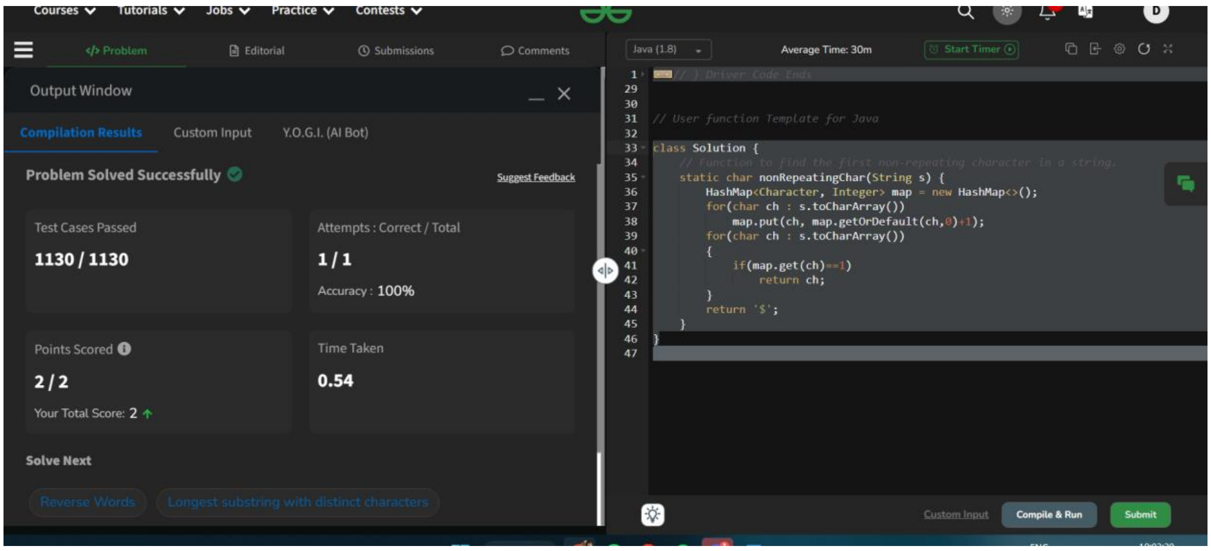
Non Repeating Character

Code:

```java
class Solution {
    // Function to find the first non-repeating character in a string.
    static char nonRepeatingChar(String s) {
        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 '$';
    }
}
```

output:



Problem Solved Successfully

Test Cases Passed
1130 / 1130

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

Points Scored
2 / 2
Your Total Score: 2

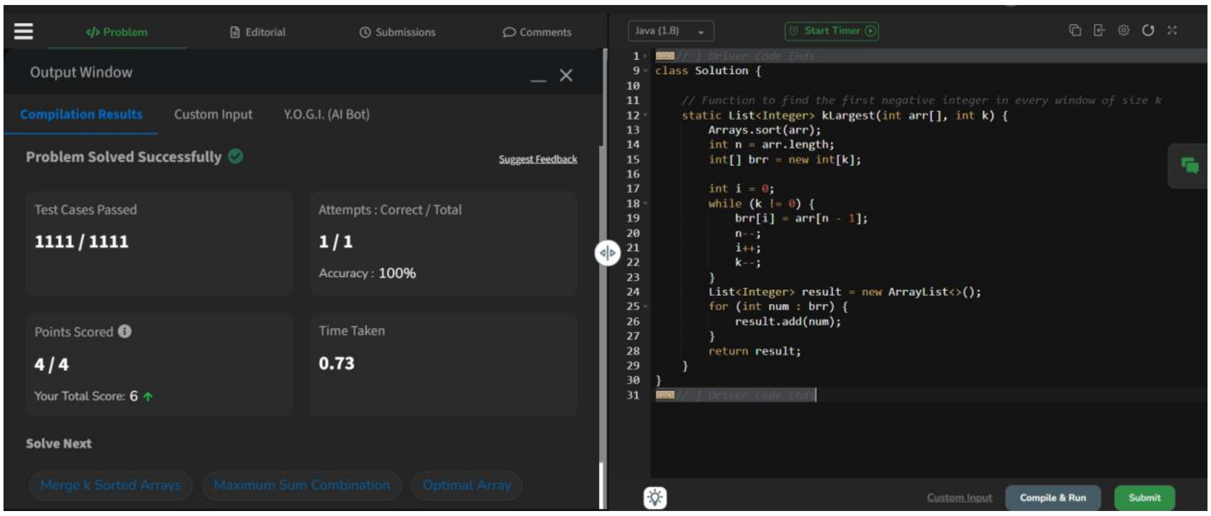
Time Taken
0.54

```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) {
        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 '$';
    }
}
```

k largest elements

output:

```
Class Solution {
  // Function to find the first negative integer in every window of size k
  static List<Integer> kLargest(int arr[], int k) {
    Arrays.sort(arr);
    int n = arr.length;
    int[] brr = new int[k];

    int i = 0;
    while (k != 0) {
      brr[i] = arr[n - 1];
      n--;
      i++;
      k--;
    }
    List<Integer> result = new ArrayList<>();
    for (int num : brr) {
      result.add(num);
    }
    return result;}
```



```
Problem Solved Successfully

Test Cases Passed          Attempts : Correct / Total
1111 / 1111                1 / 1
                           Accuracy : 100%

Points Scored              Time Taken
4 / 4                      0.73
Your Total Score: 6
```

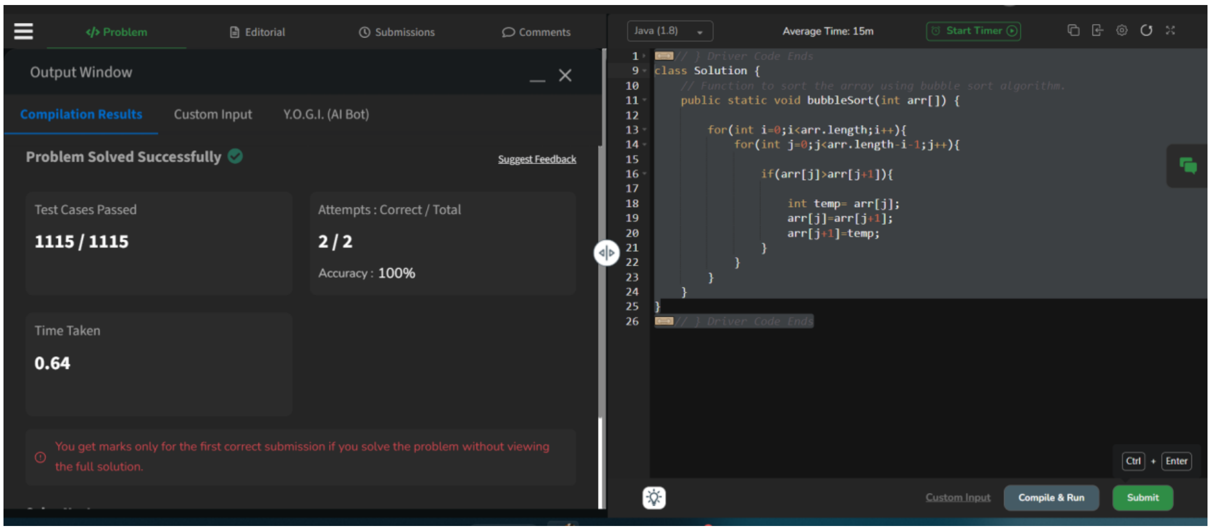
```java
// } Driver Code Ends
class Solution {

    // Function to find the first negative integer in every window of size k
    static List<Integer> kLargest(int arr[], int k) {
        Arrays.sort(arr);
        int n = arr.length;
        int[] brr = new int[k];

        int i = 0;
        while (k != 0) {
            brr[i] = arr[n - 1];
            n--;
            i++;
            k--;
        }
        List<Integer> result = new ArrayList<>();
        for (int num : brr) {
            result.add(num);
        }
        return result;
    }
}
// } Driver Code Ends
```

Bubble Sort

Code:

```
class Solution {
    // Function to sort the array using bubble sort algorithm.
    public static void bubbleSort(int arr[]) {

        for(int i=0;i<arr.length;i++){
            for(int j=0;j<arr.length-i-1;j++){

                if(arr[j]>arr[j+1]){

                    int temp= arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=temp;

                }

            }
        }
}
```

Output:



Average Time: 15m

Problem Solved Successfully

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
2 / 2
Accuracy : 100%

Time Taken
0.64

You get marks only for the first correct submission if you solve the problem without viewing the full solution.

```
// } Driver Code Ends
class Solution {
    // Function to sort the array using bubble sort algorithm.
    public static void bubbleSort(int arr[]) {

        for(int i=0;i<arr.length;i++){
            for(int j=0;j<arr.length-i-1;j++){

                if(arr[j]>arr[j+1]){

                    int temp= arr[j];
                    arr[j]=arr[j+1];
                    arr[j+1]=temp;
                }
            }
        }
    }
}
// } Driver Code Ends
```

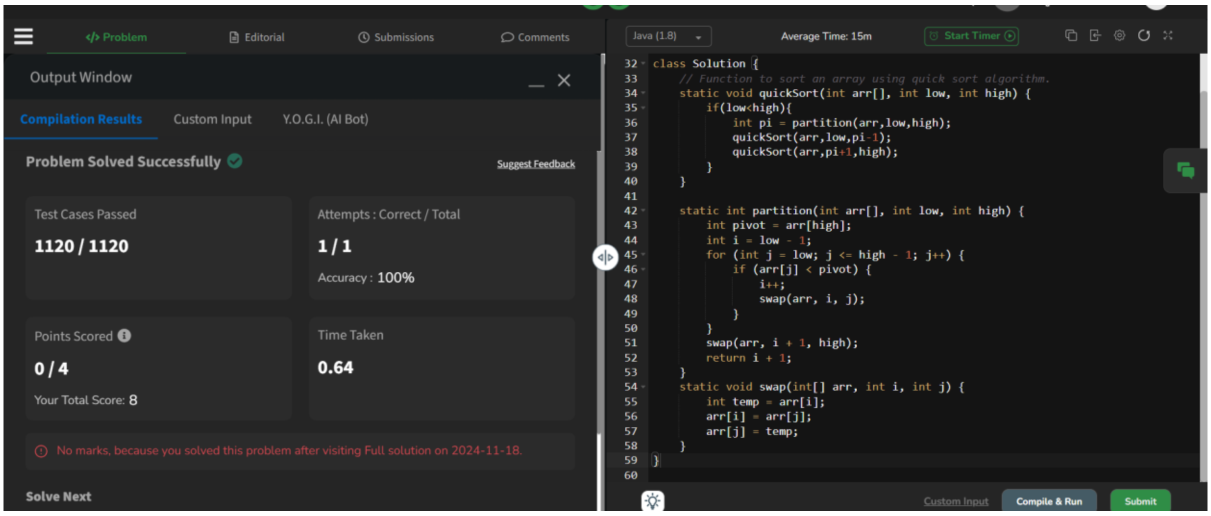
QUITE SORT

Code:

```
class Solution {
   // Function to sort an array using quick sort algorithm.
   static void quickSort(int arr[], int low, int high) {
      if(low<high){
         int pi = partition(arr,low,high);
         quickSort(arr,low,pi-1);
         quickSort(arr,pi+1,high);
      }
   }

   static int partition(int arr[], int low, int high) {
      int pivot = arr[high];
      int i = low - 1;
      for (int j = low; j <= high - 1; j++) {
         if (arr[j] < pivot) {
            i++;
            swap(arr, i, j);
         }
      }
      swap(arr, i + 1, high);
      return i + 1;
   }
   static void swap(int[] arr, int i, int j) {
      int temp = arr[i];
      arr[i] = arr[j];
      arr[j] = temp;
```

```
    }
}
```

Output:



Output Window

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

Problem Solved Successfully

Test Cases Passed
1120 / 1120

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

Points Scored
0 / 4
Your Total Score: 8

Time Taken
0.64

No marks, because you solved this problem after visiting Full solution on 2024-11-18.

Solve Next

Java (1.8) Average Time: 15m

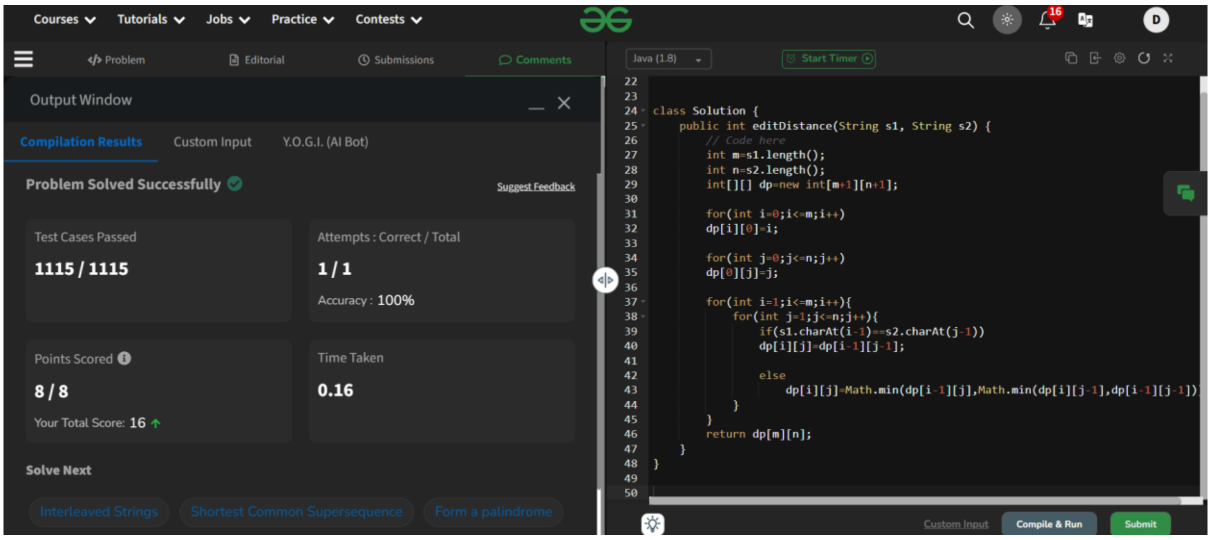
```
class Solution {
    // Function to sort an array using quick sort algorithm.
    static void quickSort(int arr[], int low, int high) {
        if(low<high){
            int pi = partition(arr,low,high);
            quickSort(arr,low,pi-1);
            quickSort(arr,pi+1,high);
        }
    }

    static int partition(int arr[], int low, int high) {
        int pivot = arr[high];
        int i = low - 1;
        for (int j = low; j <= high - 1; j++) {
            if (arr[j] < pivot) {
                i++;
                swap(arr, i, j);
            }
        }
        swap(arr, i + 1, high);
        return i + 1;
    }
    static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}
```

Edit Distance

CODE:

```
class Solution {
    public int editDistance(String s1, String s2) {
        // Code here
        int m=s1.length();
        int n=s2.length();
        int[][] dp=new int[m+1][n+1];

        for(int i=0;i<=m;i++)
        dp[i][0]=i;

        for(int j=0;j<=n;j++)
        dp[0][j]=j;

        for(int i=1;i<=m;i++){
            for(int j=1;j<=n;j++){
                if(s1.charAt(i-1)==s2.charAt(j-1))
                dp[i][j]=dp[i-1][j-1];

                else
                    dp[i][j]=Math.min(dp[i-1][j],Math.min(dp[i][j-1],dp[i-1][j-1]))+1;
            }
        }
        return dp[m][n];
    }
}
```

Output:



Output Window

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

Problem Solved Successfully   Suggest Feedback

Test Cases Passed
1115 / 1115

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

Points Scored
8 / 8
Your Total Score: 16

Time Taken
0.16

Solve Next

Interleaved Strings   Shortest Common Supersequence   Form a palindrome

```
class Solution {
    public int editDistance(String s1, String s2) {
        // Code here
        int m=s1.length();
        int n=s2.length();
        int[][] dp=new int[m+1][n+1];

        for(int i=0;i<=m;i++)
        dp[i][0]=i;

        for(int j=0;j<=n;j++)
        dp[0][j]=j;

        for(int i=1;i<=m;i++){
            for(int j=1;j<=n;j++){
                if(s1.charAt(i-1)==s2.charAt(j-1))
                dp[i][j]=dp[i-1][j-1];

                else
                    dp[i][j]=Math.min(dp[i-1][j],Math.min(dp[i][j-1],dp[i-1][j-1]))
            }
        }
        return dp[m][n];
    }
}
```