CMPS 12B-02, Fall 2017
HW4: SlugFest Bank Teller Lines

Due: Tuesday, Nov 21, 2017 by 11:59pm

- All assignments must be submitted through git. Please look at the Piazza guide on submitting assignments.
- Please follow instructions properly, include naming conventions, and carefully read through the input/output formats. There's no spec file associated with HW4, so you won't have to run the checking script. However, you're still responsible for submitting all the files that are required by this homework.
- If your HW4 solution is submitted by Tuesday, November 21, 2017 by 11:59pm, then it may get full credit. One point will be deducted for late HW4 solutions that are submitted by Wednesday, November 22 by 11:59pm. Solutions submitted after that will not be graded. Don't forget to fill in your commit id in the Google Form. [Note that Thursday, November 23 and Friday, November 24 are Thanksgiving holidays in the U.S., with no classes, Lab Sections or Office hours.]
- Clearly acknowledge sources in a README file, and indicate if you discussed the problems with other students or groups. Please submit a README file even if you didn't use any sources. In all cases, the course policy on collaboration applies, and you should refrain from getting direct answers from anybody or any source. If in doubt, please ask the Instructor or the TAs.

1 Problem description

Main objective: Simulate 5 Bank Teller lines, similar to the single Bank Teller line that we discussed in class, which is also described in Chapter 8 of the P&C textbook. Customers come into the SlugFest Bank, and they wait on one of the Teller lines. Each input line contains the name of the customer (with no spaces in that name), the number of the Teller line that the customer waits on (from 1 to 5), the arrival time for the customer, and the number of minutes that the customer's banking transaction takes. Arrival time for a customer is just the number of minutes since the simulation started that the customer arrived. For example, the input line:

```
4 Jefferson 25 8
```

means that customer Jefferson arrives 25 minutes after the simulation started, waits on Teller 4's line, and has a banking transaction that takes 8 minutes.

But that doesn't mean that Jefferson gets to begin her transaction at time 25.

There could be other customers who arrived before her who are on line, or who are being served by Teller 4. You'll have to maintain a Queue (subtle hint) of the customers on each Teller's line. Whenever the current customer served by a Teller line finishes her transaction, the next customer on that Teller's line immediately begins to be served by that Teller. So if Teller 4 begins Jefferson's transaction at time 47 and completes Jefferson's transaction at time 55, and the next customer on Teller 4's line (whom we'll call Madison) requires 4 minutes for his transaction, then Madison will complete his transaction at time 59.

The Simulation begins at time 0. Sometimes there may not be any customers on a Teller's line, so the Teller is idle. The arrival times in the input file are always increasing (or stay the same). That is, you'll never find an input line describing a customer arriving at time 72 that is followed by an input line for a customer arriving at an earlier time, such as 71. Two customers might arrive at the same time, but only if they line up for different Tellers. You can assume that there never will be two customers who arrive at the same time and line up for the same Teller.

Output: Your output file should contain a line giving the time at which each customer begins her transaction at a Teller, and another line giving the time at which each customer ends her transaction with a Teller. Note that the begin/end information for all 5 Tellers should appear together in the output. For example, the output might be:

```
2 Washington begins 40
4 Jefferson begins 47
5 Adams begins 51
4 Jefferson ends  55
4 Madison begins 59
2 Washington ends 62
5 Adams ends 62
2 Lincoln begins 64
4 Madison ends 68
2 Lincoln ends 70
```

Please follow this format precisely. Note that the simulation time (the last number on each output line) always increases (or stays the same); your output should also follow this rule. It's possible that there will be two (or more) customers who finish their banking transactions at the same time (like Adams and Washington, who both end at time 62). If that occurs, then the smaller Teller number should appear first in the output, which is why "2 Washington ends 62" appears before "5 Adams ends 62".

You must implement your solution to HW4 in Java (not C) using reference-based queues. Furthermore, you have to write your own queue implementation from scratch. You cannot use any built-in libraries for queues or linked lists, nor can you use an ADT for lists or an array-based queue. Please do not submit a question on Piazza asking how many queues you need for HW4; such questions will be deleted. Figuring that out is part of the HW, and moreover, it should be obvious.

• ***Also: Your Reference-Based Queue implementation should have a single external reference, lastNode, not two external references (lastNode and firstNode).***

Format: You should provide a Makefile. Running `make` should create "SlugFest.jar". Your program should take two command line arguments, an input file and an output file.

Input The input file consists of a series of input lines as described above, each (including the last line) ending with an end-of-line.

Output: On running (say) :

```
java -jar SlugFest.jar my-input.txt my-output.txt
```

the file `my-input.txt` should be read in, and the file `my-output.txt` should be output. The output file should have contents and format as described above.

Examples: Piazza now has a zip file for HW4 called `HW4files.zip`. The files `test-input.txt` and `test-output.txt` should help you see if your program is correct. An additional file, `more-input.txt` that is also in the zip file will also help you test your program … but you're not given the corresponding output file `more-output.txt`; you'll have to figure out what that file should look like yourselves. Please do not provide `more-output.txt` to other students, either on Piazza or directly; for HW4, that's unacceptable sharing.

Clarifications about output format: The following clarifications about output format have also been posted as a Piazza announcement for HW4.

- Customers may arrive at the bank starting at time 0, but there are no negative times.
- Customers may arrive at the bank at the same time, possibly even joining the same Teller line. If that happens, then the Customer whose line appears first in the input goes on the Teller line first.
- Please use the format shown below precisely for your output lines, with no space at the beginning of the line, and single spaces separately tokens/words from each other.

```
2 Washington begins 40
4 Jefferson begins 47
5 Adams begins 51
4 Jefferson ends  55
4 Madison begins 59
2 Washington ends 62
5 Adams ends 62
2 Lincoln begins 64
4 Madison ends 68
2 Lincoln ends 70
```

- In your output, earlier times must always appear before later times, as shown in the example above.

  But multiple things may happen at the same time. For example, Customer X might begin at time 40 on line 3, Customer Y might begin at time 40 on line 2, and another Z might end at time 40 on line 2. Events occurring <u>at the same time</u> for line 2 should <u>always</u> appear before events for line 3, and similarly for other lines numbers.

  And when two events on the same line occur at the same time, the ending event should obviously appear before the beginning event. That is, Customer Z ends her transaction on line 2 before Customer Y begins his transaction on line 2.

  So for the X, Y and Z events occurring at time 40, the output should be:

  2 Z ends 40
  2 Y begins 40
  3 X begins 40

2 Grading

Your code should terminate within 3 minutes for all runs. If it doesn't, we will not give you credit. No spec file for checker is provided for HW4, but you still should be sure to submit (via your GitLab repository) your java files (not your jar file), your `Makefile` (which will create the Slugfest.jar file, as described under **Format** above), `README`, `test-input.txt`, `test-output.txt`, `more-input.txt` , as well as the `more-output.txt` that you generated when you ran your program on `more-input.txt`. To receive credit, you must also submit your git commit id using the Google Form for HW4.

1. (10 points) Full solution as described above.
2. You will lose one point for each error that you have when we test your program on `test-input.txt` and on `more-input.txt`
3. You may receive 0 points if you do not submit the right files (as described in the first paragraph of this section), or if you do not submit your git commit id properly.