**DSA PRACTICE – 5- 14/11/2024**

**1.BUY AND SELL STOCKS**

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<ArrayList<Integer>>();

for(int i=0; i<n-1;i++){

if(A[i+1]>A[i]){

ArrayList<Integer> al = new ArrayList<>();

al.add(i);

al.add(i+1);

ans.add(al);

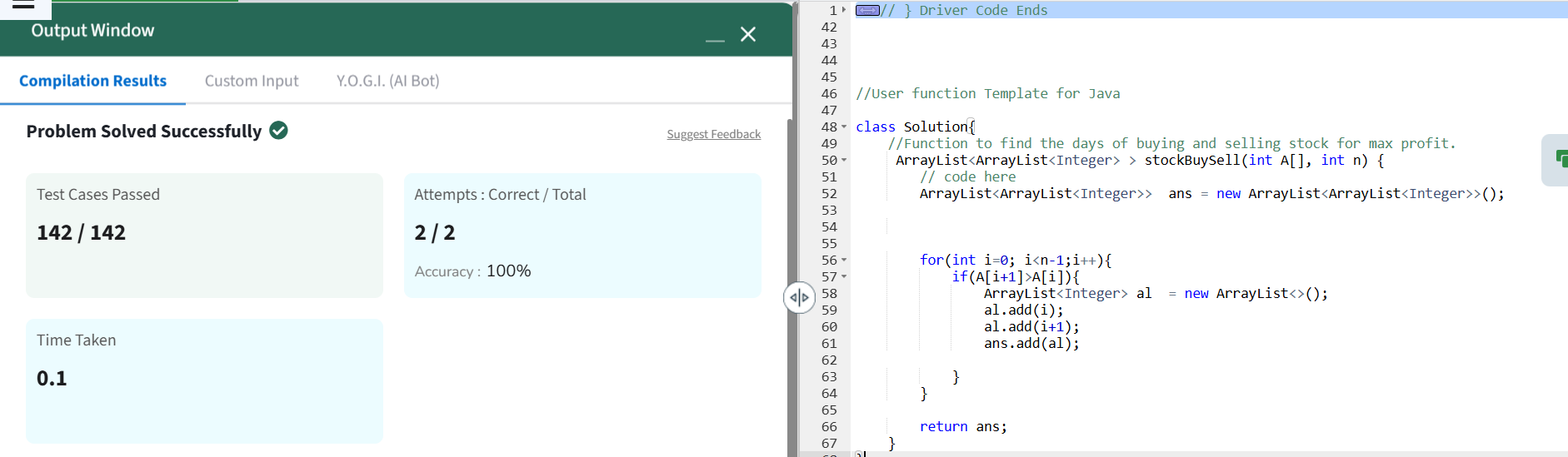
}

}

return ans;

}

}



Time Complexity : O(n)

Space Complexity: O(n)

**2.COIN CHANGE**

class Solution {

public int count(int coins[], int sum) {

// code here.

int[] dp = new int[sum+1];

dp[0]=1;

for(int i : coins){

for(int j=i;j<=sum;j++){

dp[j]+=dp[j-i];

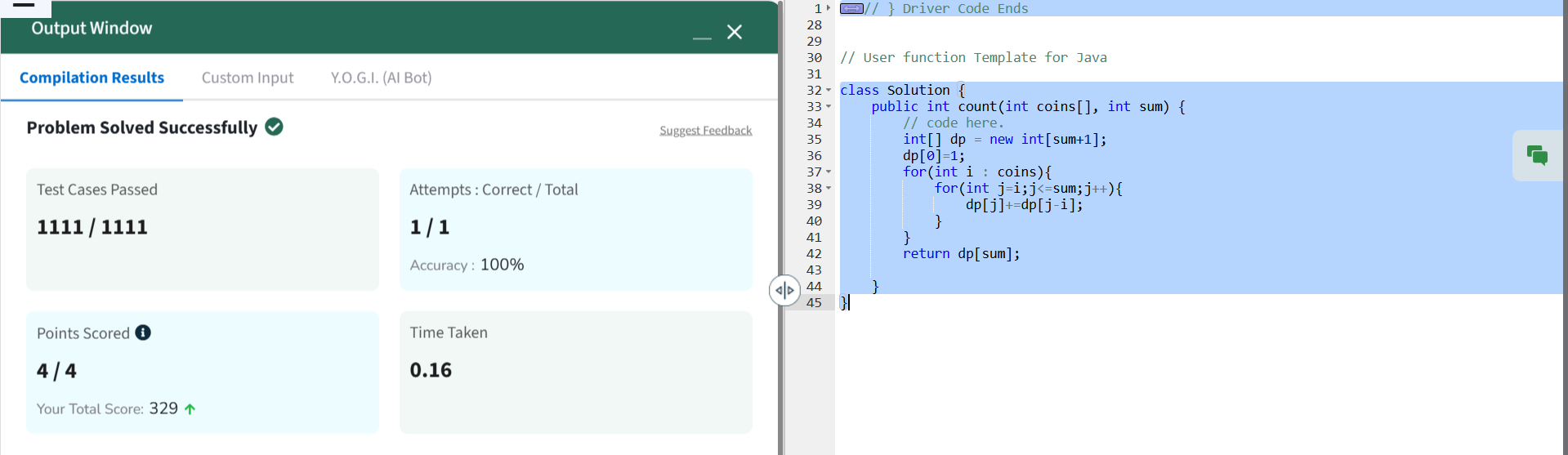
}

}

return dp[sum];

}

}



Time Complexity : O(n\*sum)

Space Complexity : O(sum)

**3.FIRST AND LAST OCCURRENCE**

class GFG {

ArrayList<Integer> find(int arr[], int x) {

ArrayList<Integer> res = new ArrayList<>();

int first = findFirstOccurrence(arr, x);

int last = findLastOccurrence(arr, x);

res.add(first);

res.add(last);

return res;

}

private int findFirstOccurrence(int[] arr, int x) {

int low = 0, high = arr.length - 1, result = -1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == x) {

result = mid;

high = mid - 1;

} else if (arr[mid] < x) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return result;

}

private int findLastOccurrence(int[] arr, int x) {

int low = 0, high = arr.length - 1, result = -1;

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == x) {

result = mid;

low = mid + 1;

} else if (arr[mid] < x) {

low = mid + 1;

} else {

high = mid - 1;

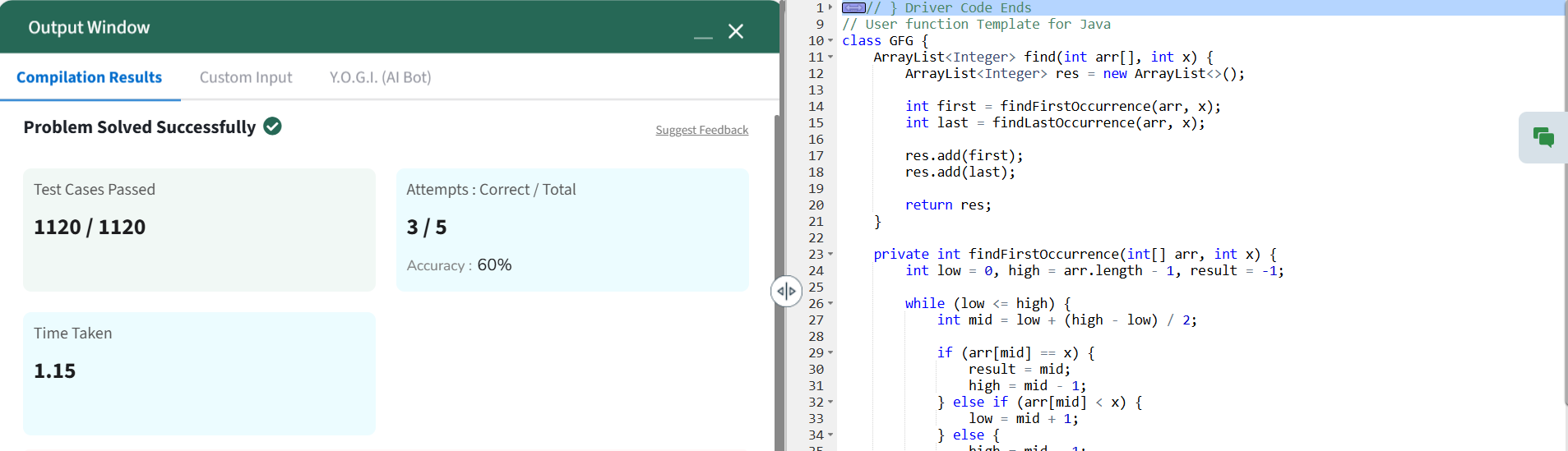
}

}

return result;

}

}

  
Time Complexity : O(log n)

Space Complexity : O(1)

**4.FIND TRANSITION POINT**

class Solution {

int transitionPoint(int arr[], int n) {

// code here

for(int i=0; i<n; i++){

if(arr[i]==1){

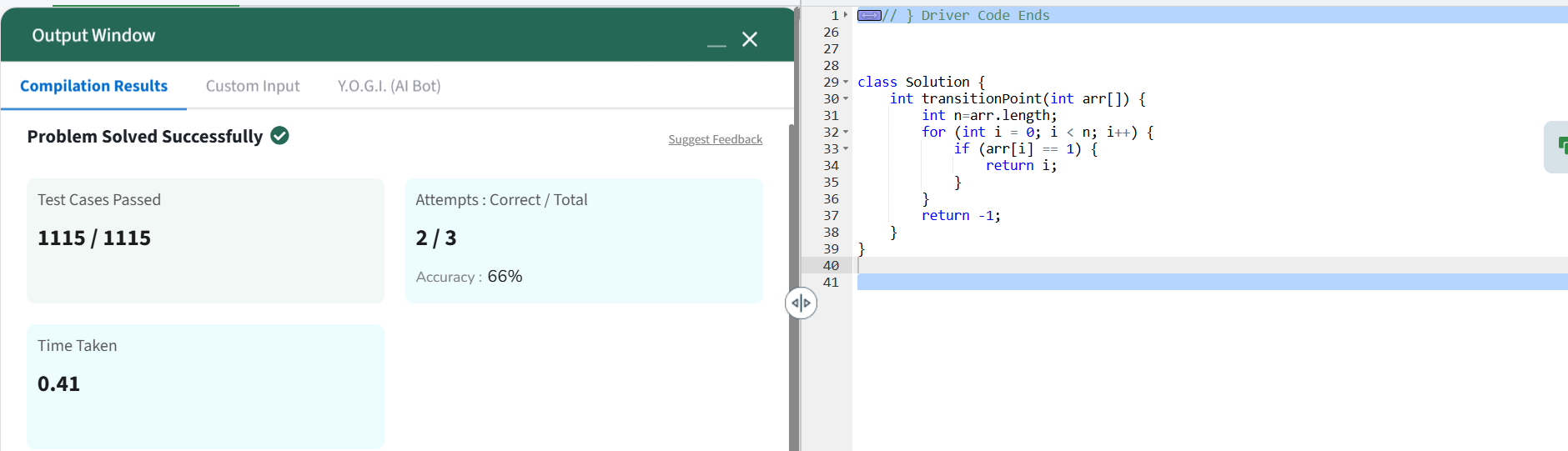
return i;

}

}

return -1;

}

}  
  
Time Complexity : O(n)

Space Complexity : O(1)

**5.FIRST REPEATING ELEMENT**class Solution {

public static int firstRepeated(int[] arr) {

HashMap<Integer, Integer> map = new HashMap<>();

ArrayList<Integer> a = new ArrayList<>();

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

if (map.containsKey(arr[i])) {

a.add(map.get(arr[i]) + 1);

} else {

map.put(arr[i], i);

}

}

if (a.isEmpty()) {

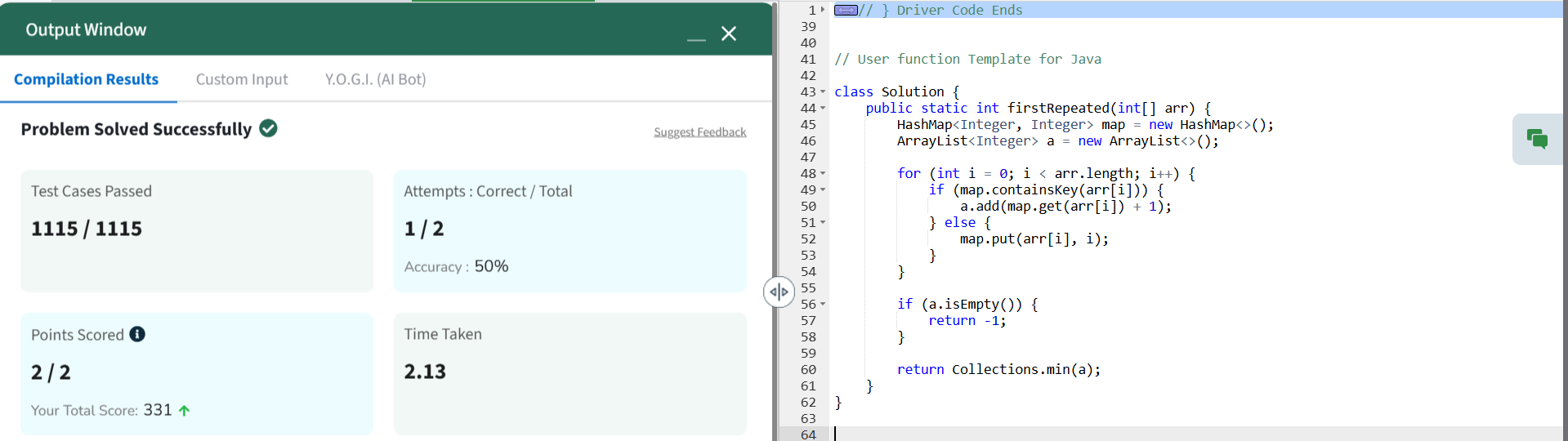
return -1;

}

return Collections.min(a);

}

}

  
Time Complexity : O(n)

Space Complexity : O(n)

**6. REMOVE DUPLICATES SORTED ARRAY**class Solution {

public int remove\_duplicate(List<Integer> arr) {

if (arr == null || arr.size() == 0) {

return 0;

}

int n = arr.size();

int index = 1;

for (int i = 1; i < n; i++) {

if (!arr.get(i).equals(arr.get(i - 1))) {

arr.set(index, arr.get(i));

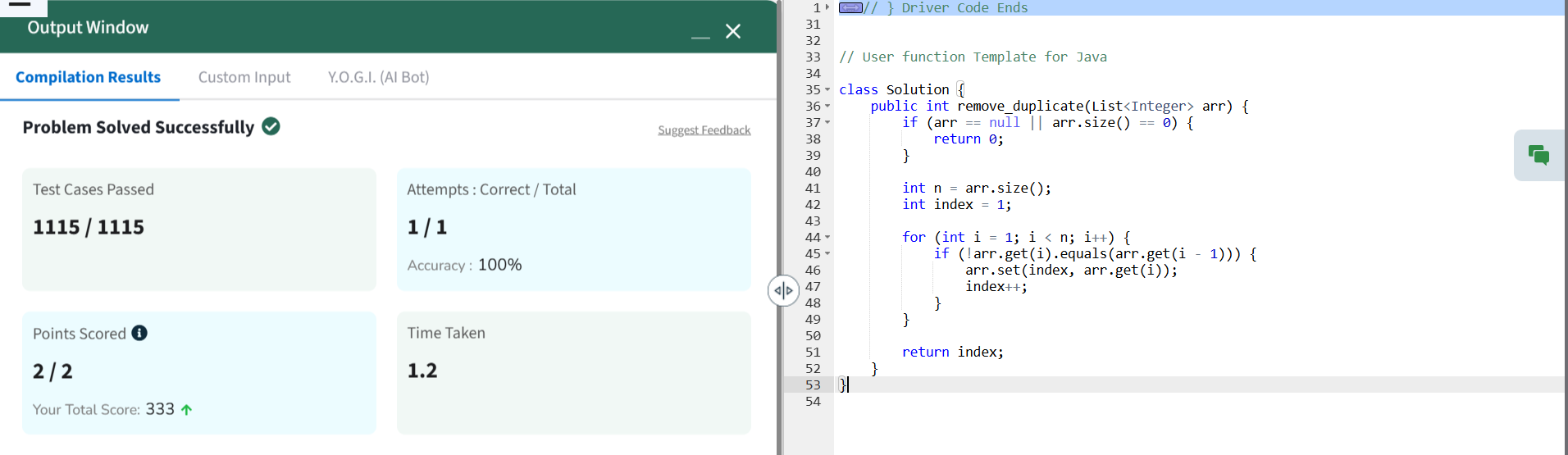
index++;

}

}

return index;

}

}  
  
Time Complexity : O(n)

Space Complexity : O(1)

**7. MAXIMUM INDEX**class Solution {

public int maxIndexDiff(int[] arr) {

int n = arr.length;

int maxElem = -1;

int ans = 0;

int[] rightMax = new int[n];

for (int i = n - 1; i >= 0; i--) {

maxElem = Math.max(maxElem, arr[i]);

rightMax[i] = maxElem;

}

int i = 0, j = 0;

while (i < n && j < n) {

if (rightMax[j] >= arr[i]) {

ans = Math.max(ans, j - i);

j++;

} else {

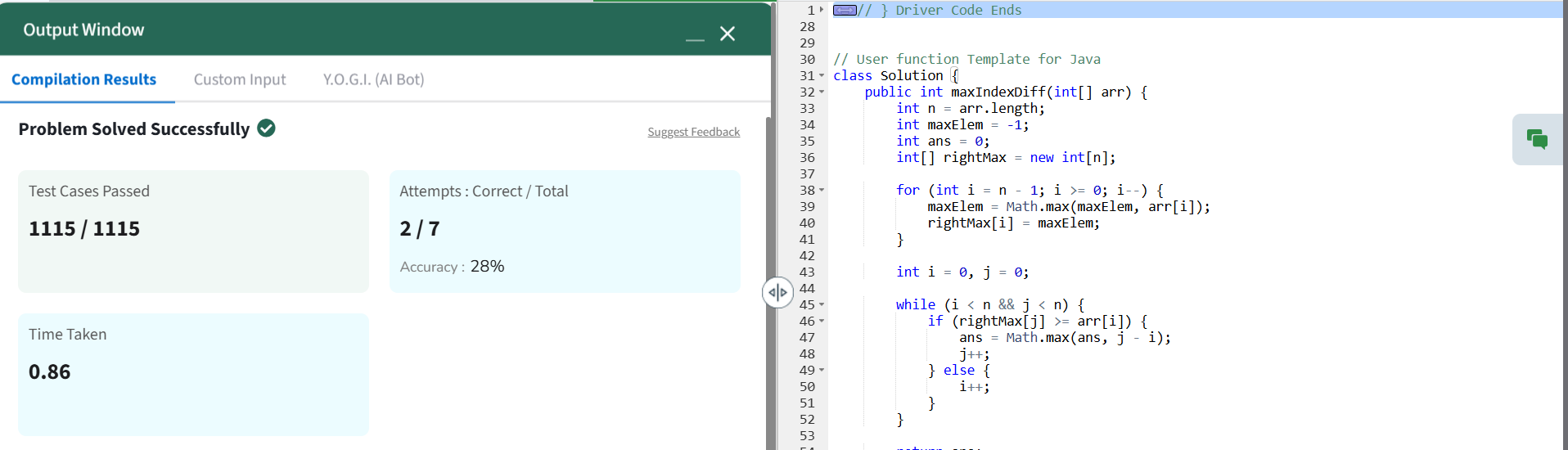
i++;

}

}

return ans;

}

}  
  
Time Complexity : O(n)

Space Complexity : O(n)

**8. WAVE ARRAY**

class Solution {

public static void convertToWave(int[] arr) {

// code here

int l=0;

int r=l+1;

while(r<arr.length){

int temp =arr[l];

arr[l]=arr[r];

arr[r]=temp;

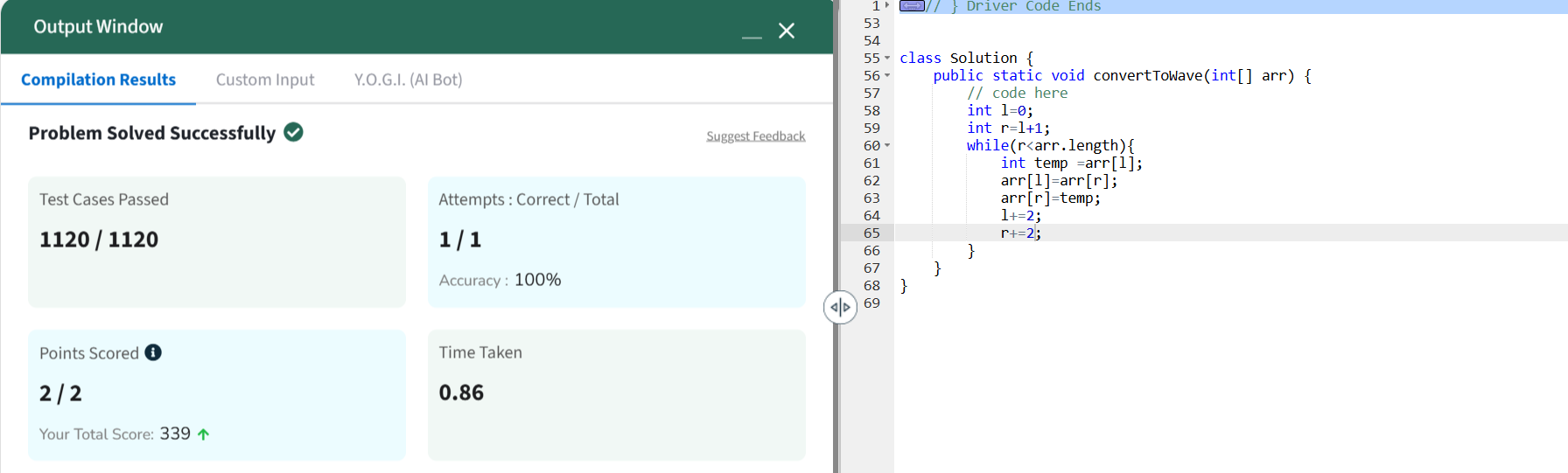
l+=2;

r+=2;

}

}

}



Time Complexity : O(n)

Space Complexity: O(1)