will take two input files from the command prompt.

The first file, called **customersfile.txt**, has the customers' information, such that each customer has one "paragraph" of two lines, with at least one blank line between one paragraph and the next. The format of the paragraph of a customer is:

ID-NUMBER: a unique integer customer id
ARRIVAL-TIME: hh:mm:ss

For example, for a person arriving 15 minutes and 27 seconds after 9am, the arrival time is 9:15:27; for a person arriving 17 minutes 35 seconds after 10am, the arrival time is 10:17:35; and for a person arriving 3:30sharp, the arrival time is 3:30:0.

The first line of **customersfile.txt** is an unsigned positive integer, representing **the constant service time per customer**, in seconds. After that line there is at least one blank line. After that, the 2-line paragraphs follow.

Note that some customers may arrive before 9am or after 5pm. Those who are not served before 5pm are dismissed without service. If the last person served begins to get service before 5pm and the service time will take the clock beyond 5pm, s/he is served completely before the employee goes home. The waiting time of a person that arrives after 5pm is 0. The waiting time of a person that arrives before 5pm but does not get served is the time from his/her arrival until 5pm.

You are guaranteed that the arrival times of the customers in **customersfile.txt are** in chronological order. That is, the arrival time of a customer A is greater than the arrival times of all the customers that occur before A in the input file. Also, you are guaranteed that all customers ids are positive, and no customer id occurs multiple times in the input file.

The second input file, called **queriesfile.txt**, will have a sequence of queries, one query per line, where the queries can be:

WAITING-TIME-OF customer-id // measured in seconds
NUMBER-OF-CUSTOMERS-SERVED
LONGEST-BREAK-LENGTH // measured in seconds **TOTAL-IDLE-TIME** // measured in seconds
MAXIMUM-NUMBER-OF-PEOPLE-IN-QUEUE-AT-ANY-TIME

Note that there could be many queries beginning with **WAITING-TIME-OF** but ending with different id numbers.

The output:

Your program must take as input the two input files above, in that order. That is, we should be able to compile your program into an executable file (called Program2, say), then call it from the command-line like this:

>> Program2 **customersfile.txt queriesfile.txt**

(Research how to accept text files as arguments in Eclipse)

Your program must then output as many lines as there are query lines: the k^{th} output line should be the answer to the k^{th} query line in file **queriesfile.txt**. Your output line answer to a query should begin by repeating the exact same query, followed by a colon (:), followed by the query-answer.