Programming Project 5 - Due: Sunday, November 20 at 23:59 PM

Your boss has heard you have learned about priority queues and has come to you for help with a problem she's been trying to address for a while. The team needs a priority queue which will give the min priority item (if the min item needs to be processed next) or the max priority item (if the max item needs to be processed next). You explain to her that priority queues are either min or max, not both, but she is certain you will come up with something and leaves you to it.

Part 1:

You have an idea! You will use a BST and will disguise it as a min/max priority queue. You will tell nobody how you did it and you will become an instant celebrity in the office!

Hints:

- A lot of the code was given to you in the lecture slides. This should be very straight forward to implement once you understand that code.

Part 2:

You anticipate that your teammates will ask you about the performance of your queue and you decide to do a runtime analysis beforehand. You are clear in your analysis about what it can and what it cannot do well.

Part 3:

You have some free time left and you decide to have a look at an industrial strength balanced BST implementation. You have heard that for any large size text, quite a large portion of the words are very common words which occur over and over again (e.g. “a”, “the”, ...). You plan to check for yourself by counting occurrences of all the words in a text file that occur more often than a given number of times and compare that number to the total of words in the file. You will print these words and the number of times they each occur so that you get some idea what kind of words these are.

Hints:

- Have a look at the TreeMap class documentation. Don't be intimidated by the large number of methods; you will only need to use the main ones we have seen.
https://docs.oracle.com/javase/7/docs/api/java/util/TreeMap.html
- Once you have all the *word – number of occurrence* pairs in the tree, you will iterate over the tree and print the pairs in order.

Deliverables:

You should submit a zip file named project5_first_last.zip (where first and last are your first and last name) containing ONLY the 2 files below.

Project5.java

report.txt - a text (not Word, etc.) file containing:

1) a 1 to 10 lines paragraph from you saying “I have tested this program and there are no known issues.” if you believe that to be the case, or a brief description of known issues in case your program has known problems or you could not fully implement it.

How you get points:

Part 1	50 points
Part 2	10 points
Part 3	40 points

How you lose points:

- You do not follow the given directions and decide to make changes "for fun". Specifically, **do not change the method signatures given to you**. Can use helper methods (and you should do so for Part 1 so you can eliminate duplicate code), but do not change the given method signatures.
- Your implementation is inefficient. Your solutions should be **efficient to a level seen in class for similar problems**.
- Your solution for Part 1 contains a lot of duplicate code in the methods. **Factor out into private helper method(s) any sizable/complex chunk of code that needs to be executed at more than one place.**
- You submit your whole workspace. **Submit only the files the project asks for.**
- If any of your code prints anything at all on the console (except for Part 3 and the main). **Remove all your print outs, debug statements, etc. Clean up your code and do not leave clutter behind.**
- Your code has no comments where needed. **Comment your code appropriately.**