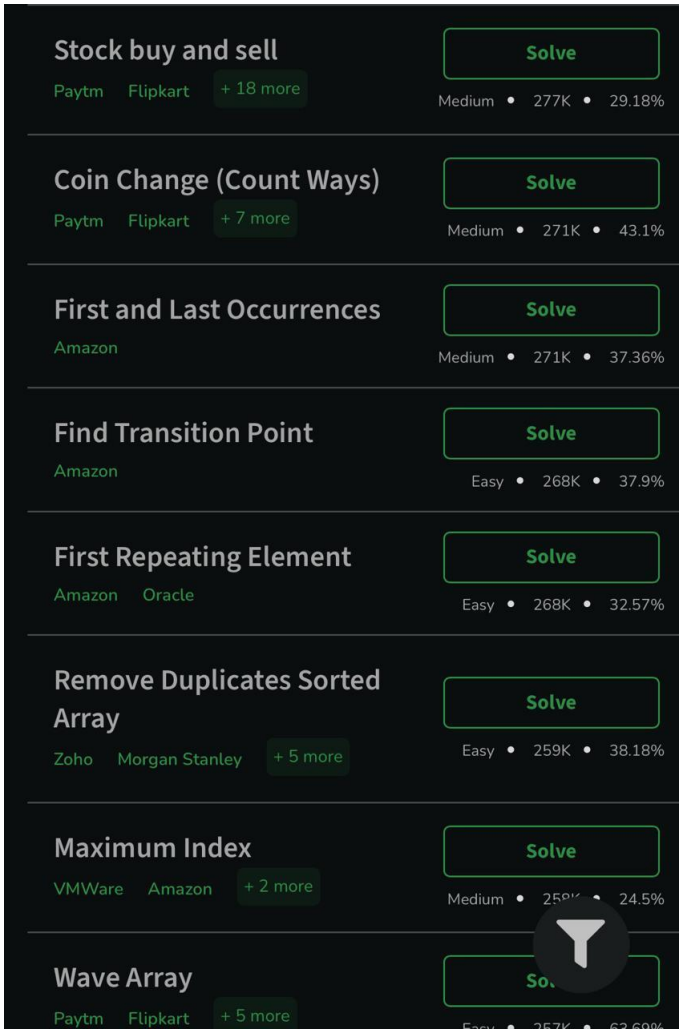
DSA Practice

Date: 14/11/2024

Problems:

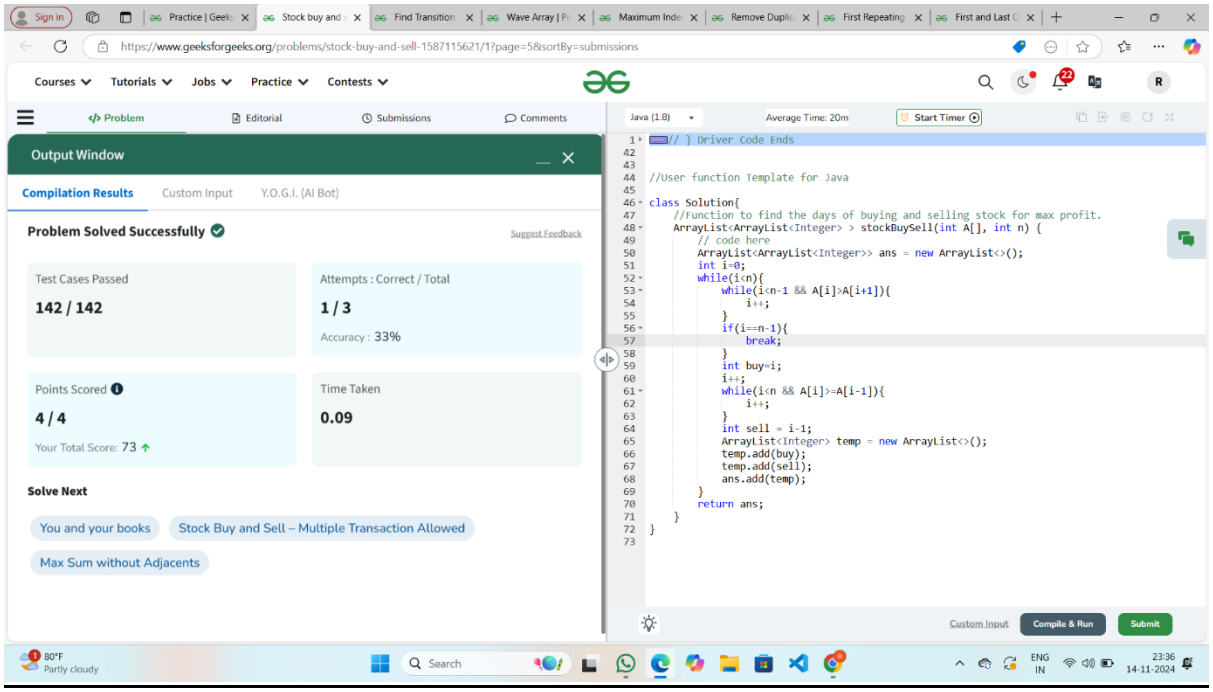


Problem 1:

```
class Solution{
    //Function to find the days of buying and selling stock for max profit.
    ArrayList<ArrayList<Integer> > stockBuySell(int A[], int n) {
        // code here
        ArrayList<ArrayList<Integer>> ans = new ArrayList<>();
        int i=0;
        while(i<n){
```

```
            while(i<n-1 && A[i]>A[i+1]){
                i++;
            }
            if(i==n-1){
                break;
            }
            int buy=i;
            i++;
            while(i<n && A[i]>=A[i-1]){
                i++;
            }
            int sell = i-1;
            ArrayList<Integer> temp = new ArrayList<>();
            temp.add(buy);
            temp.add(sell);
            ans.add(temp);
        }
        return ans;
    }
}
```

Output:

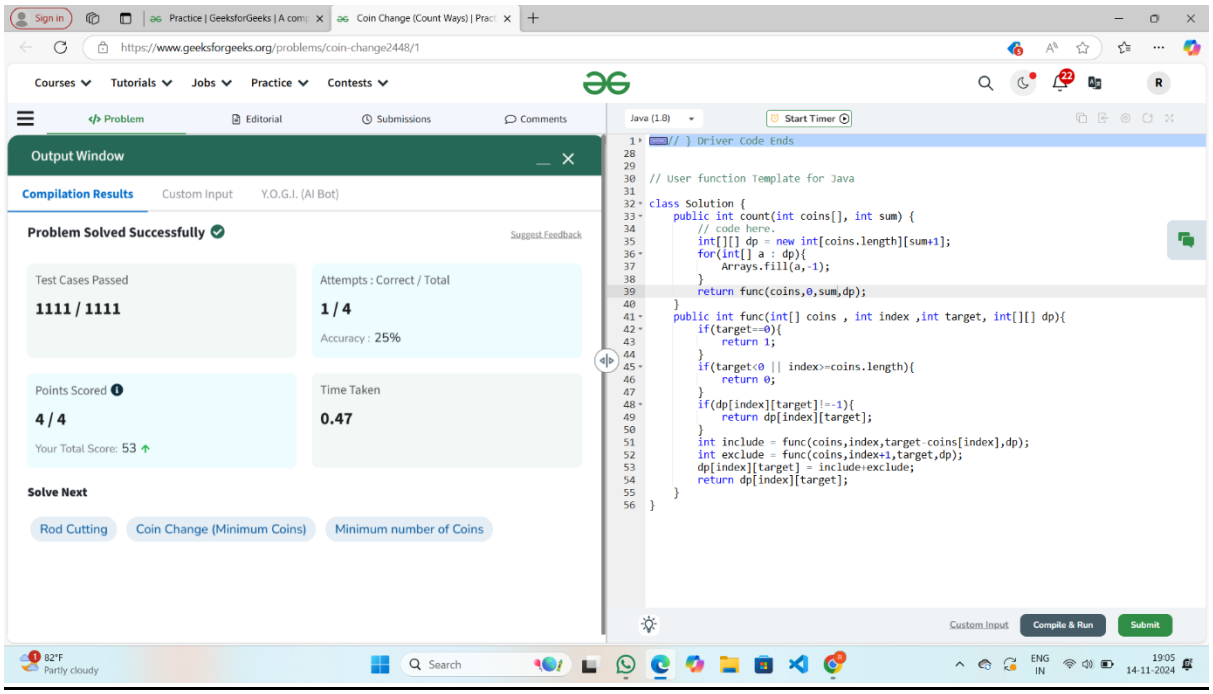


Problem 2:

```
class Solution {
    public int count(int coins[], int sum) {
        // code here.
        int[][] dp = new int[coins.length][sum+1];
        for(int[] a : dp){
            Arrays.fill(a,-1);
        }
        return func(coins,0,sum,dp);
    }
    public int func(int[] coins , int index ,int target, int[][] dp){
        if(target==0){
            return 1;
        }
        if(target<0 || index>=coins.length){
            return 0;
```

```
        }
        if(dp[index][target]!=-1){
            return dp[index][target];
        }
        int include = func(coins,index,target-coins[index],dp);
        int exclude = func(coins,index+1,target,dp);
        dp[index][target] = include+exclude;
        return dp[index][target];
    }
}
```

Output:

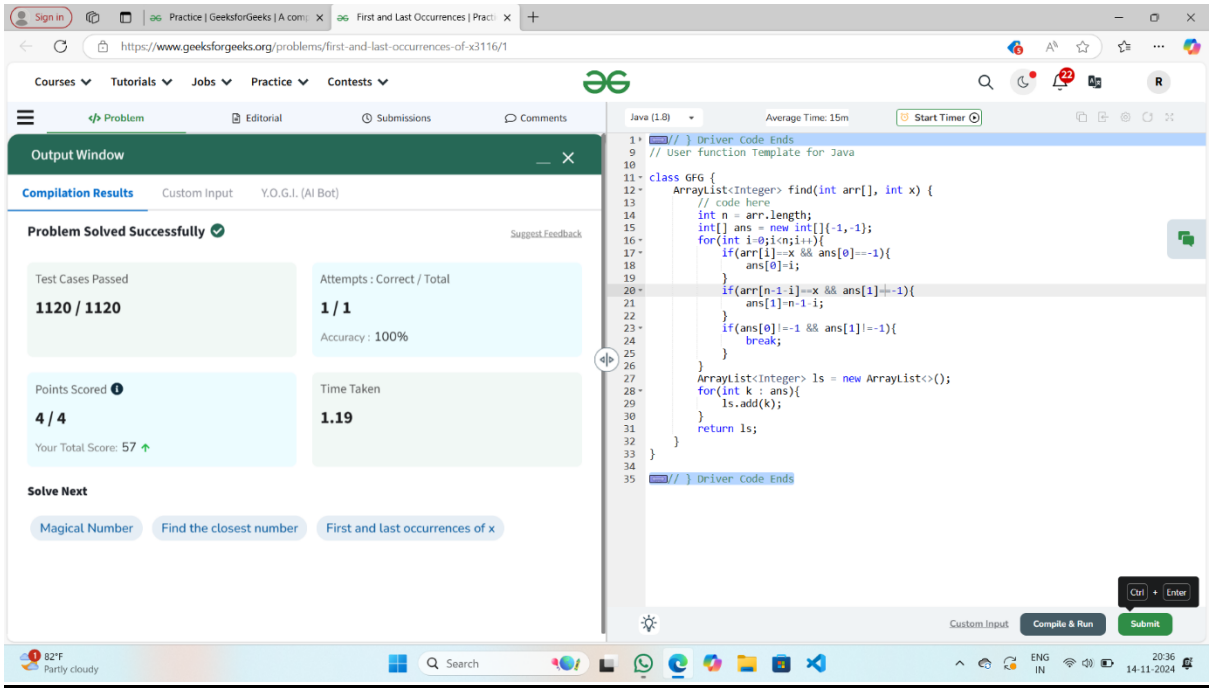


Problem 3:

```
class GFG {
    ArrayList<Integer> find(int arr[], int x) {
        // code here
        int n = arr.length;
```

```
        int[] ans = new int[]{-1,-1};
        for(int i=0;i<n;i++){
            if(arr[i]==x && ans[0]==-1){
                ans[0]=i;
            }
            if(arr[n-1-i]==x && ans[1]==-1){
                ans[1]=n-1-i;
            }
            if(ans[0]!=-1 && ans[1]!=-1){
                break;
            }
        }
        ArrayList<Integer> ls = new ArrayList<>();
        for(int k : ans){
            ls.add(k);
        }
        return ls;
    }
}
```

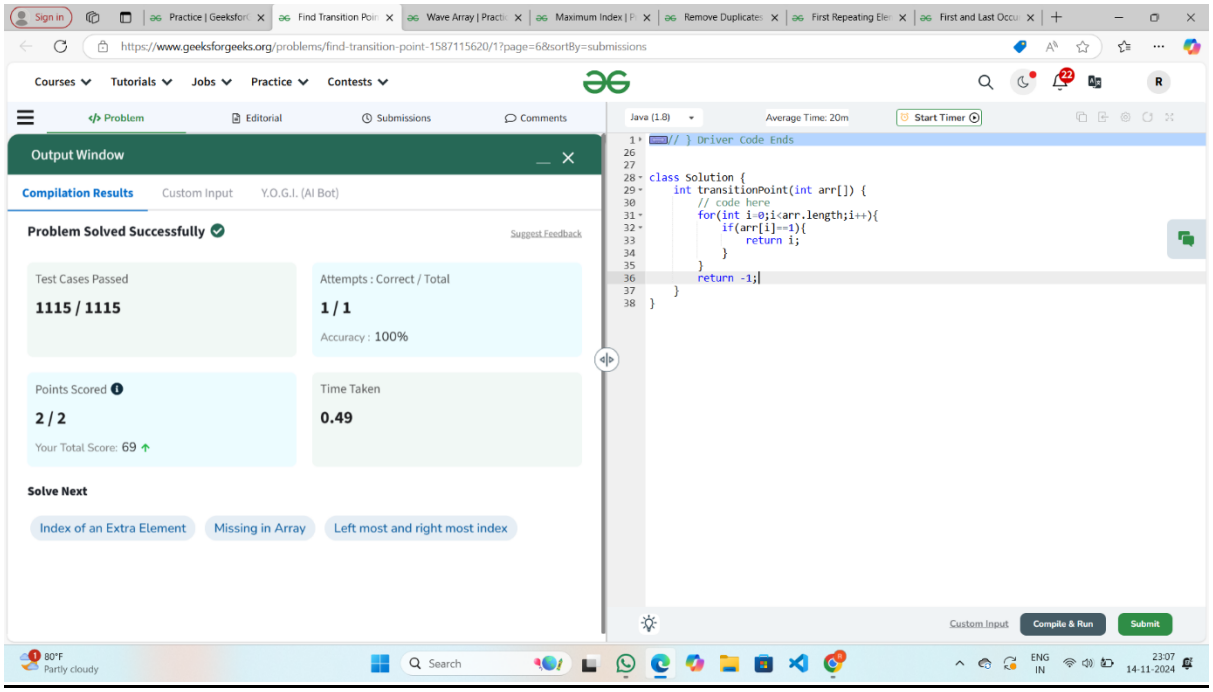
Output:



Problem 4:

```
class Solution {
    int transitionPoint(int arr[]) {
        // code here
        for(int i=0;i<arr.length;i++){
            if(arr[i]==1){
                return i;
            }
        }
        return -1;
    }
}
```

Output:

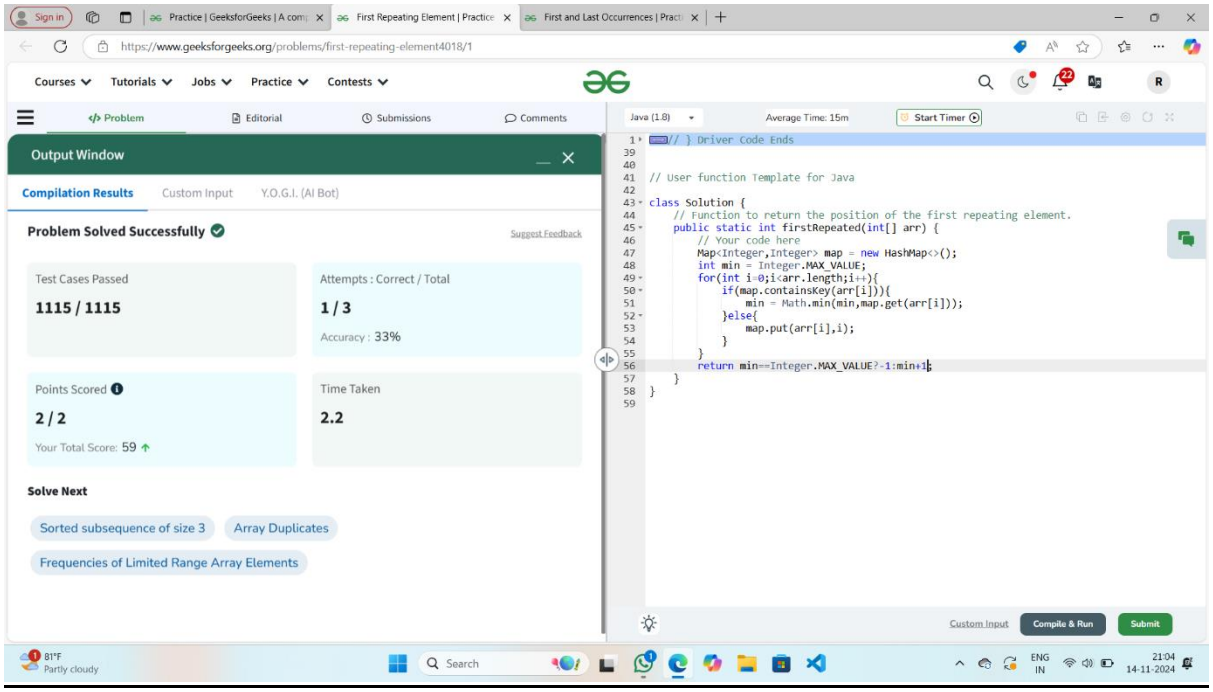


Problem 5:

```
class Solution {
    // Function to return the position of the first repeating element.
    public static int firstRepeated(int[] arr) {
        // Your code here
        Map<Integer,Integer> map = new HashMap<>();
        int min = Integer.MAX_VALUE;
        for(int i=0;i<arr.length;i++){
            if(map.containsKey(arr[i])){
                min = Math.min(min,map.get(arr[i]));
            }else{
                map.put(arr[i],i);
            }
        }
        return min==Integer.MAX_VALUE?-1:min+1;
    }
```

}

Output:

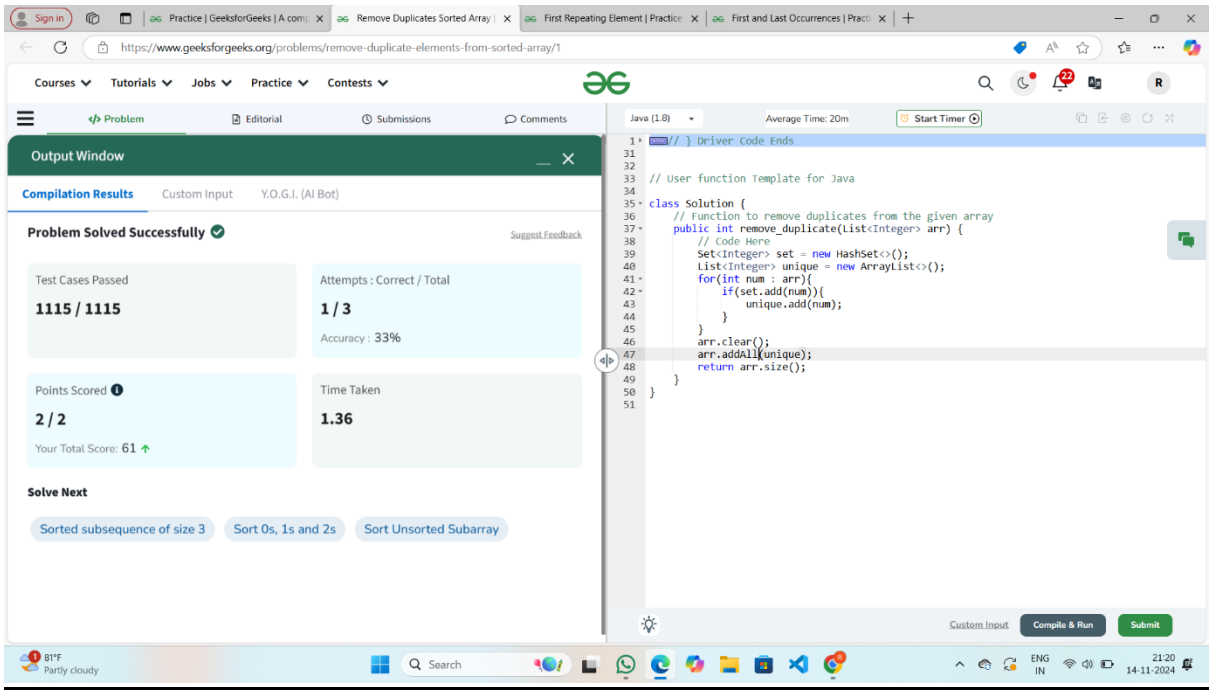


Problem 6:

```
class Solution {
    // Function to remove duplicates from the given array
    public int remove_duplicate(List<Integer> arr) {
        // Code Here
        Set<Integer> set = new HashSet<>();
        List<Integer> unique = new ArrayList<>();
        for(int num : arr){
            if(set.add(num)){
                unique.add(num);
            }
        }
        arr.clear();
        arr.addAll(unique);
```

```
        return arr.size();
    }
}
```

Output:

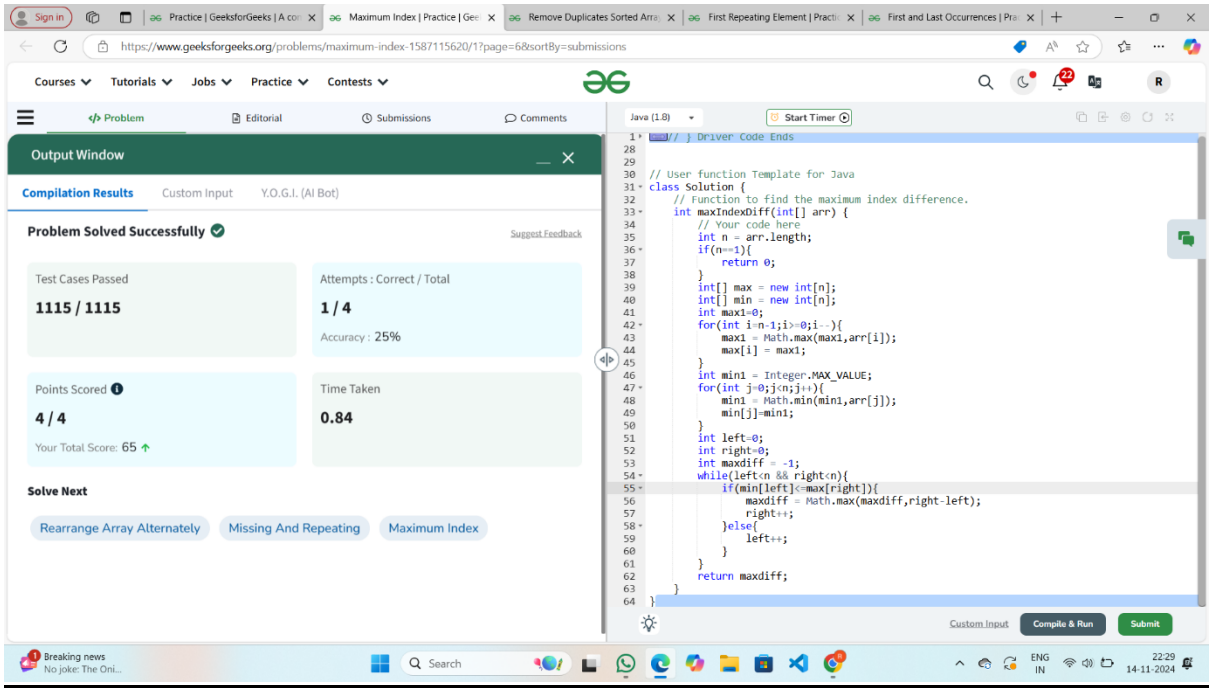


Problem 7:

```
class Solution {
    // Function to find the maximum index difference.
    int maxIndexDiff(int[] arr) {
        // Your code here
        int n = arr.length;
        if(n==1){
            return 0;
        }
        int[] max = new int[n];
        int[] min = new int[n];
        int max1=0;
```

```
        for(int i=n-1;i>=0;i--){
            max1 = Math.max(max1,arr[i]);
            max[i] = max1;
        }
        int min1 = Integer.MAX_VALUE;
        for(int j=0;j<n;j++){
            min1 = Math.min(min1,arr[j]);
            min[j]=min1;
        }
        int left=0;
        int right=0;
        int maxdiff = -1;
        while(left<n && right<n){
            if(min[left]<=max[right]){
                maxdiff = Math.max(maxdiff,right-left);
                right++;
            }else{
                left++;
            }
        }
        return maxdiff;
    }
}
```

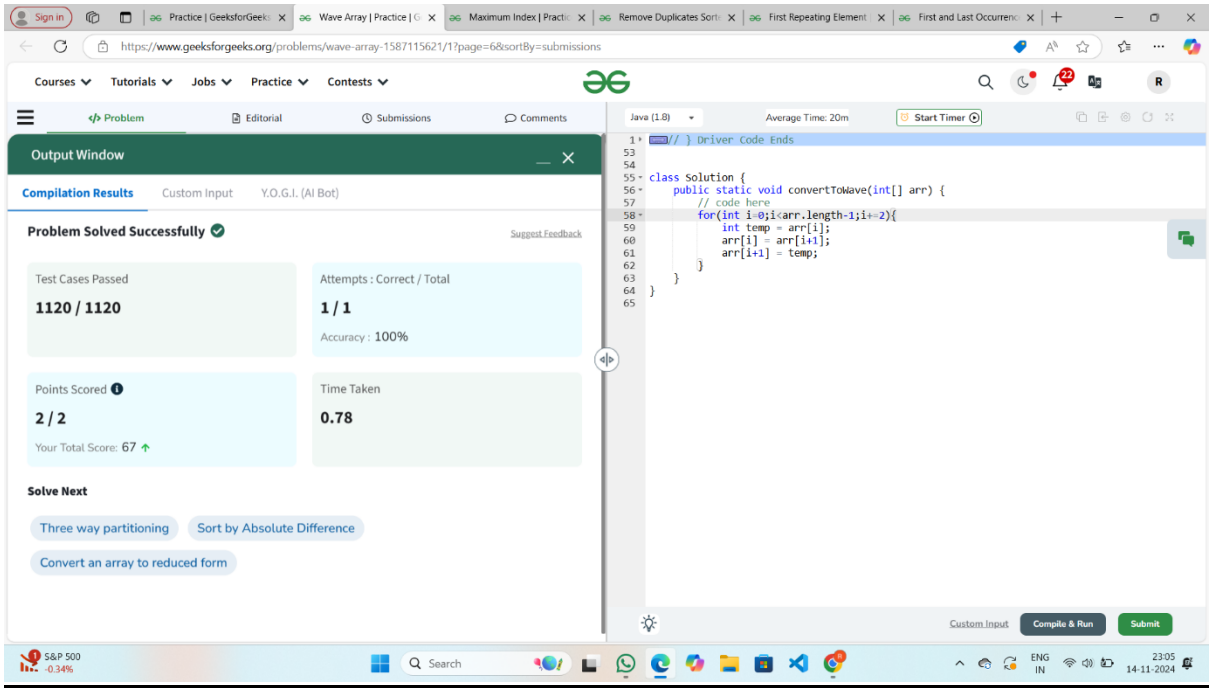
Output:



Problem 8:

```
class Solution {
    public static void convertToWave(int[] arr) {
        // code here
        for(int i=0;i<arr.length-1;i+=2){
            int temp = arr[i];
            arr[i] = arr[i+1];
            arr[i+1] = temp;
        }
    }
}
```

Output:


Practice | GeeksforGeeks
Wave Array | Practice |
Maximum Index | Practic
Remove Duplicates Sorte
First Repeating Element
First and Last Occurrenc
https://www.geeksforgeeks.org/problems/wave-array-1587115621/1?page=6&sortBy=submissions
Courses
Tutorials
Jobs
Practice
Contests
Problem
Editorial
Submissions
Comments
Output Window
Compilation Results
Custom Input
Y.O.G.I. (AI Bot)
Problem Solved Successfully
Suggest Feedback
Test Cases Passed
1120 / 1120
Attempts : Correct / Total
1 / 1
Accuracy : 100%
Points Scored
2 / 2
Your Total Score: 67
Time Taken
0.78
Solve Next
Three way partitioning
Sort by Absolute Difference
Convert an array to reduced form
Java (1.8)
Average Time: 20m
Start Timer
// } Driver Code Ends
class Solution {
public static void convertToWave(int[] arr) {
// code here
for(int i=0;i<arr.length-1;i+=2){
int temp = arr[i];
arr[i] = arr[i+1];
arr[i+1] = temp;
}
}
}
Custom Input
Compile & Run
Submit
S&P 500
-0.34%
Search
ENG
IN
23:05
14-11-2024