Homework 2

Due: 4pm on Nov 10, 2016

Objectives:

- Understand how Queue/Deque can be used as an aid in an algorithm.
- Explore and compare different data structures (ArrayDeque and LinkedList) and their methods to solve the same problem.

Josephus Problem (from Wikipedia)
There are people standing in a circle waiting to be executed. A certain number of people are skipped and then one person is executed. This is repeated starting with the person after the person who was executed. The elimination proceeds around the circle (which is becoming smaller as the executed people are removed) until one person remains.

Named after Flavius Josephus, a Jewish historian living in the 1st century. According to Josephus' account of the siege of Yodfat, he and his comrade soldiers were trapped in a cave, the exit of which was blocked by Romans. They chose suicide over capture and decided that they would form a circle and start killing themselves using a step of three. Josephus and another man remained alive last and gave up to the Romans.

The goal is to determine where to stand in the circle so that you are the last person alive.

Imagine that you are in the same situation just like Flavius Josephus. You are in danger!

However, you are a programmer and smart enough to build an algorithm that tells you where to stand in the circle to survive.

As you can imagine, time is critical here. You need to quickly implement and compare three different methods in the `Josephus` class and decide which method is the fastest to use for you to survive.

You are to implement three different methods to solve the Josephus problem.

Step-by-step guide:

- Understand how Josephus problem works.
  - For your understanding, the steps of execution are as follows using Queue data structure (size = 6, rotation = 3).

    Initial Circle: 1 2 3 4 5 6
    After the 1st execution: 4 5 6 1 2
    After the 2nd execution: 1 2 4 5
    After the 3rd execution: 5 1 2
    After the 4th execution: 5 1
    After the 5th execution: 1 (1 survives)

- Read carefully the JavaDoc comments of each method in the `Josephus` class provided.
  - Think carefully about which methods of the two Java classes ought to be used in your algorithm.
  - ArrayDeque (http://docs.oracle.com/javase/8/docs/api/java/util/ArrayDeque.html)
  - LinkedList (http://docs.oracle.com/javase/8/docs/api/java/util/LinkedList.html)
- Write your code on paper first.

  - For this homework, you need to write `Josephus` class.
  - You need to submit the paper in class.
  - Give enough space per line as you write.
  - Try to finish within 30 minutes. We will NOT deduct any points because of mistakes made at this point.
- Implement the methods on Eclipse or any other IDEs you prefer.
  - You also need to submit the source code file `Josephus.java` using Autolab.
- Test your code extensively!
  - Test code in `HW2Driver.java` file is just a starting point to test your program. When you run the program, you should see the following result. Do not make any assumptions and make sure to print the same values as you see below.

size (number of people)	rotation	survivor
1	5	1
5	1	5
5	2	3
11	1	11
40	7	24
55	1	55
66	100	7
1000	1000	609
-1	2	RuntimeException
5	0	RuntimeException

  - Notice that rotation value can be bigger than size that is the number of the people in the circle.
  - Remember! Time is critical here! You cannot afford to be slow to find out where to stand. So, in EACH iteration, your algorithm should be able to find the elimination position without rotating the circle more than its current size.
  - Don't be confused! Position here does not mean index value. It means the position in the circle to survive.
- Update your paper code with comments.
  - Focus on the areas that were not done correctly or inefficiently. (Use a different colored pen.)

Deliverables:

- A few sheets of paper that have your initial code as well as your comments (Submit this in class that is on the due date.)
  - In your comments, clearly write WHICH method(s) you would use out of three methods, assuming that there are many people in the circle, to find the survivor's position and WHY.
  - Take picture(s) of your paperwork and save them as low-resolution images and submit them on Blackboard before the due.
  - Also, submit your actual paperwork in class that is on the due date.
- Your source code file. (Make sure to include your Andrew ID and NAME in the header comments as author tag. And, make sure to remove package.) Submit your source code file `Josephus.java` using Autolab (https://autolab.andrew.cmu.edu). Make sure to submit your source code file (.java file), not .class file.

Grading:

Autolab will grade your assignment as follows.

- Working code: 70 points

- Coding conventions: 10 points
    - We'll deduct one point for each coding convention issue detected.

In case you do not pass test case(s), please spend some time to think about edge cases you may have missed and test thoroughly before asking questions to the TAs and resubmitting.

Autolab will check your coding conventions quite strictly. Make sure to read the Java Coding Conventions document when in doubt.

Autolab will show you the results of its grading within approximately a few minutes of your submission. You may submit multiple times so as to correct any problems with your assignment. Autolab uses the last submission as your grade.

The TAs will read and grade your paper that is worth the remaining 20 points.

Also, the TAs may take a look at your source code to check correctness and design. The most important criterion is always correctness. Buggy code is useless (even if you may think a found bug is very minor). It is important that your code be readable and well organized. This includes proper use of the module system and clear comments. Points will be deducted for poor design decisions, unreadable code. Your comments on your paper should also demonstrate your proper understanding of the code.

As mentioned in the syllabus, we will be using the Moss system to detect software plagiarism. Make sure to read the cheating policy and penalty in the syllabus. *Any cheating incident will be considered very seriously. The University also places a record of the incident in the student's permanent record.*

Late submissions will NOT be accepted and, if you have multiple versions of your code file, make sure you do submit the correct version.