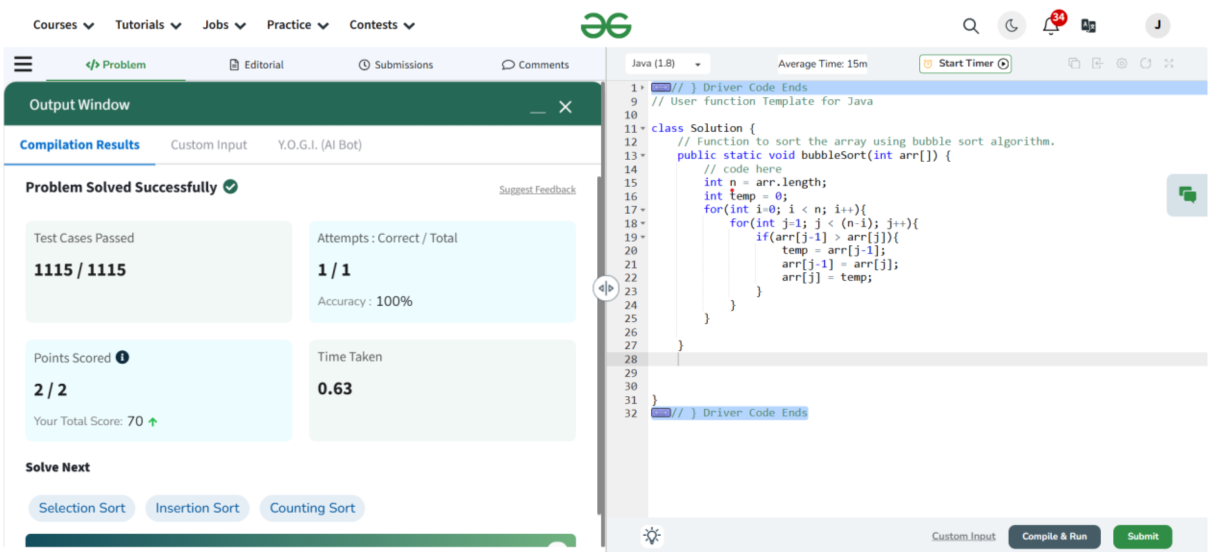
# CODING QUESTION PRACTICE – 6

## 1.Bubble Sort



Problem Solved Successfully

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
2 / 2
Your Total Score: 70

Time Taken
0.63

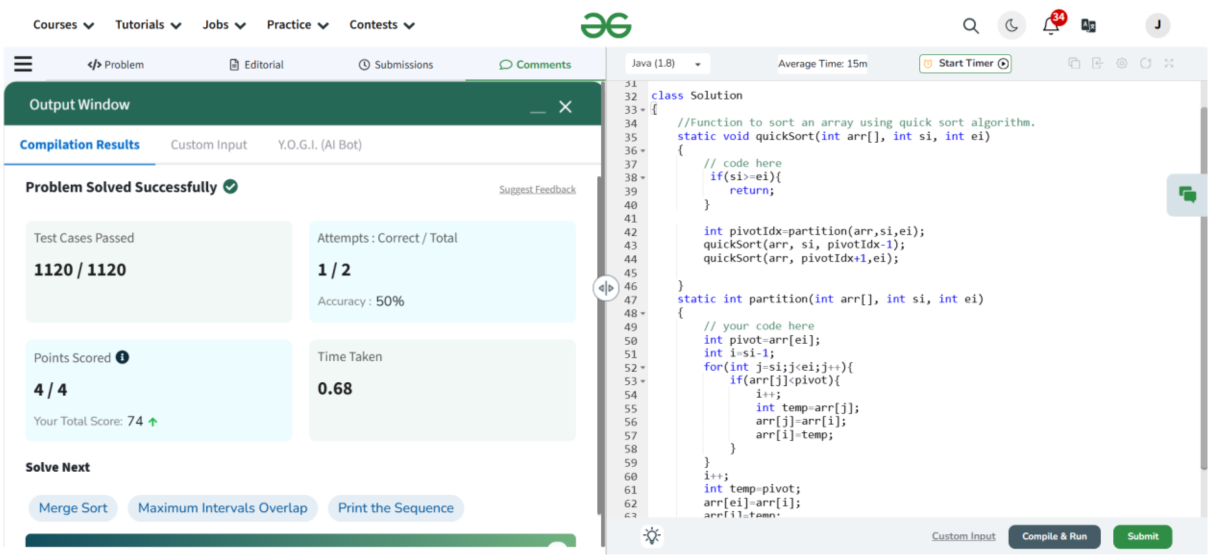
Solve Next
Selection Sort
Insertion Sort
Counting Sort

```java
// } Driver Code Ends
// User function Template for Java

class Solution {
    // Function to sort the array using bubble sort algorithm.
    public static void bubbleSort(int arr[]) {
        // code here
        int n = arr.length;
        int temp = 0;
        for(int i=0; i < n; i++){
            for(int j=1; j < (n-i); j++){
                if(arr[j-1] > arr[j]){
                    temp = arr[j-1];
                    arr[j-1] = arr[j];
                    arr[j] = temp;
                }
            }
        }

    }



}
// } Driver Code Ends
```

**Time Complexity: O(n²)**

**Space Complexity: O(1)**

## 2.Quick Sort



Problem Solved Successfully

Test Cases Passed
1120 / 1120

Attempts : Correct / Total
1 / 2
Accuracy : 50%

Points Scored
4 / 4
Your Total Score: 74

Time Taken
0.68

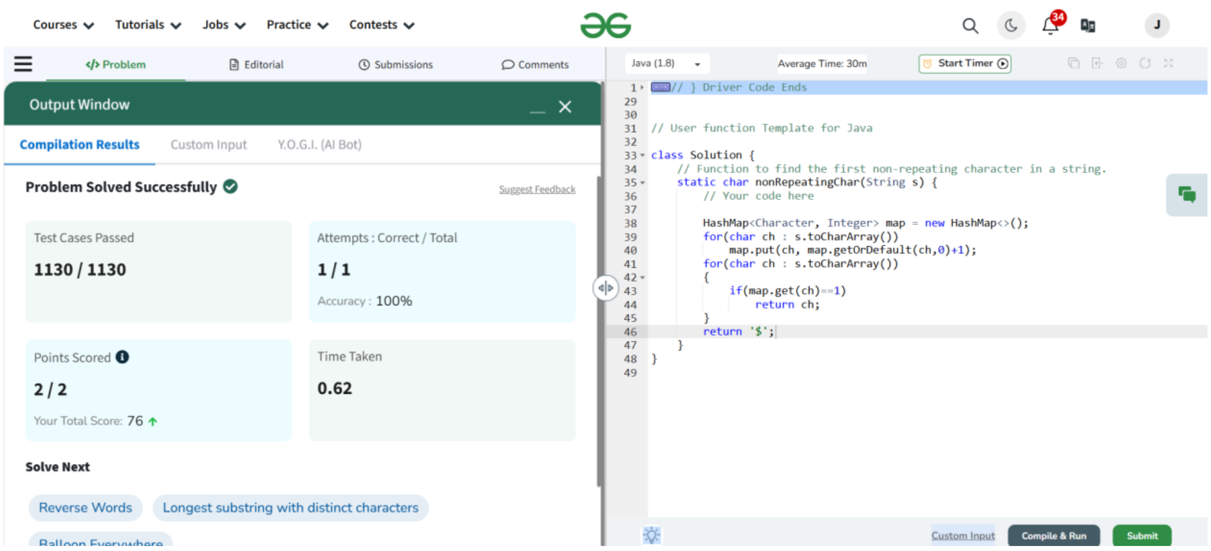
Solve Next
Merge Sort
Maximum Intervals Overlap
Print the Sequence

```java
class Solution
{
    //Function to sort an array using quick sort algorithm.
    static void quickSort(int arr[], int si, int ei)
    {
        // code here
         if(si>=ei){
            return;
        }

        int pivotIdx=partition(arr,si,ei);
        quickSort(arr, si, pivotIdx-1);
        quickSort(arr, pivotIdx+1,ei);

    }
    static int partition(int arr[], int si, int ei)
    {
        // your code here
        int pivot=arr[ei];
        int i=si-1;
        for(int j=si;j<ei;j++){
            if(arr[j]<pivot){
                i++;
                int temp=arr[j];
                arr[j]=arr[i];
                arr[i]=temp;
            }
        }
        i++;
        int temp=pivot;
        arr[ei]=arr[i];
        arr[i]=temp;
```

**Time Complexity: O(n log n) average case**

**Space Complexity: O(log n)**

## 3.Non Repeating Character



Problem Solved Successfully

Test Cases Passed
1130 / 1130

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
2 / 2
Your Total Score: 76

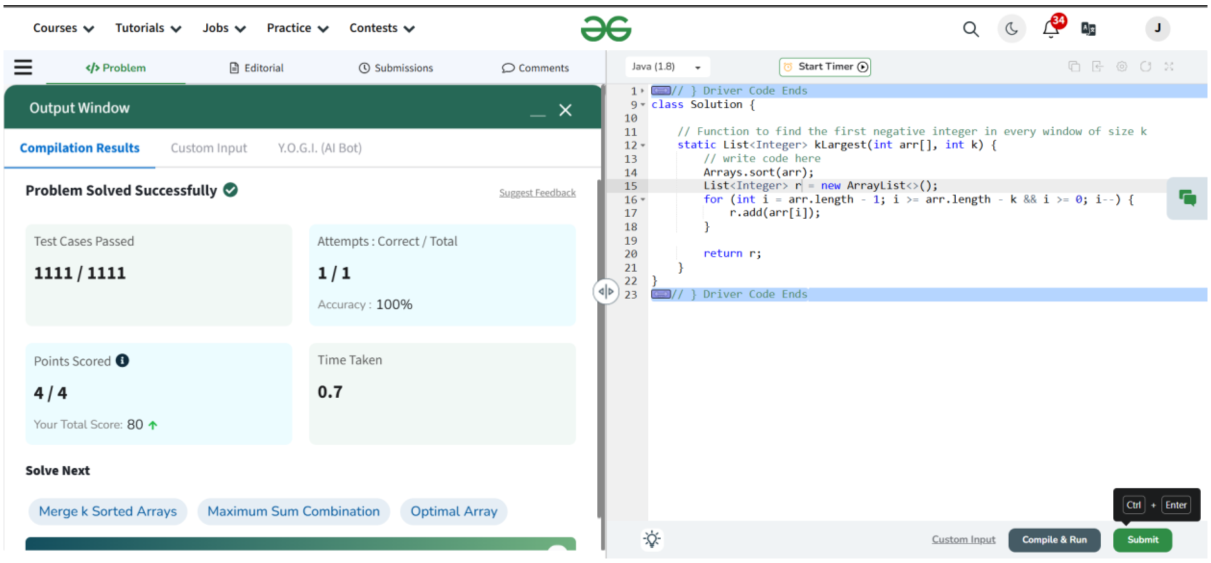
Time Taken
0.62

```java
// } Driver Code Ends


// User function Template for Java

class Solution {
    // Function to find the first non-repeating character in a string.
    static char nonRepeatingChar(String s) {
        // Your code here

        HashMap<Character, Integer> map = new HashMap<>();
        for(char ch : s.toCharArray())
            map.put(ch, map.getOrDefault(ch,0)+1);
        for(char ch : s.toCharArray())
        {
            if(map.get(ch)==1)
                return ch;
        }
        return '$';
    }
}
```

**Time Complexity: O(n)**

**Space Complexity: O(1)**

## 4.k largest elements



Problem Solved Successfully

Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 80

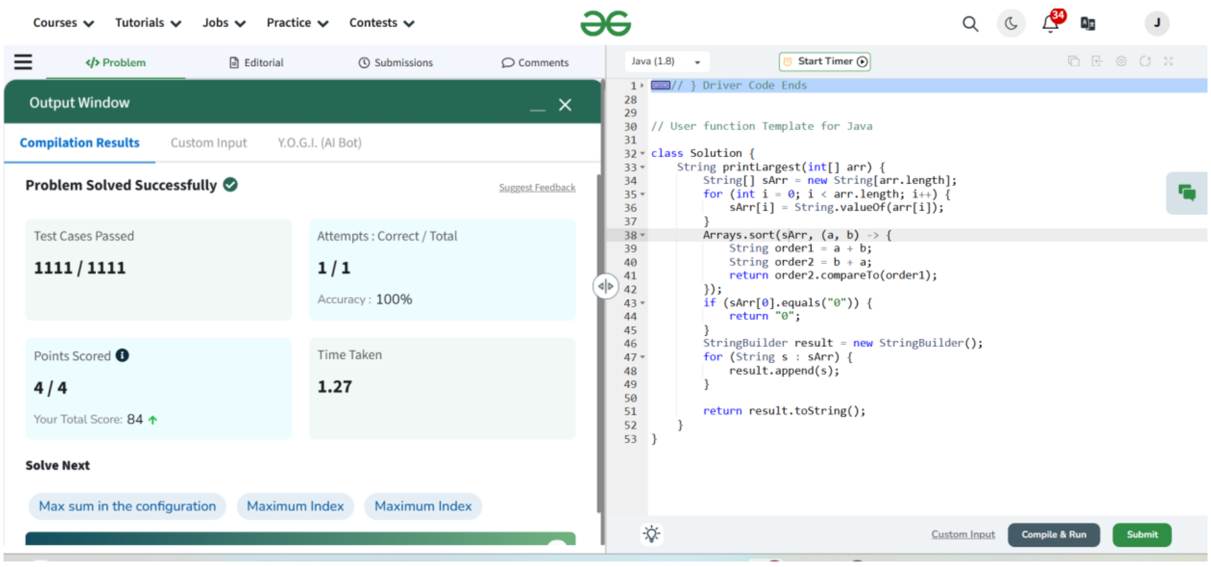
Time Taken
0.7

```java
// } Driver Code Ends
class Solution {

    // Function to find the first negative integer in every window of size k
    static List<Integer> kLargest(int arr[], int k) {
        // write code here
        Arrays.sort(arr);
        List<Integer> r = new ArrayList<>();
        for (int i = arr.length - 1; i >= arr.length - k && i >= 0; i--) {
            r.add(arr[i]);
        }

        return r;
    }
}
// } Driver Code Ends
```

**Time Complexity: O(n log n)**

**Space Complexity: O(k)**

## 5.Form the Largest Number



Output Window

Compilation Results | Custom Input | Y.O.G.I. (AI Bot)

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 84

Time Taken
1.27

Solve Next

Max sum in the configuration | Maximum Index | Maximum Index

```java
// } Driver Code Ends


// User function Template for Java

class Solution {
    String printLargest(int[] arr) {
        String[] sArr = new String[arr.length];
        for (int i = 0; i < arr.length; i++) {
            sArr[i] = String.valueOf(arr[i]);
        }
        Arrays.sort(sArr, (a, b) -> {
            String order1 = a + b;
            String order2 = b + a;
            return order2.compareTo(order1);
        });
        if (sArr[0].equals("0")) {
            return "0";
        }
        StringBuilder result = new StringBuilder();
        for (String s : sArr) {
            result.append(s);
        }

        return result.toString();
    }
}
```

**Time Complexity: O(n log n)**

**Space Complexity: O(n)**