CS 501B – Introduction to JAVA Programming
Fall 2023 Semester
Due: 11/24/2023 Friday at 11:59 PM

Instructions:

1. You are not allowed to use any package/library unless stated otherwise.
2. A skeleton Assignment9.java file is attached as a stub of code.
3. You are required to make the necessary changes in the Assignment9.java file.
4. We will be testing your code on hidden test cases.
5. Test your code with your own test cases.
6. Please comment your name and CWID in the first two lines of the Assignment9.java file.
7. Add comments in your code about what you are doing.
8. You are required to zip only Assignment9.java file as FirstName_LastName_Assignment#.zip (Ex: Roushan_Kumar_Assignment9.zip).
9. This assignment covers topics from week 11.
10. Students are not allowed to collaborate with classmates and any other people outside. All work must be done individually. Any work having evidence of showing academic dishonesty violation is subjected to zero for the assignment.

Penalty:

1. 10 marks will be deducted for invalid format of file / assignment submission.
2. If you submit after the due date then 10 marks will be deducted for every day after the due day.
3. If you submit an assignment after two weeks from the due date then you will get zero marks.
4. You will receive zero, if code doesn't compile / run.

Questions:
Each question carries **25** points and total points is **100.**

1. Implement **`int maxSubarrayLengthWithDistinctElements(List<Integer> elements, int k)`** to find the maximum length of a subarray with exactly **`k`** distinct integers, optimized for a large list. If there are multiple subarrays with the same maximum length, the function should return the length of any one of them.

   Example 1:
   System.out.println(maxSubarrayLengthWithDistinctElements(Arrays.asList(1, 2, 1, 3, 4, 2, 3), 3)); // Output: 4 (subarray is [1, 2, 1, 3])

   Example 2:

System.out.println(maxSubarrayLengthWithDistinctElements(Arrays.asList(5, 1, 3, 5, 2, 3, 4), 2)); // Output: 2

2. Complete **`String frequencySort(String s)`** to sort characters in a string by frequency in descending order using a Map, with same-frequency characters sorted by ascending ASCII values..

   Example 1:
   System.out.println(frequencySort("programming"));   // Output: "ggmmrrainop"

   Example 2:
   System.out.println(frequencySort("occurrence"));   // Output: "ccceerrnou"

3. Implement the **`ArrayList<Integer> spiralOrder(int[][] matrix)`** method to return an ArrayList containing the elements of a 2D matrix in spiral order. Start from the top-left corner and proceed rightwards, then downwards, then leftwards, and finally upwards, repeatedly narrowing down the traversal area in each iteration until all elements are traversed.

   Example 1:
   int[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
   System.out.println(spiralOrder(matrix)); // Output: [1, 2, 3, 6, 9, 8, 7, 4, 5]

   Example 1:
   int[][] matrix = {{1, 2}, {3, 4}, {5, 6}};
   System.out.println(spiralOrder(matrix)); // Output: [1, 2, 4, 6, 5, 3]

4. Implement **`List<Long> efficientPrimeFactors(long number)`** to return a list of prime factors in ascending order, optimized for large numbers.

   Example 1:
   List<Long> factors = efficientPrimeFactors(60);
   System.out.println(factors); // Output: [2, 2, 3, 5]

   Example 2:
   List<Long> factors = efficientPrimeFactors(100);
   System.out.println(factors); // Output: [2, 2, 5, 5]