DSA ASSIGNMENT:

1.K-th smallest element

2.Minimize the heights-II

3.Parenthesis checker

4.Equilibrium point

5.Binary Search

6.Next greater element

7.Union of two arrays with duplicate elements  
  
1.Kth Smallest Element

class Solution {

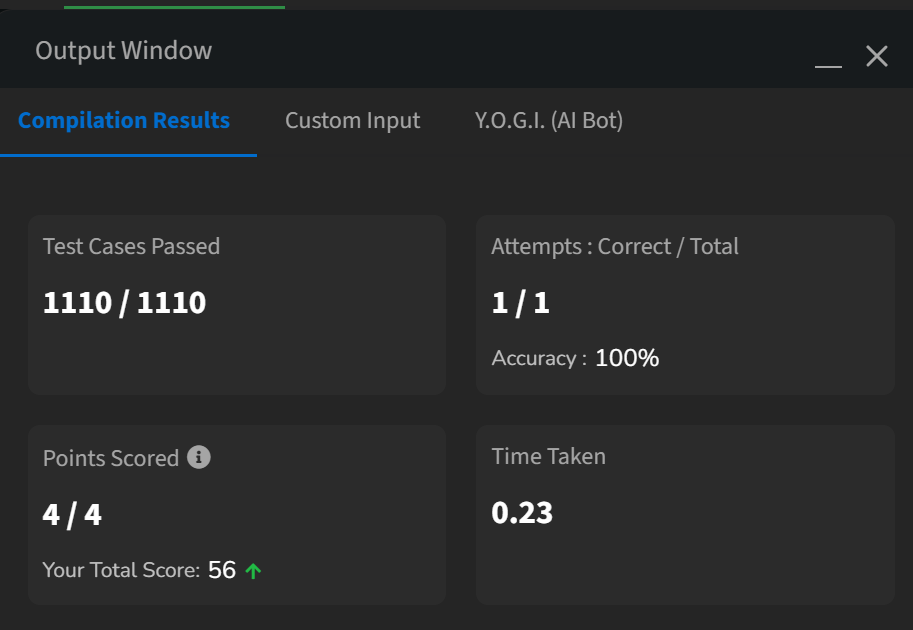
public static int kthSmallest(int[] arr, int k) {

// Your code here

Arrays.sort(arr);

return arr[k-1];

}}  
Output:



3. Parenthesis Checker:

class Solution {

static boolean isParenthesisBalanced(String s) {

Stack<Character> st=new Stack<>();

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

if(i=='('){

st.push(')');

}

else if(i=='{'){

st.push('}');

}

else if(i=='['){

st.push(']');

}

else if(st.isEmpty()||st.pop()!=i){

return false;

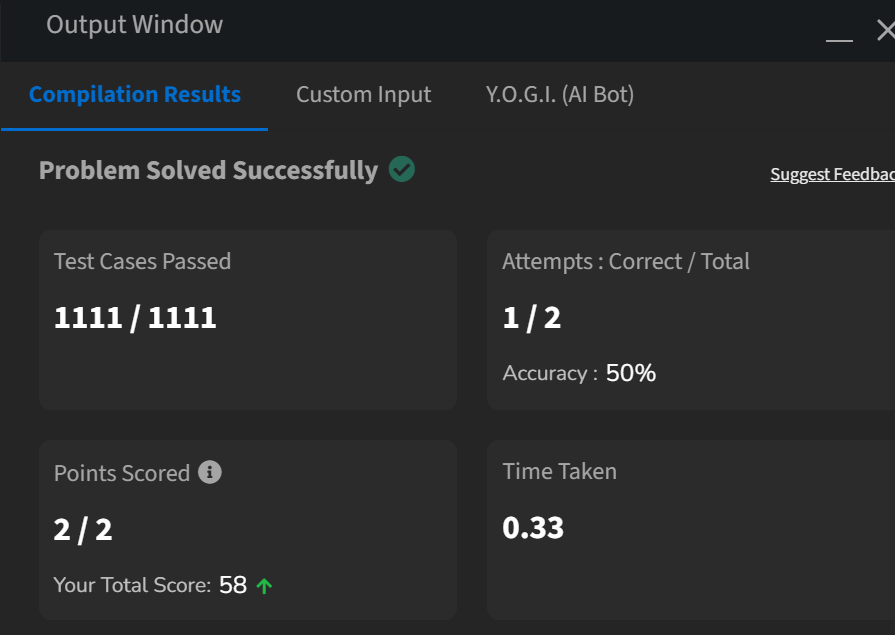
}

}

return st.isEmpty();

}

}  
Output:



4.Equilibrium point:

Class Solution{

public static int equilibriumPoint(int arr[]) {

int n = arr.length;

if (n == 1) {

return 1;

}

int totalSum = 0;

for (int num : arr) {

totalSum += num;

}

int leftSum = 0;

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

int rightSum = totalSum - arr[i] - leftSum;

if (leftSum == rightSum) {

return i+1;

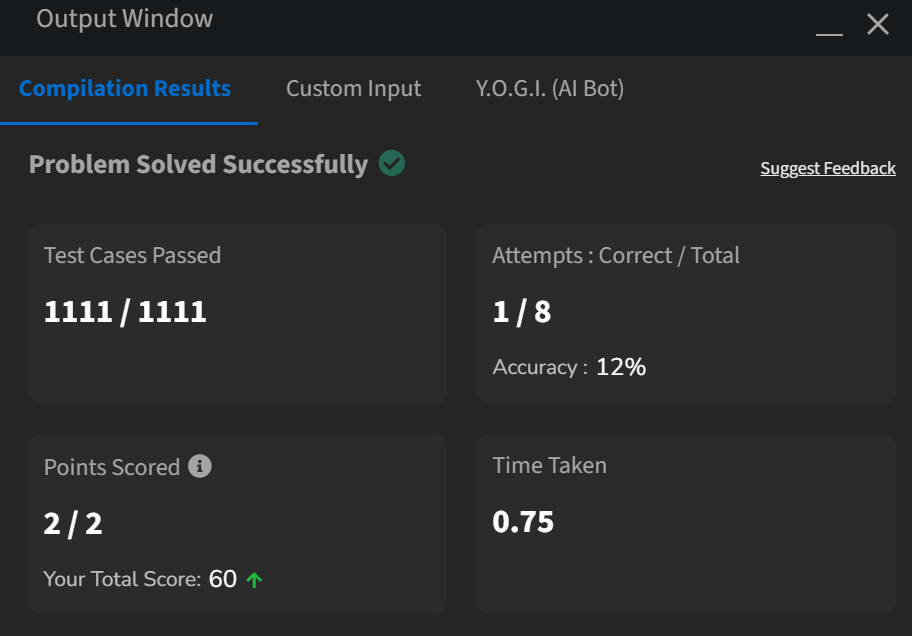
}

leftSum += arr[i];

}

return -1;} }

Output:



2. Minimize Heights 2:

class Solution {

int getMinDiff(int[] arr, int k) {

int n = arr.length;

Arrays.sort(arr);

int initialDifference = arr[n - 1] - arr[0];

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

if(arr[i]-k<0) continue;

int newMin = Math.min(arr[0] + k, arr[i] - k);

int newMax = Math.max(arr[i - 1] + k, arr[n - 1] - k);

initialDifference = Math.min(initialDifference, newMax - newMin);

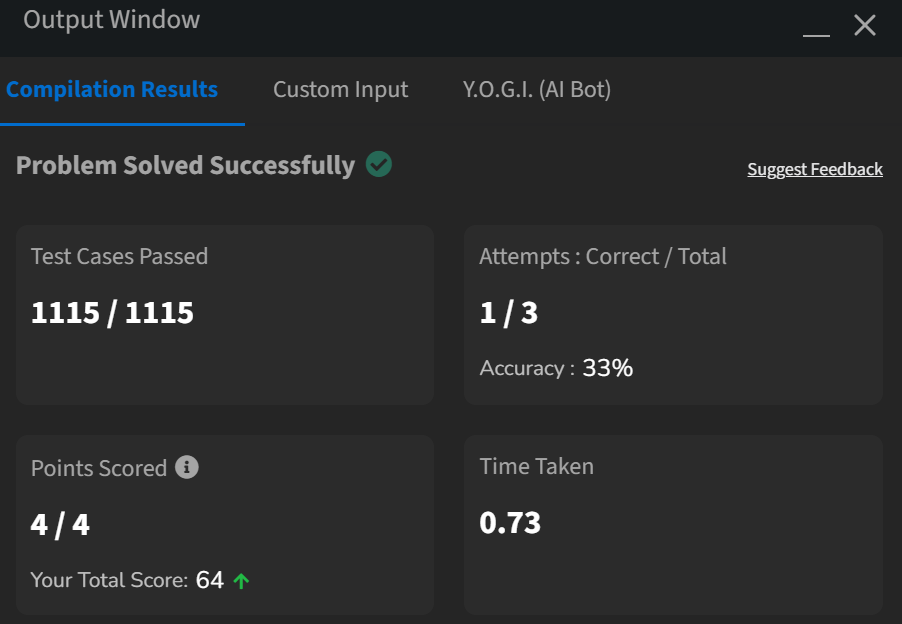
}

return initialDifference;

}

}

Output:



5.Binary Search:

class Solution {

public int search(int[] nums, int target) {

int left = 0; // initialize left pointer to 0

int right = nums.length – 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (nums[mid] == target) {

return mid; // return the middle index

} else if (nums[mid] < target) {

left = mid + 1;

} else {

right = mid – 1;

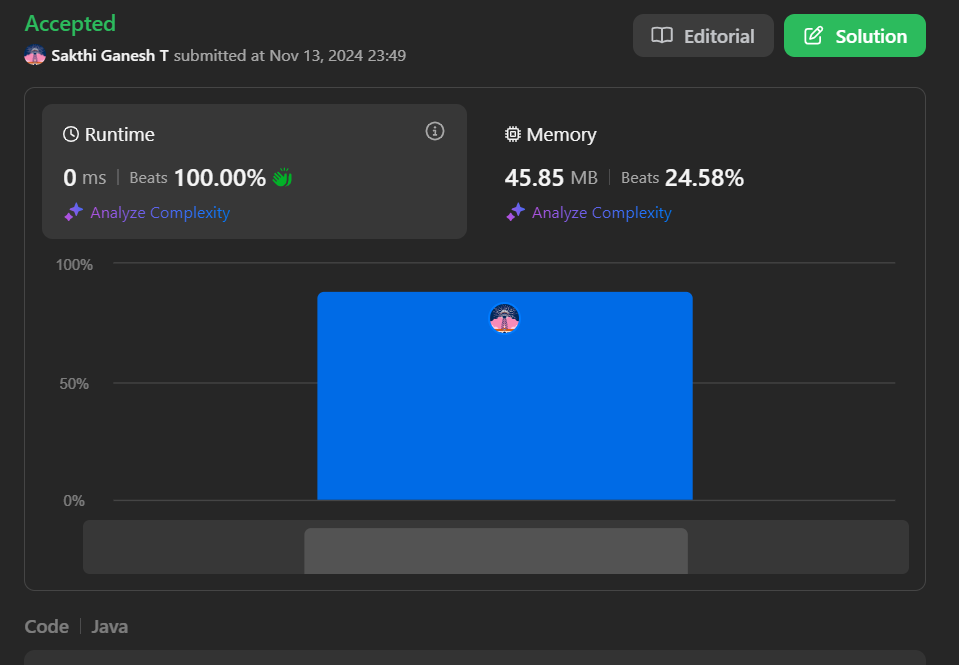
}

}

return -1;

}

}  
Output:



6. Union of two arrays with duplicate elements:

class Solution {

public static int findUnion(int a[], int b[]) {

// code here

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

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

{

if(!arr.contains(a[i]))

{

arr.add(a[i]);

}

}

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

{

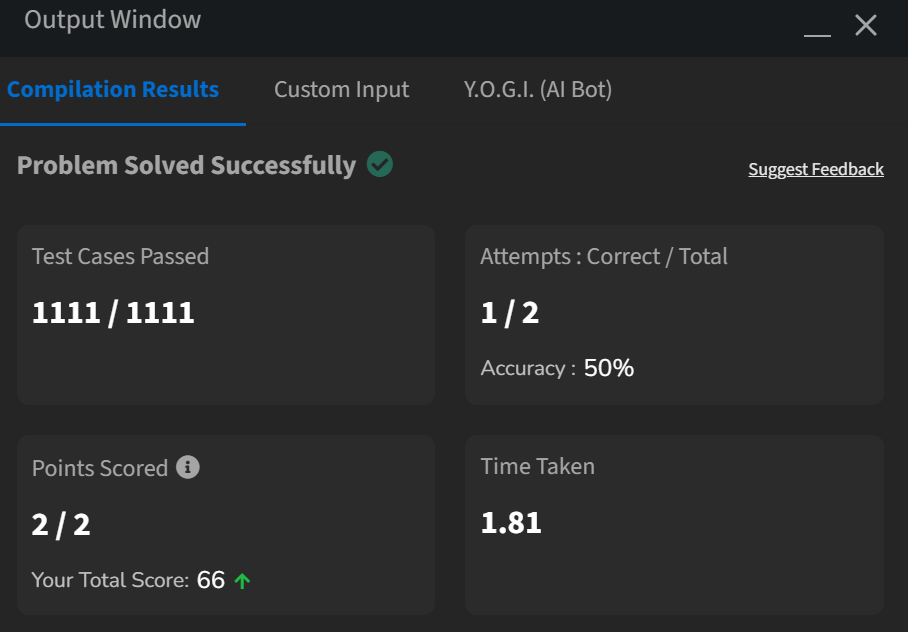
if(!arr.contains(b[i]))

{

arr.add(b[i]);} }

return arr.size();}}

Output:



7. Next Greater Element:

class Solution {

public ArrayList<Integer> nextLargerElement(int[] arr) {

// code here

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

int n=arr.length;

int next, i, j;

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

next = -1;

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

if (arr[i] < arr[j]) {

next = arr[j];

break;

}

}

a.add(next);

}

return a;

}

}  
output:

