Georgia Gwinnett College
School of Science and Technology
ITEC 3150: Advanced Programming
Homework Assignment 5

In each of the following problems, you are to complete a method, the skeleton of which has been provided in a corresponding file. You can add helper methods or classes as you want, but please ***DO NOT*** modify the existing code in the file. You can write a main method to test your method. The main method will not be graded.

Problem 1 [50 Points]

Write a method, **twoSumSorted2**, that solves the following variant of the Two-Sum problem: Given a *sorted* array of integers where *each element is unique* and a target integer, return in an Array List, the *indices* of *all pairs* of elements that sum up to the target. Each pair of indices is also represented as an Array List (of two elements). Therefore, the method returns an Array List of an Array List of size 2. If no pair in the input array sums up to the target, then the method should return an empty list.

The following are two sample runs:

Input: $A = [-7,\ -5,\ -2,\ 0,\ 1,\ 6,\ 7,\ 8,\ 9],\ \ target = 1$
Return value: $\big[[0,7],[1,5],[3,4]\big]$
Explanation: The pairs in the input array that sum up to 1 are $[-7,8], [-5,6]$ and $[0,1]$. Their *indices* are $[0,7], [1,5]$ and $[3,4]$ respectively.

Input: $A = [-2,\ 0,\ 1,\ 6,\ 7,\ 8],\ \ target = 3$
Return value: [ ]
Explanation: No pair of elements in input array sums up to 3. The returned list is thus empty.

Your method must have time complexity $O(n)$, where n is the length of the input array.

The skeleton for the method **twoSumSorted** is provided in the file **Hw5_p1.java**.

Problem 2 [30 Points] The same as problem 1 but the array can be unsorted. The time complexity must be O(n) as well. Please use the starter file **Hw5_p2.java**.

Problem 3 [20 Points]

Write a method, **removeDuplicates**, that given a *sorted* array, (1) removes the duplicates so that each distinct element appears *exactly once in the sorted order* at beginning of the *original* array, and (2) returns the number of distinct elements in the array.

For example, on input $A = [0, 0, 1, 1, 1, 2, 2, 3, 3, 4]$, your method should:

- Return 5 as the number of distinct elements
- Have distinct elements 0, 1, 2, 3, 4 occupy the first 5 slots of the original array A. That is, after the method the array A should look like $A = [0, 1, 2, 3, 4, *, *, *, *, *]$, where each * is a “don’t care”, that is, we don’t care what element occupies that slot.

Note:

- For this problem you are ***NOT allowed*** to use a Set or a Map.
- Your method must have ***time complexity $O(n)$ and space complexity $O(1)$***, where n is the length of the input array.
- **Hint**: Use two pointers.

The skeleton for the method **removeDuplicates** is provided in the file **Hw5_p3.java**.