**DSA PRACTICE – 6 – 18/11/2024**

1. **BUBBLE SORT**

class Solution {

// Function to sort the array using bubble sort algorithm.

public static void bubbleSort(int arr[]) {

int temp;

boolean flag;

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

flag = false;

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

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

temp = arr[j];

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

arr[j + 1] = temp;

flag = true;

}

}

if(!flag) {

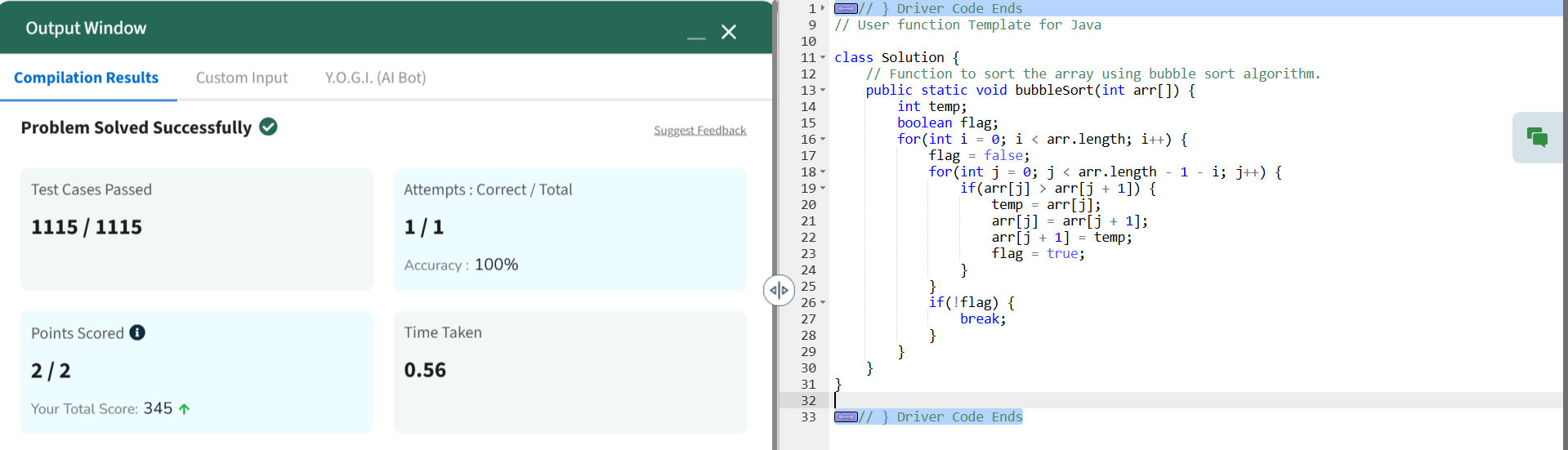
break;

}

}

}

}



**2.QUICK SORT**class Solution {

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; j++) {

if (arr[j] <= pivot) {

i++;

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

int temp = arr[i + 1];

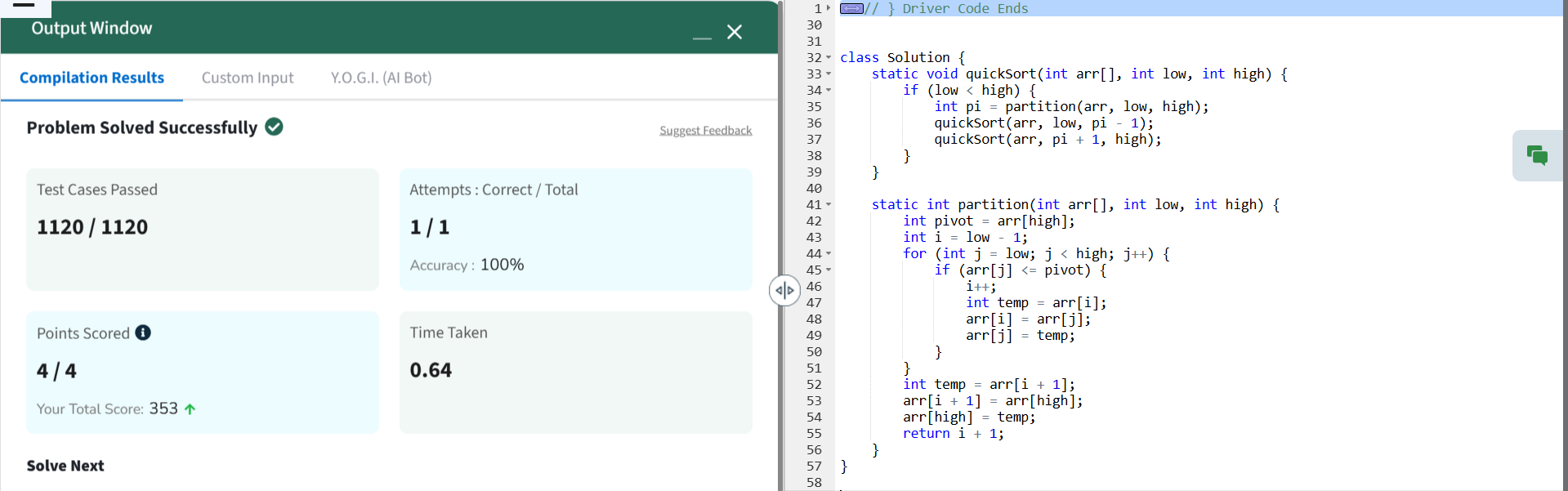
arr[i + 1] = arr[high];

arr[high] = temp;

return i + 1;

}

}



**3.NON REPEATING CHARACTER**class Solution {

// Function to find the first non-repeating character in a string.

static char nonRepeatingChar(String s) {

// Your code here

int[] charC = new int[26];

for(char ch: s.toCharArray()){

charC[ch-'a']++;

}

for(char ch: s.toCharArray()){

if(charC[ch-'a']==1){

return ch;

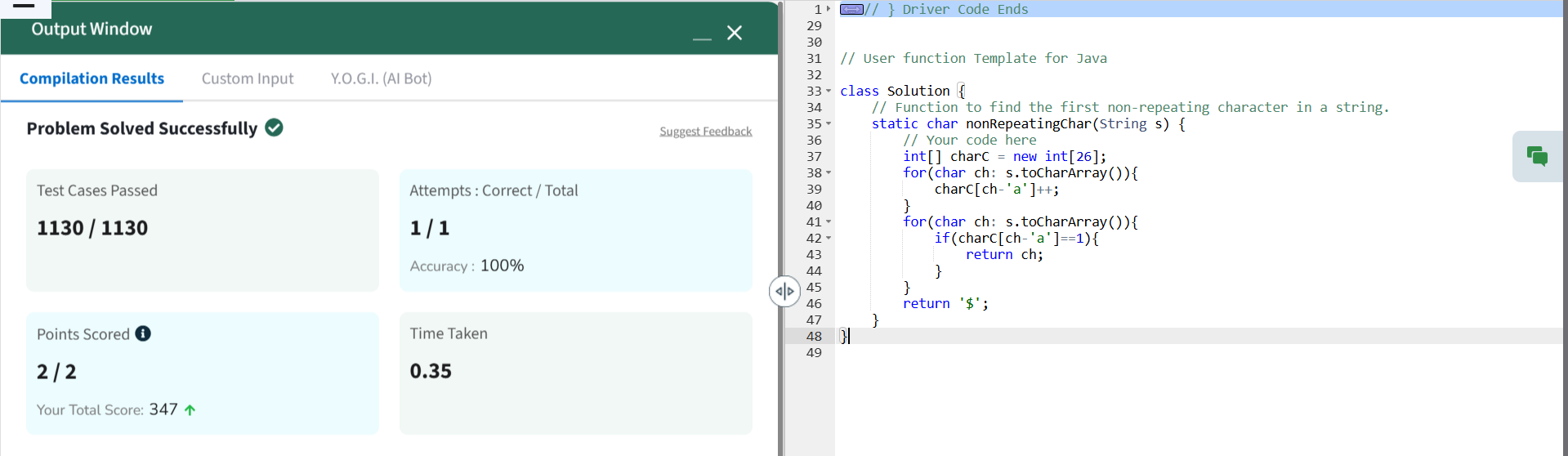
}

}

return '$';

}

}



**4. EDIT DISTANCE**  
class Solution {

public int editDistance(String s1, String s2) {

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;

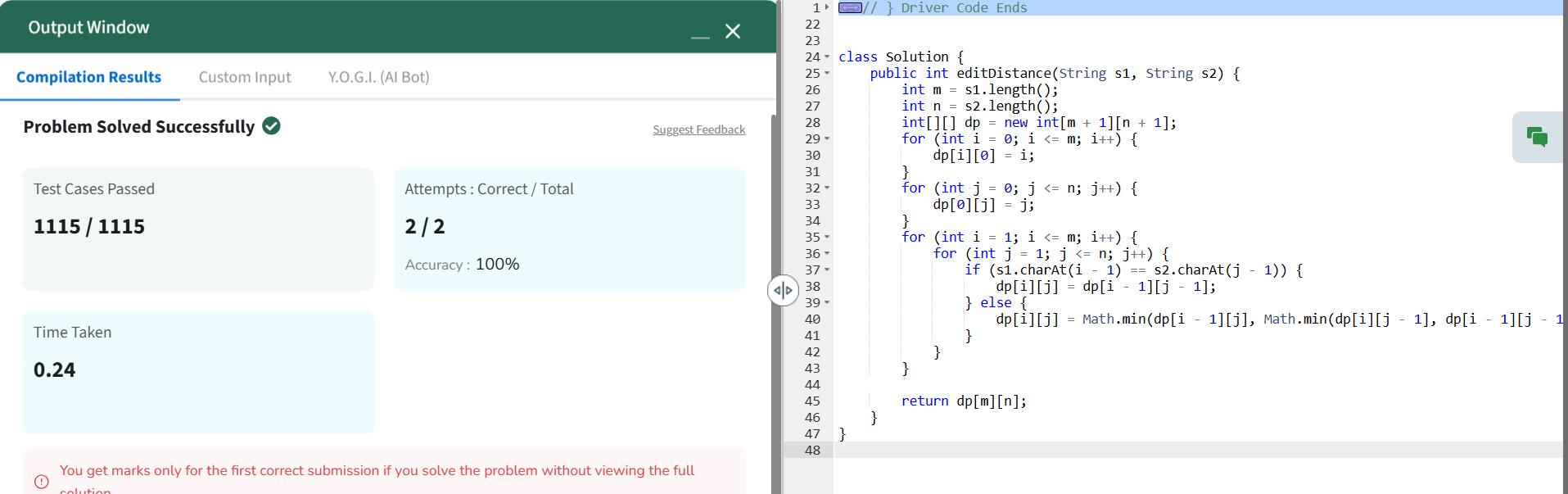
}

}

}

return dp[m][n];

}

}  


**5. K LARGEST ELEMENT**

class Solution {

// Function to find the k largest elements in decreasing order

static List<Integer> kLargest(int arr[], int k) {

PriorityQueue<Integer> minHeap = new PriorityQueue<>();

for (int num : arr) {

minHeap.add(num);

if (minHeap.size() > k) {

minHeap.poll();

}

}

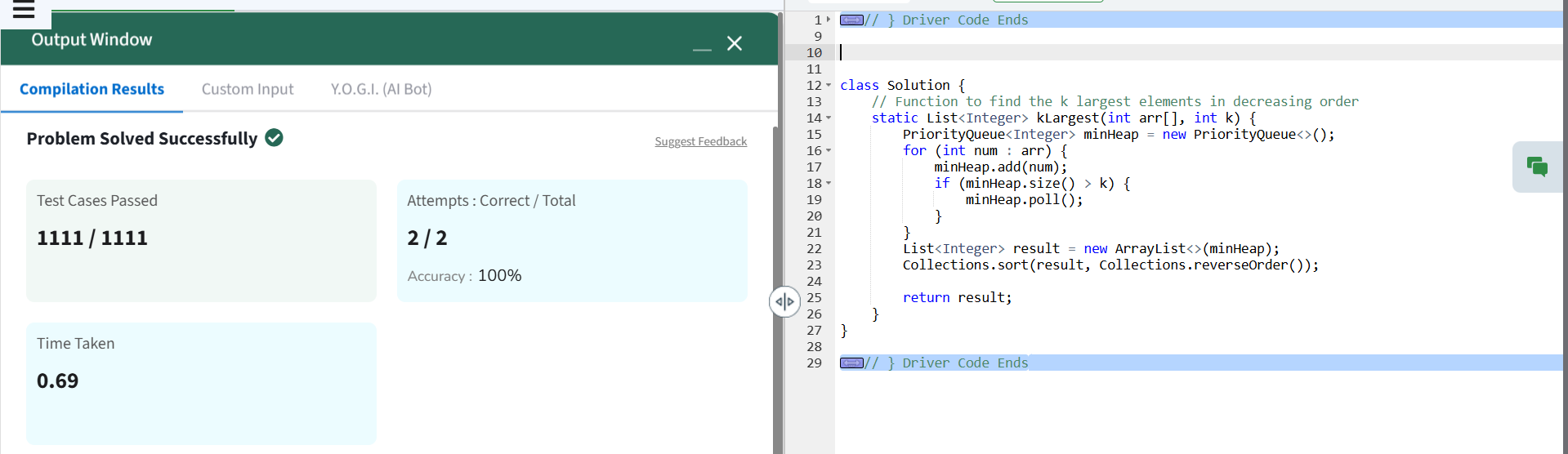
List<Integer> result = new ArrayList<>(minHeap);

Collections.sort(result, Collections.reverseOrder());

return result;

}

}



**6. FORM LARGEST NUMBER**class Solution {

public String MaxNumber(int arr[]) {

List<String> numList = new ArrayList<>();

for (int num : arr) {

numList.add(String.valueOf(num));

}

Collections.sort(numList, new Comparator<String>() {

public int compare(String a, String b) {

return (b + a).compareTo(a + b);

}

});

if (numList.get(0).equals("0")) {

return "0";

}

StringBuilder result = new StringBuilder();

for (String num : numList) {

result.append(num);

}

return result.toString();

}

}

