Georgia Gwinnett College
School of Science and Technology
ITEC 3150: Advanced Programming
Homework Assignment 3

<u>Problem 1</u> [16 Points]

Using the Big O notation, estimate the time complexity of the code in

- Exercise 22.3.1 (c) & (d)
- Exercise 22.3.2 (a) – (d)
- Exercise 22.3.3 (c) & (d)

on Page 845 of the textbook. (In the 10th Edition they are Exercise 22.3 (c) & (d), Exercise 22.4 (a) – (d) and Exercise 22.5 (c) & (d) on Page 827). Briefly explain how you derive your answer.

In each of the following problems, you are asked to complete a method, the skeleton of which has been provided in a corresponding file in the attached zip folder. Provide your code between the comments "Your code starts" and "Your code ends". You can add helper methods or classes as you want, but please DO NOT modified the existing code in the file. You can write a main method to test your method. The main method will not be graded. Please upload each completed program to the homework dropbox.

<u>Problem 2</u> [50 Points]

In class we wrote a method **closestPairFast** that on an input array of numbers, finds the distance of the closest pair by first sorting the input array and then finding the closest adjacent pair (see attached ClosestPair1D.java). In this problem, you are asked to modify the method so that it returns an integer array consisting of the ***indices*** of the closest pair in the ***original*** array. If there is a tie, just return the indices of an arbitrary closest pair. The skeleton for the new method, **closestPairFast2**, is provided in the file **Hw3_p2.java**.

The following is a sample run:

Input: 8 15 19 3 12
Return value: 1 4
Explanation: The closest pair is (15, 12). The indices of 15 and 12 in the original array are 1 and 4 respectively.

The method **closestPairFast2** should have the same structure as **closestPairFast** and have time complexity $\boldsymbol{O(n \log n)}$, where n is the length of the input array.

Hint: Define a helper class consisting of two attributes: (1) the *value* of an element and (2) its *index* in the original input array. Have the class *implement the Comparable interface* so that you can sort by value and keep the original indices.

Problem 3 [34 Points]

In an ITEC 3150 section students have just taken a test. All the test scores are stored in an array. As the instructor you want to find what the highest score is on the test and how many students received that score. Write a method, **topScore**, that on an input array of integers representing students' test scores, returns an integer array consisting of the highest score and the number of times it occurs. The skeleton for the method **topScore** is provided in the file **Hw3_p3.java**.

The following is a sample run:

Input: 54 78 62 65 74 90 90 75
Return value: 90 2
Explanation: 90 is the highest score and occurs twice in the input array.

In order to receive full credit, your method must have time complexity $\boldsymbol{O(n)}$, where n is the length of the input array. Therefore, you ***do not have time to sort*** because sorting requires $O(n \log n)$ time. **Hint**: You can go through the array a couple of times.

Note: For this problem you are *NOT* allowed to use a HashMap or TreeMap. If you don't know what these are, don't worry. We will cover them in a few weeks.

Problem 4 (Extra Credits 25 points)

For the NBA All-Star Game, the fans vote for their favorite players. All the votes are stored in an ArrayList. The name of a player appears once in the ArrayList every time he receives a vote. A software expert, you are hired by the Commissioner to write an app that helps decide the starting lineups of the game. Write a method **voteCounts** that on the input ArrayList of votes, returns an ArrayList of vote getters in ***descending order*** of the number of votes received. The skeleton for the method **voteCounts** is provided in the file **Hw3_p4.java**. Also defined in the same file is the **Player** class whose attributes are player name and the number of votes received. The method **voteCounts** should return an ArrayList of Player objects in ***descending order*** of the number of votes received.

The following is a sample run:

Input: ["Kawhi", "Kevin", "Kevin", "LeBron", "Steph", "LeBron", "LeBron"]
Return: [("LeBron", 3), ("Kevin", 2), ("Steph", 1), ("Kawhi", 1)]

In the above the notation ("Lebron", 3) means a Player object whose name field is "LeBron" and count field is 3.

In order to receive full credit, your method must have time complexity $\boldsymbol{O(n \log n)}$, where n is the size of the input ArrayList. **Hint**: You can sort ArrayLists using the method **Collections.sort** whose time complexity is $O(n \log n)$**.**

You will receive *partial credit* if your method has the correct vote counts for each vote getter and returns the list in an unsorted order; you will receive more partial credit if the player receiving the most votes appears at the beginning of the list.

Note: For this problem you are *NOT* allowed to use a HashMap or TreeMap. If you don't know what these are, don't worry. We will cover them in a few weeks.