DSA Practice

Date: 18/11/2024

Problems:

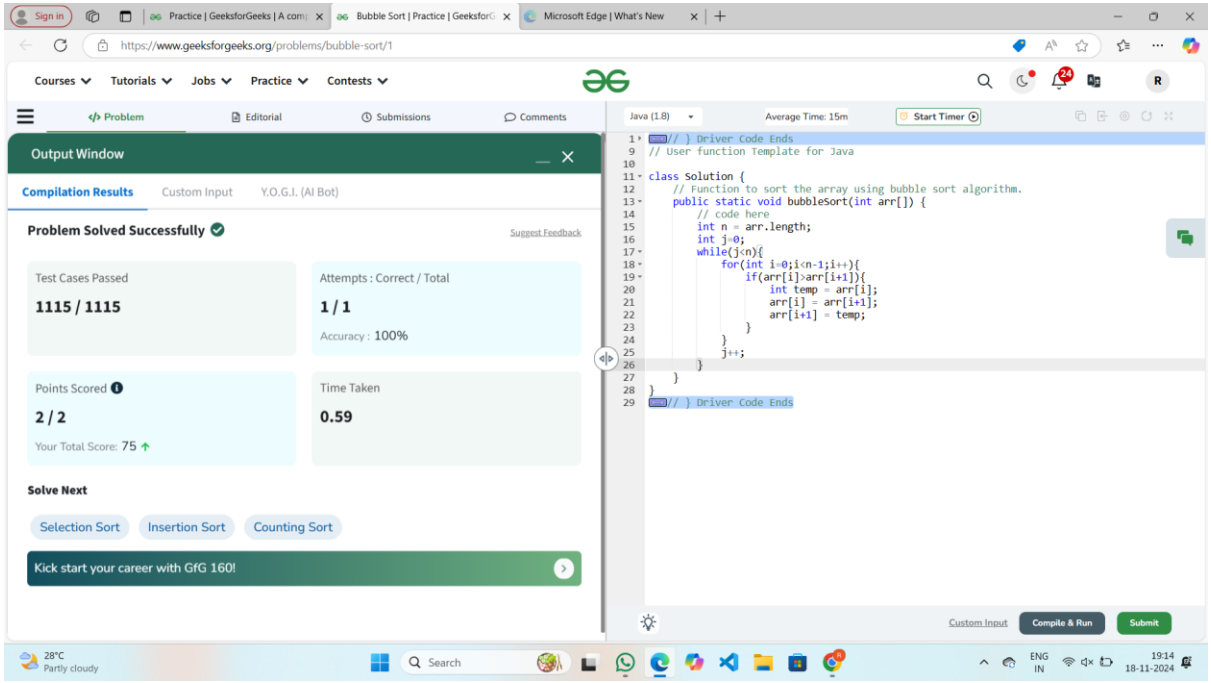
1).Bubble sort

2).Quick sort
3).Non repeating characters
4).Edit distance
5).k largest elements
6).form the largest number
Problem1:

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

}

Output:



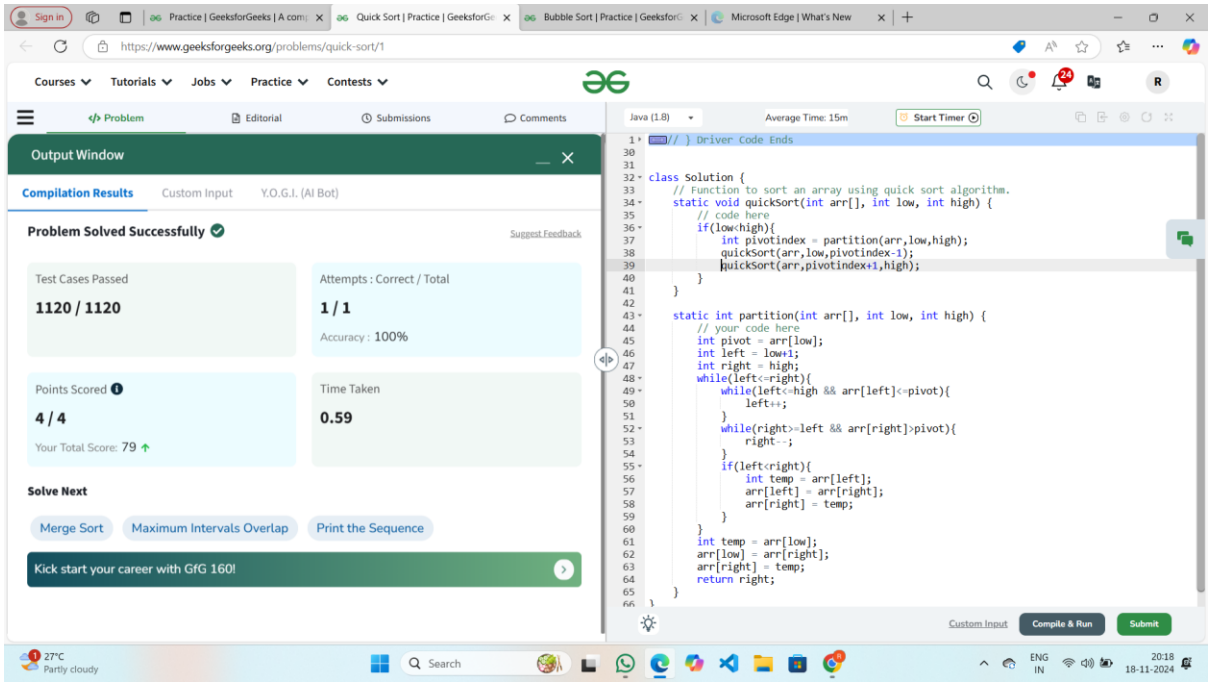
Problem2:

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

    static int partition(int arr[], int low, int high) {
        // your code here
```

```
        int pivot = arr[low];
        int left = low+1;
        int right = high;
        while(left<=right){
            while(left<=high && arr[left]<=pivot){
                left++;
            }
            while(right>=left && arr[right]>pivot){
                right--;
            }
            if(left<right){
                int temp = arr[left];
                arr[left] = arr[right];
                arr[right] = temp;
            }
        }
        int temp = arr[low];
        arr[low] = arr[right];
        arr[right] = temp;
        return right;
    }
}
```

Output:

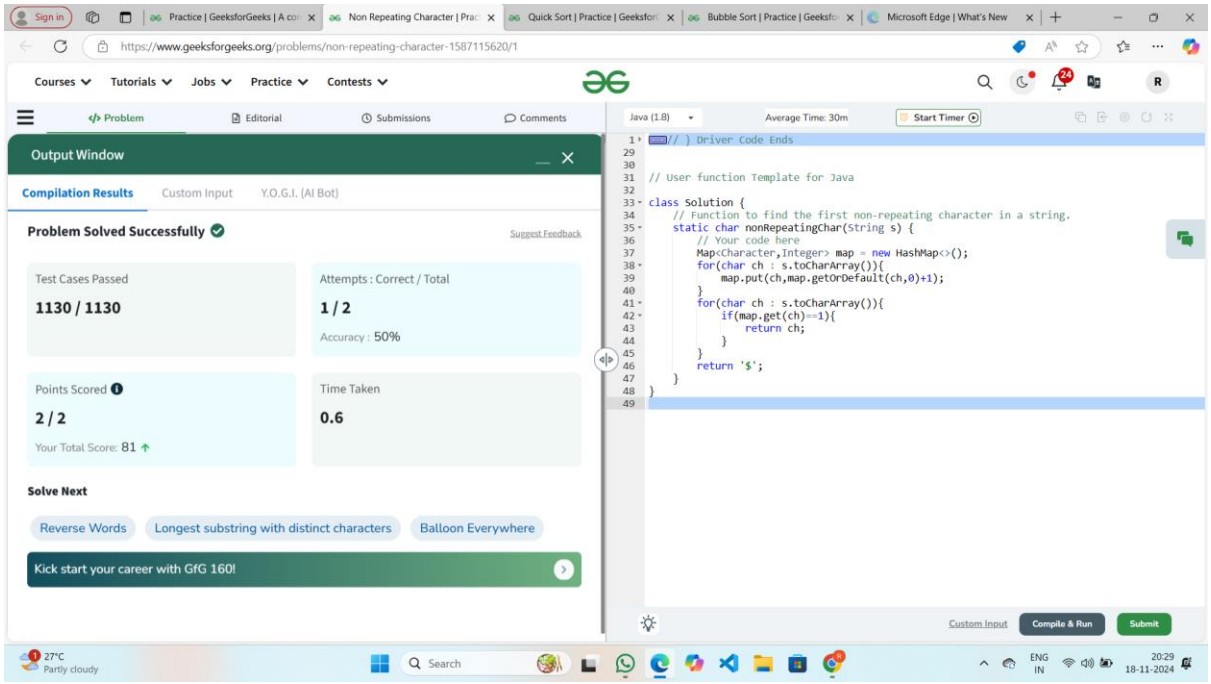


Problem3:

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

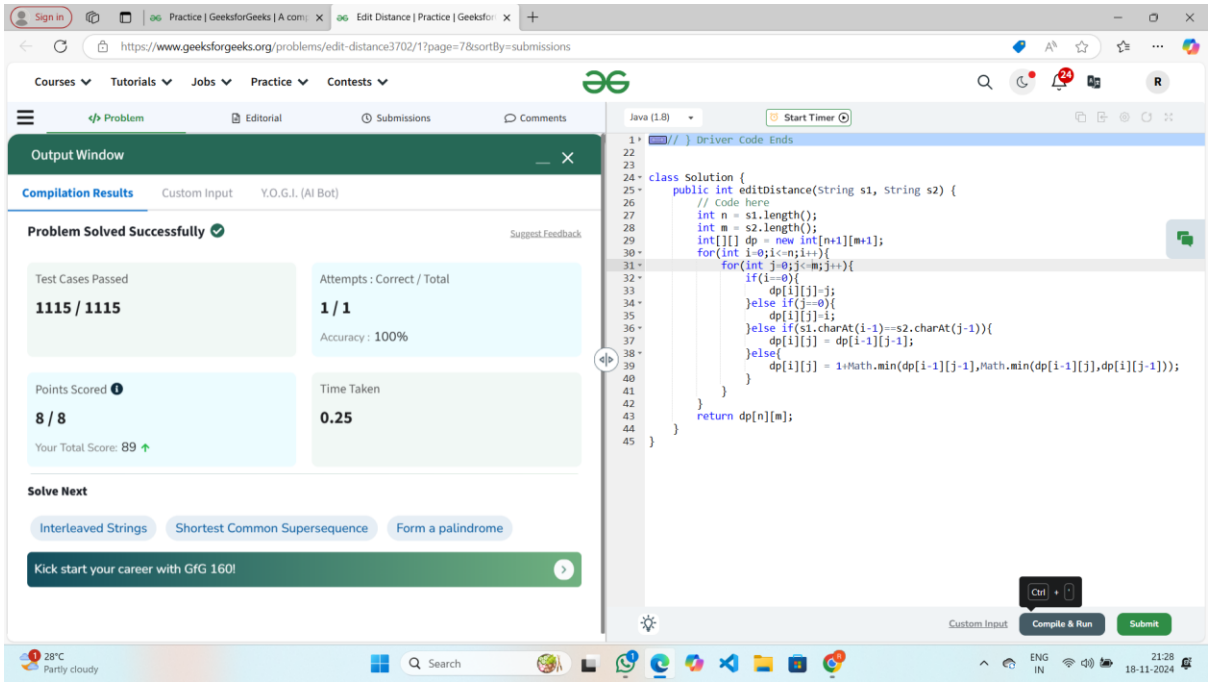


Problem4:

```
class Solution {
    public int editDistance(String s1, String s2) {
        // Code here
        int n = s1.length();
        int m = s2.length();
        int[][] dp = new int[n+1][m+1];
        for(int i=0;i<=n;i++){
            for(int j=0;j<=m;j++){
                if(i==0){
                    dp[i][j]=j;
                }else if(j==0){
                    dp[i][j]=i;
                }else if(s1.charAt(i-1)==s2.charAt(j-1)){
```

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

Output:

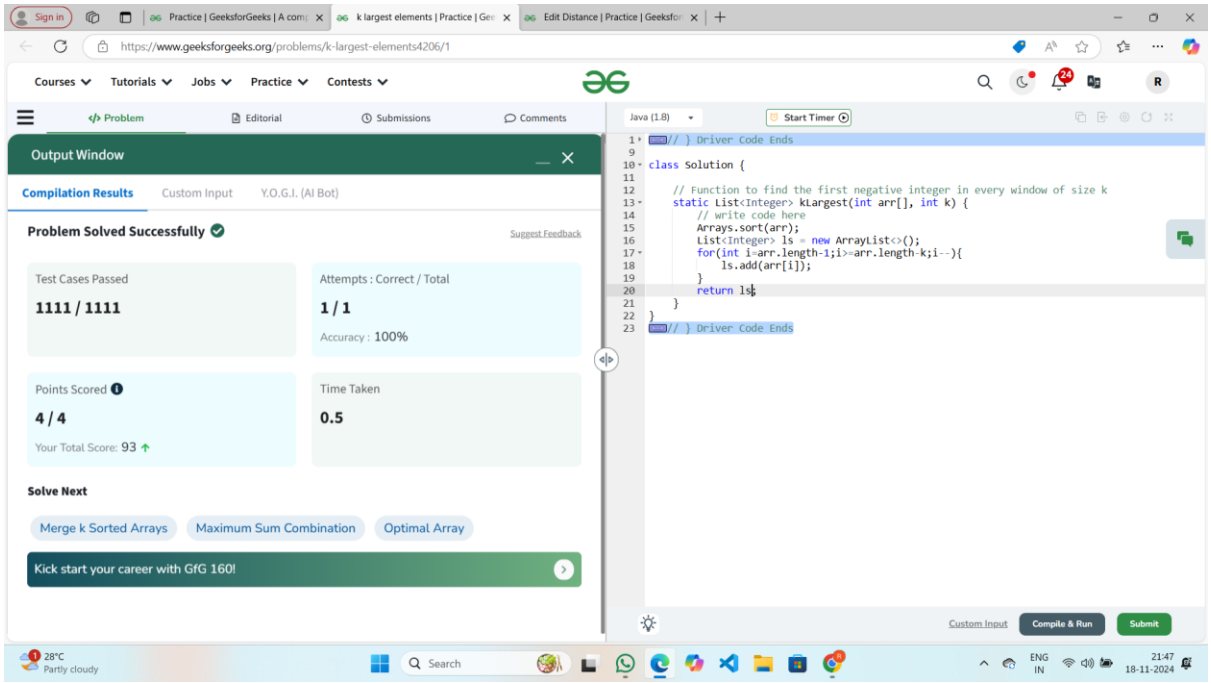


Problem5:

```
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> ls = new ArrayList<>();
        for(int i=arr.length-1;i>=arr.length-k;i--){
            ls.add(arr[i]);
        }
        return ls;
    }
}
```

Output:



Problem6:

```
class Solution {
    String printLargest(int[] arr) {
        // code here
        Integer[] arr1 = Arrays.stream(arr).boxed().toArray(Integer[]::new);
        Arrays.sort(arr1,(a,b)->{
            String temp1 = Integer.toString(a);
            String temp2 = Integer.toString(b);
```

```
        return (temp1 + temp2).compareTo(temp2 + temp1);
    });
    StringBuilder sb = new StringBuilder();
    for(int no : arr1){
        sb.append(Integer.toString(no));
    }
    return sb.toString();
  }
}
```

Output: