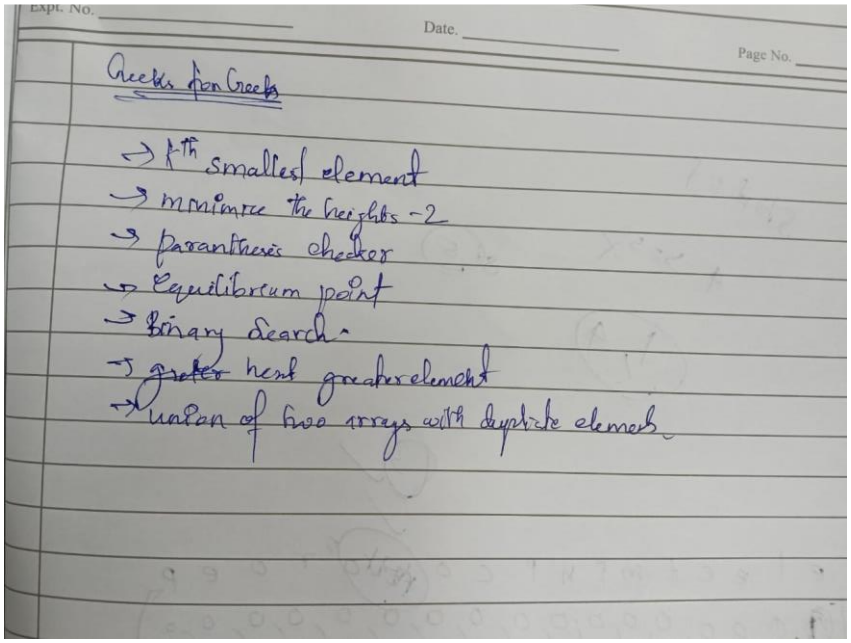
DSA Practice

Date: 13/11/2024

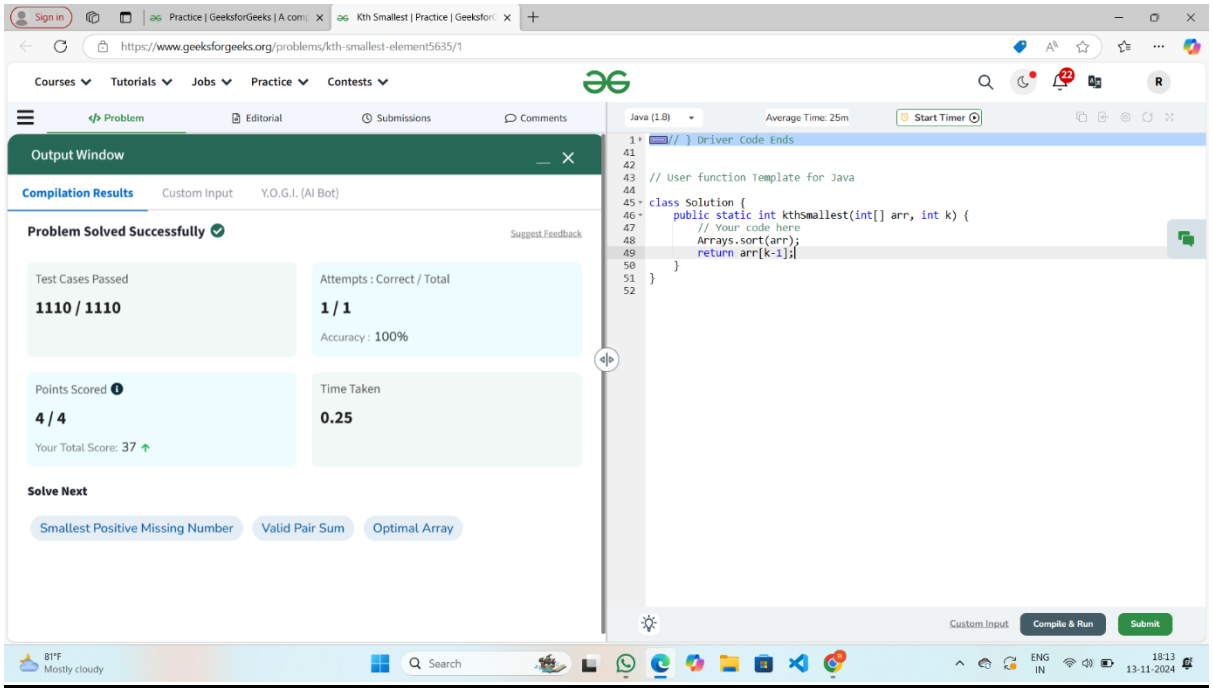
Problems:



Problem 1:

```
class Solution {

    public static int kthSmallest(int[] arr, int k) {

        // Your code here

        Arrays.sort(arr);

        return arr[k-1];

    }

}
```

Output:



Problem 2:

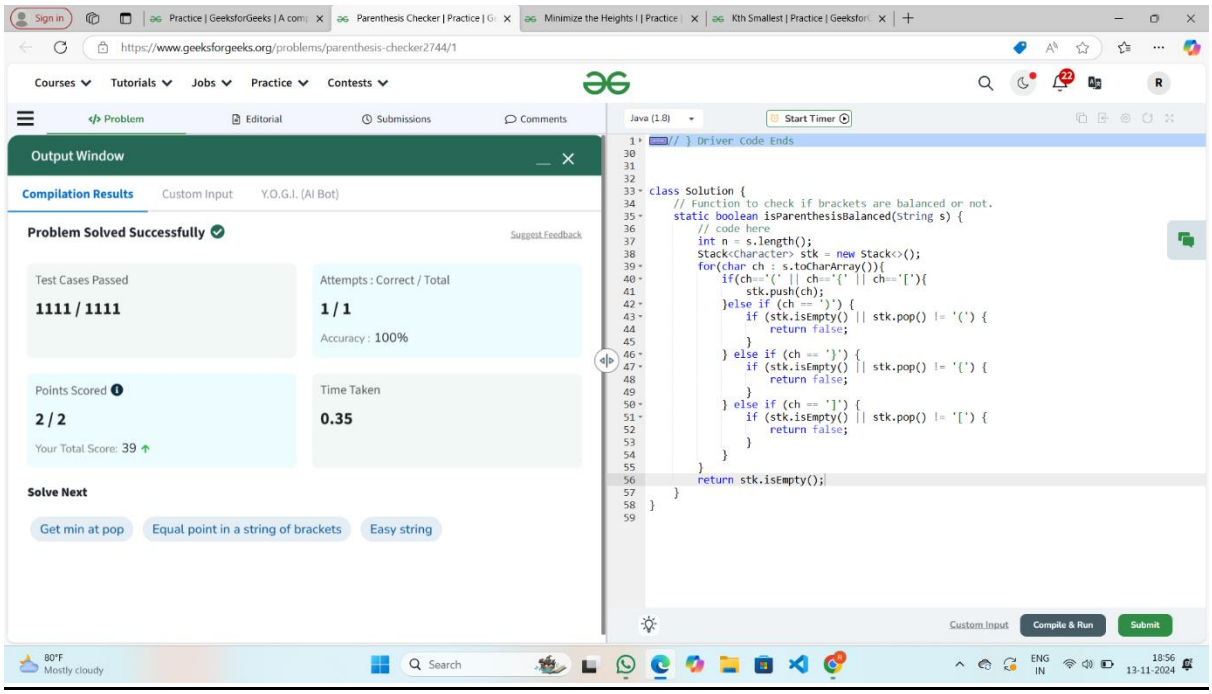
Output:

Problem 3:

```
class Solution {
    // Function to check if brackets are balanced or not.
    static boolean isParenthesisBalanced(String s) {
        // code here
        int n = s.length();
        Stack<Character> stk = new Stack<>();
        for(char ch : s.toCharArray()){
            if(ch=='(' || ch=='{' || ch=='['){
                stk.push(ch);
            }else if (ch == ')') {
                if (stk.isEmpty() || stk.pop() != '(') {
                    return false;
                }
```

```
        } else if (ch == '}') {
            if (stk.isEmpty() || stk.pop() != '{') {
                return false;
            }
        } else if (ch == ']') {
            if (stk.isEmpty() || stk.pop() != '[') {
                return false;
            }
        }
    }
    return stk.isEmpty();
  }
}
```

Output:

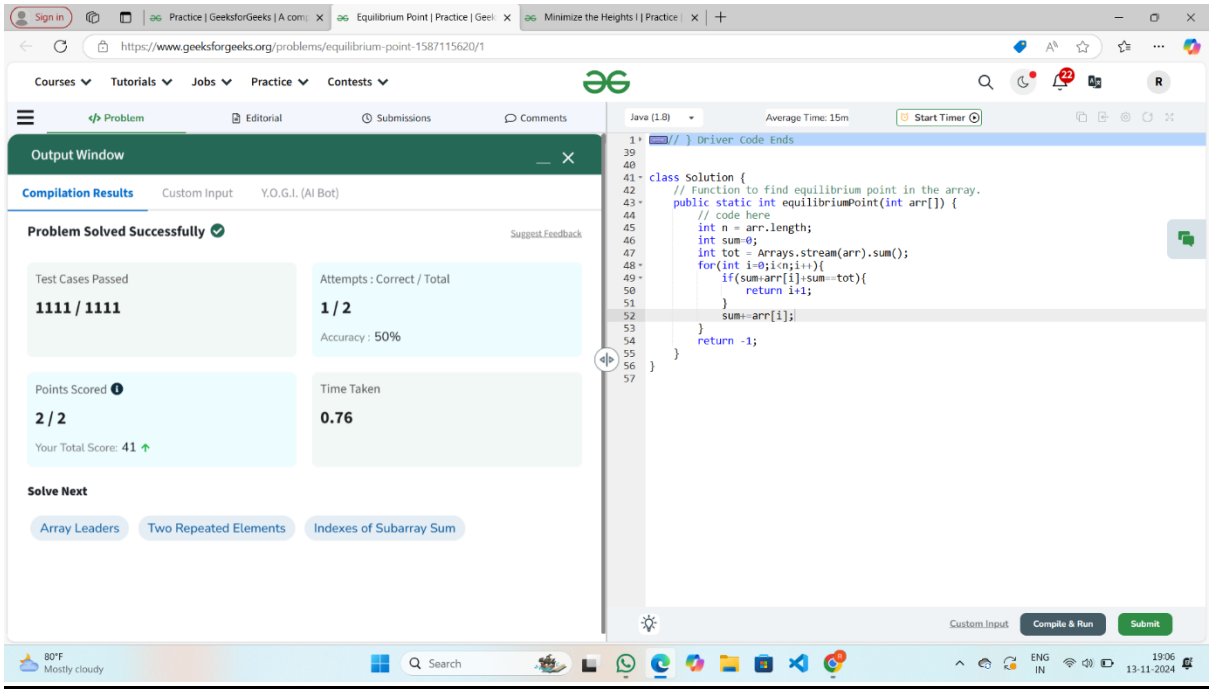


Problem 4:

```
class Solution {
```

```
    // Function to find equilibrium point in the array.
    public static int equilibriumPoint(int arr[]) {
        // code here
        int n = arr.length;
        int sum=0;
        int tot = Arrays.stream(arr).sum();
        for(int i=0;i<n;i++){
            if(sum+arr[i]+sum==tot){
                return i+1;
            }
            sum+=arr[i];
        }
        return -1;
    }
}
```

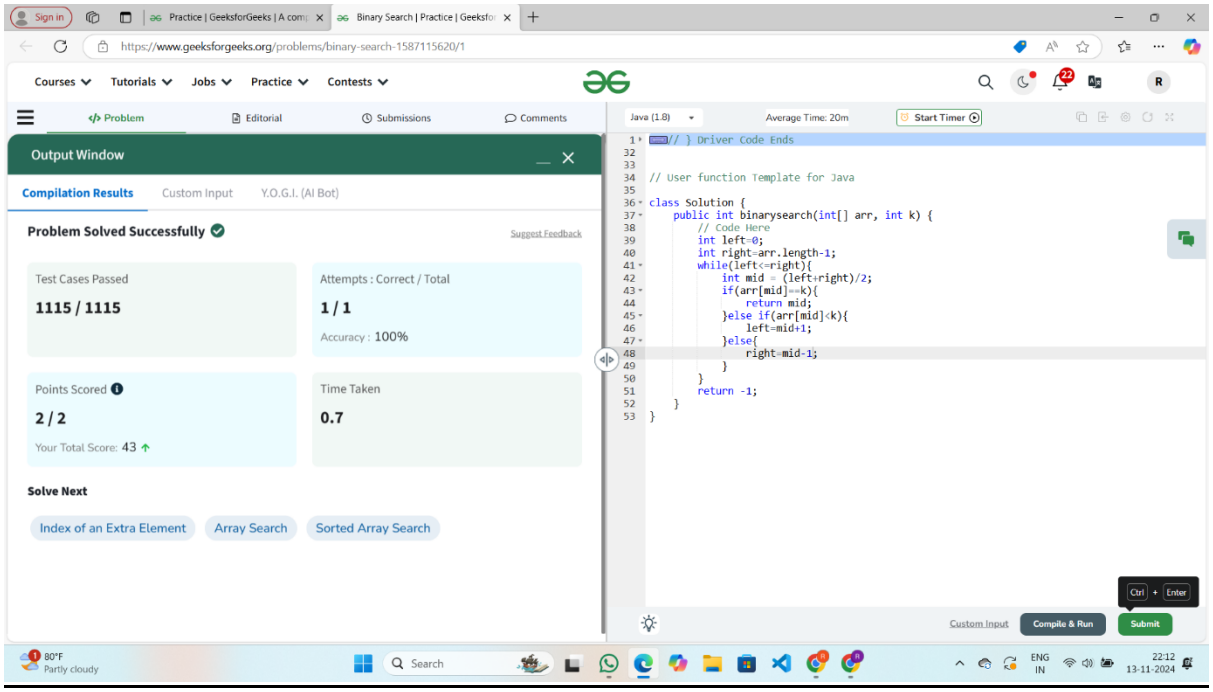
Output:



Problem 5:

```
class Solution {
    public int binarysearch(int[] arr, int k) {
        // Code Here
        int left=0;
        int right=arr.length-1;
        while(left<=right){
            int mid = (left+right)/2;
            if(arr[mid]==k){
                return mid;
            }else if(arr[mid]<k){
                left=mid+1;
            }else{
                right=mid-1;
            }
        }
        return -1;
    }
}
```

Output:

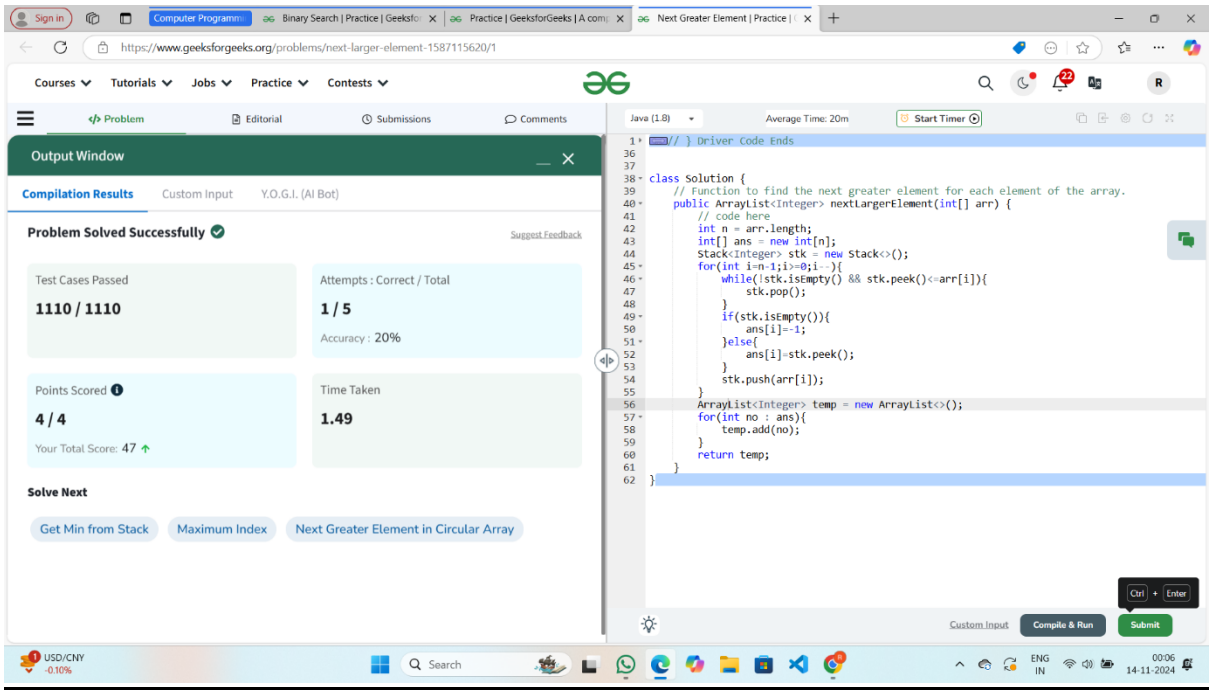


Problem 6:

```
class Solution {
    // Function to find the next greater element for each element of the array.
    public ArrayList<Integer> nextLargerElement(int[] arr) {
        // code here
        int n = arr.length;
        int[] ans = new int[n];
        Stack<Integer> stk = new Stack<>();
        for(int i=n-1;i>=0;i--){
            while(!stk.isEmpty() && stk.peek()<=arr[i]){
                stk.pop();
            }
            if(stk.isEmpty()){
                ans[i]=-1;
            }else{
                ans[i]=stk.peek();
```

```
        }
        stk.push(arr[i]);
    }
    ArrayList<Integer> temp = new ArrayList<>();
    for(int no : ans){
        temp.add(no);
    }
    return temp;
  }
}
```

Output:



Problem 7:

```
class Solution {
    public static int findUnion(int a[], int b[]) {
        // code here
        int n = a.length;
```

```
        Set<Integer> set = new HashSet<>();
        for(int i=0;i<n;i++){
            set.add(a[i]);
        }
        for(int no : b){
            set.add(no);
        }
        return set.size();
    }
}
```

Output: