CSE 102L Computer Programming Laboratory – Exercise 11

Due Date: 23:59 Sunday June 25th

Disclosure

You will submit your file to an assignment that is given through MS teams. Your filename should be *Ex11_yourStudentNumber.java*. Submissions made after the deadline will not be accepted, be sure to submit your work before the due date and make sure to click turn in button. Your code will be automatically controlled, so be sure to have the same class, method, variable names as described here. Failure to do so may result in you receiving 0 from this exercise. **You will have a single Java file with a single class that will be Ex11_yourStudentNumber. Be careful naming your file. If your editor inserts the file into a package, remove that line from the file but do NOT delete the import statements.**

Write set of methods according to the following specifications. Declare all methods as **static** if not requested otherwise and use camelCase formatting for naming.

YOU CAN'T USE ANY SORTING/SEARCHING METHOD THAT YOU DID NOT WRITTEN.

Methods

1. numOfTriplets(arr: int[], int sum): int – Given an array of random distinct integers and a sum value, find count of triplets with their sum is smaller than given sum value. Given the array = {-2, 0, 1, 3}, and the sum = 2; output will be 2. Because sum of (-2, 0, 1) and (-2, 0, 3) are smaller than 2.

   Expected time complexity is **O(n^2).** This means your method's execution time shouldn't be longer than approximately 500ms when n = 10000.

2. kthSmallest(arr: int[], k: int): int – Given an array and a number k where k is smaller than the size of the array, find k'th smallest element in the given array and return it.

   Expected time complexity is **O(nlogn).**

3. subSequence(str: String): String – return the maximum increasingly ordered subsequence of characters. İf str = "Welcome" should return "Welo". Analyze the time complexity of your method and display it before returning the result. For example if you think your method is O(n^3) display "n^3"

4. isSubString(str1: String, str2: String): int – given two string test whether str2 is substring of str1. Return the index where str2 begins in str1. For example given str1="Welcome" and str2="come" your method should return 3. If it is not substring, return -1. Do **NOT** use indexOf method or other similar ones. Your algorithm needs to be at least **O(n+m)**

5. findRepeats(arr: int[], int n): None – Write a method that finds and displays all elements which repeat more than n times in an array. Minimum acceptable complexity is **O(nlogn)**