Company Name: Google

Problem-1: Remove Duplicate element from sorted array

Code:

class Solution {

public:

vector<int> removeDuplicates(vector<int> &arr) {

vector<int> result;

if (arr.empty()) return result;

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

if (arr[i]!=arr[i+1]) {

result.push\_back(arr[i]);

}

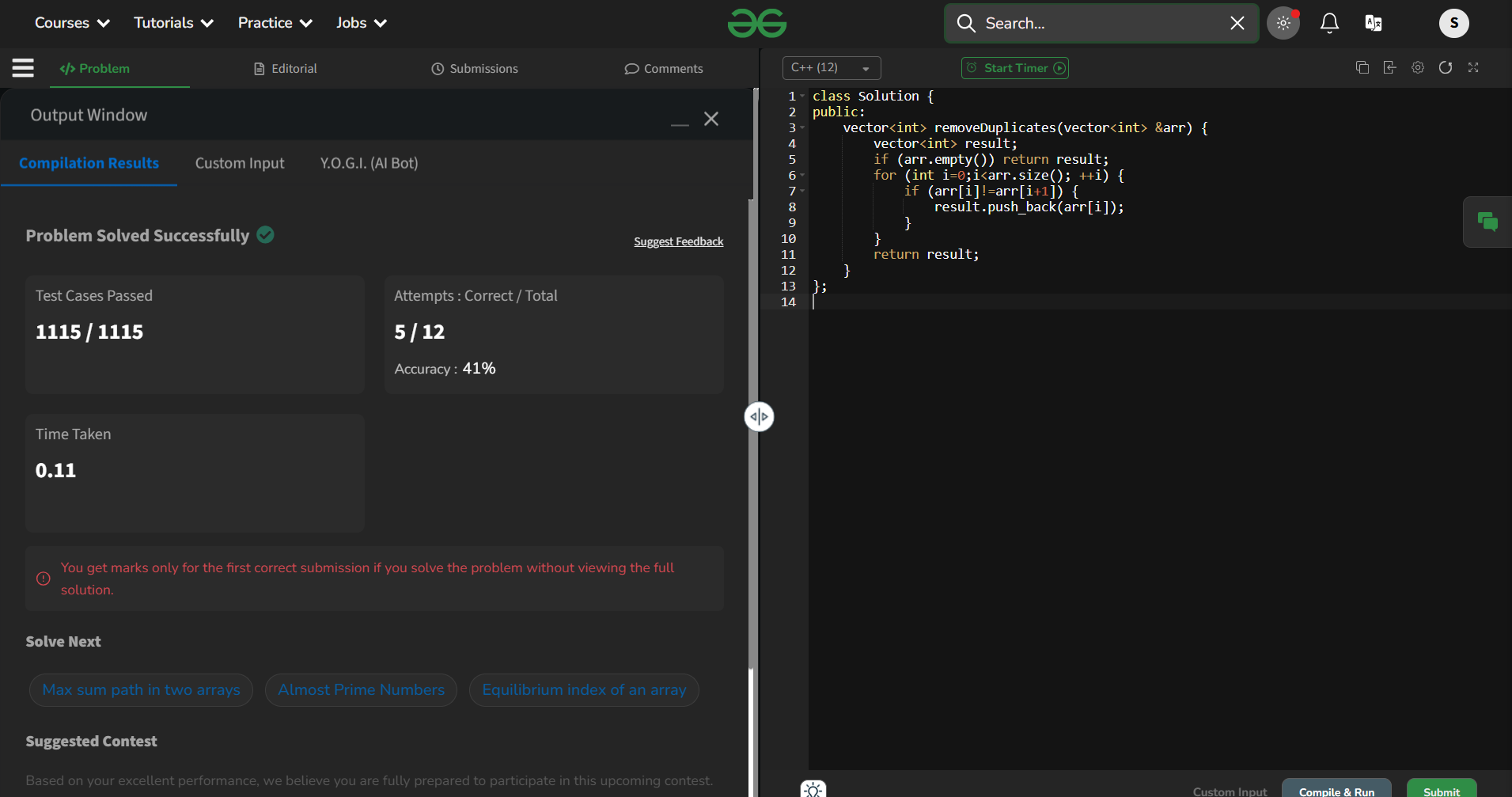
}

return result;

}

};

Output:



Problem-2:

Code:

class Solution {

public:

int smallestSubWithSum(int x, vector<int>& arr) {

// Your code goes here

int res = INT\_MAX;

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

int curr = 0;

for(int j=i;j<arr.size();j++){

curr += arr[j];

if(curr > x){

res = min(res, j-i+1);

break;

}

}

}

if (res == INT\_MAX){

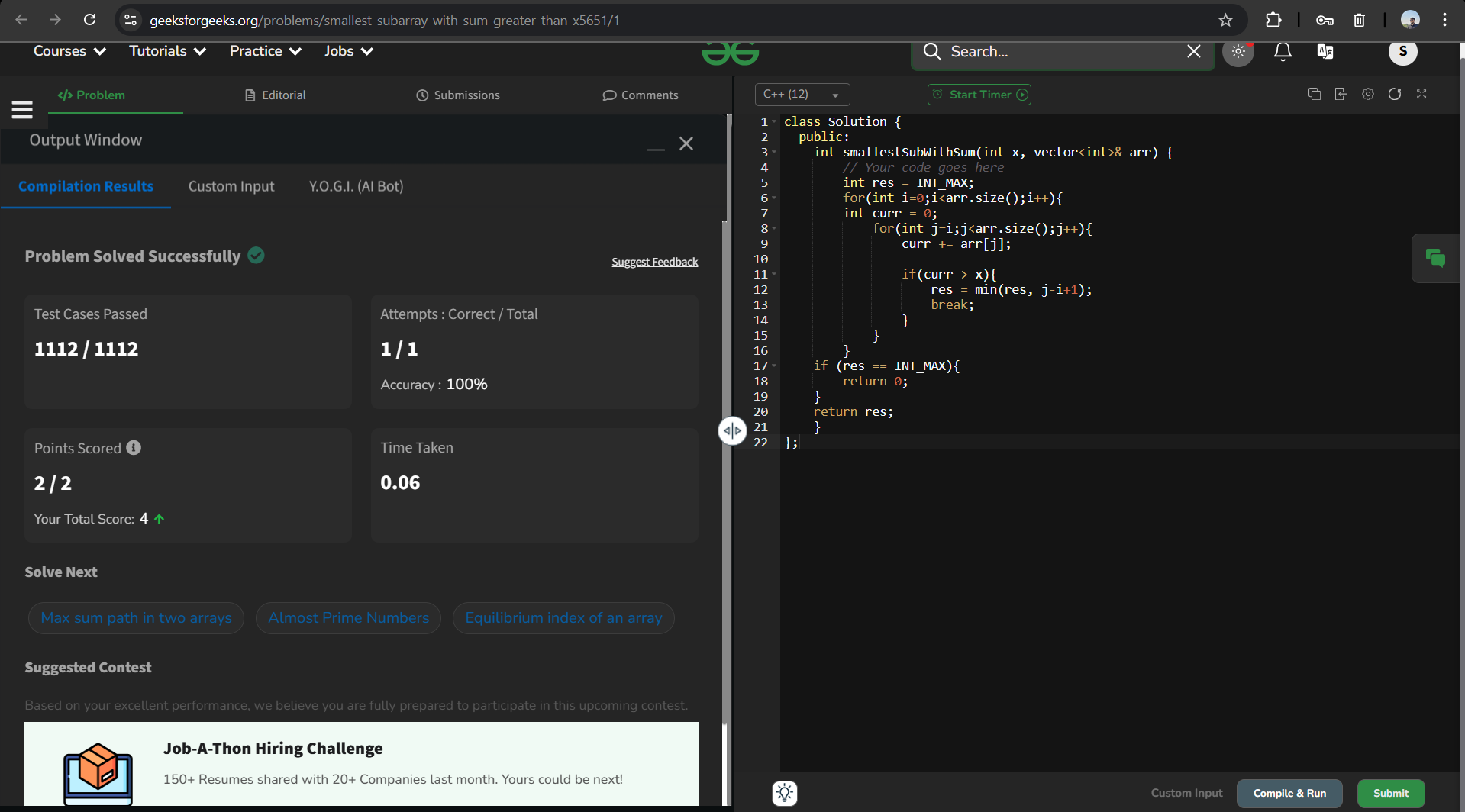
return 0;

}

return res;

}

};



Problem-3: **Left most and right most index**

**Code:**

class Solution {

public:

pair<long, long> indexes(vector<long long> v, long long x) {

// code here

long first=-1, last=-1;

for (long i = 0; i < v.size();++i) {

if (v[i]==x) {

if (first==-1) first=i;

last = i;

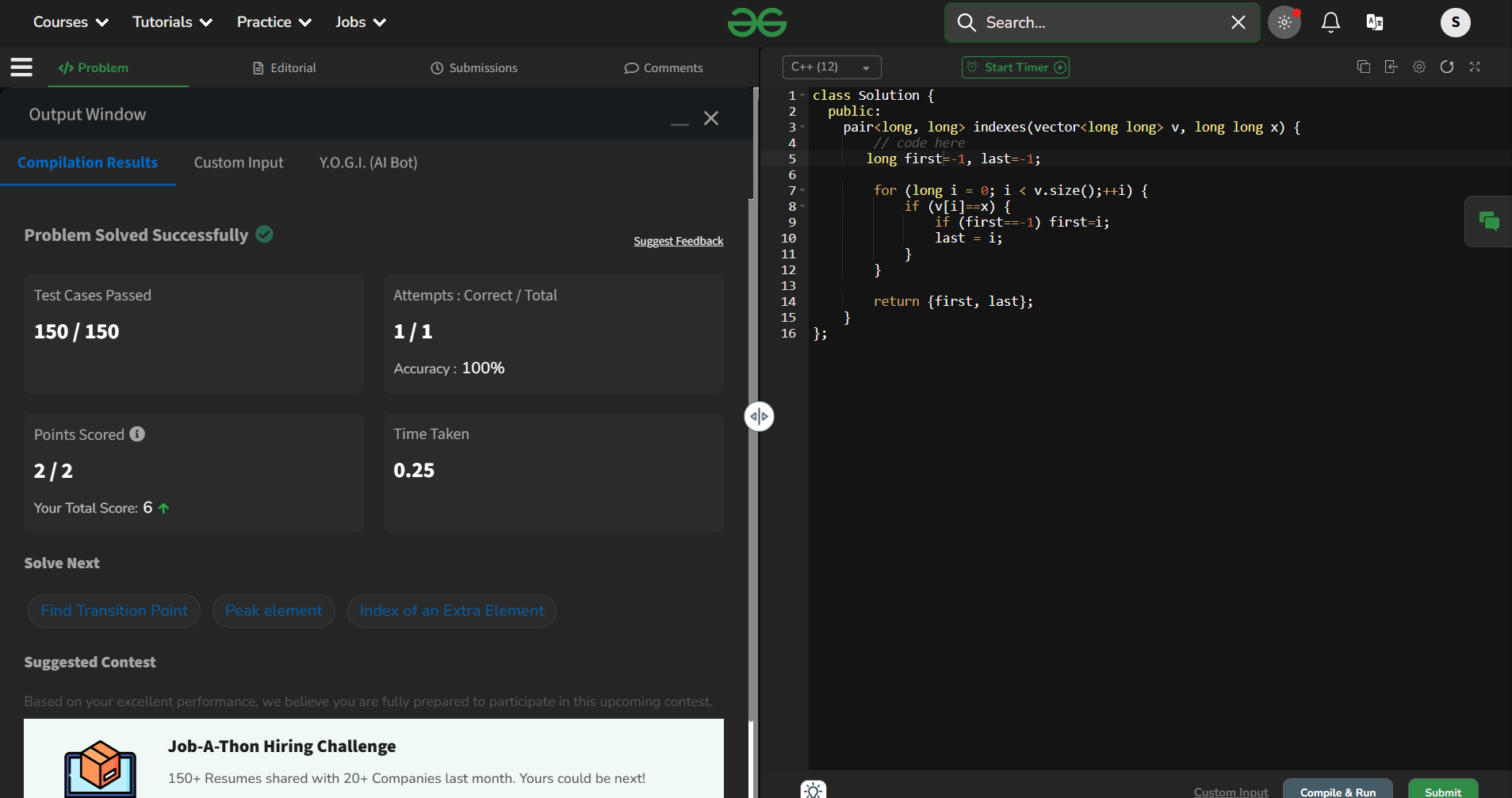
}

}

return {first, last};

}

};



Problem-4:

Code:

/\*

// Tree Node

class Node {

public:

int data;

Node\* left;

Node\* right;

// Constructor to initialize a new node

Node(int val) {

data = val;

left = NULL;

right = NULL;

}

};

\*/

class Solution {

public:

int getCount(Node \*root, int l, int h) {

if(root==nullptr) return 0;

if (root->data <=h && root->data>=l){

return 1 + getCount(root->left,l,h) + getCount(root->right,l,h);

}

else if (root->data<l){

return getCount(root->right,l,h);

}

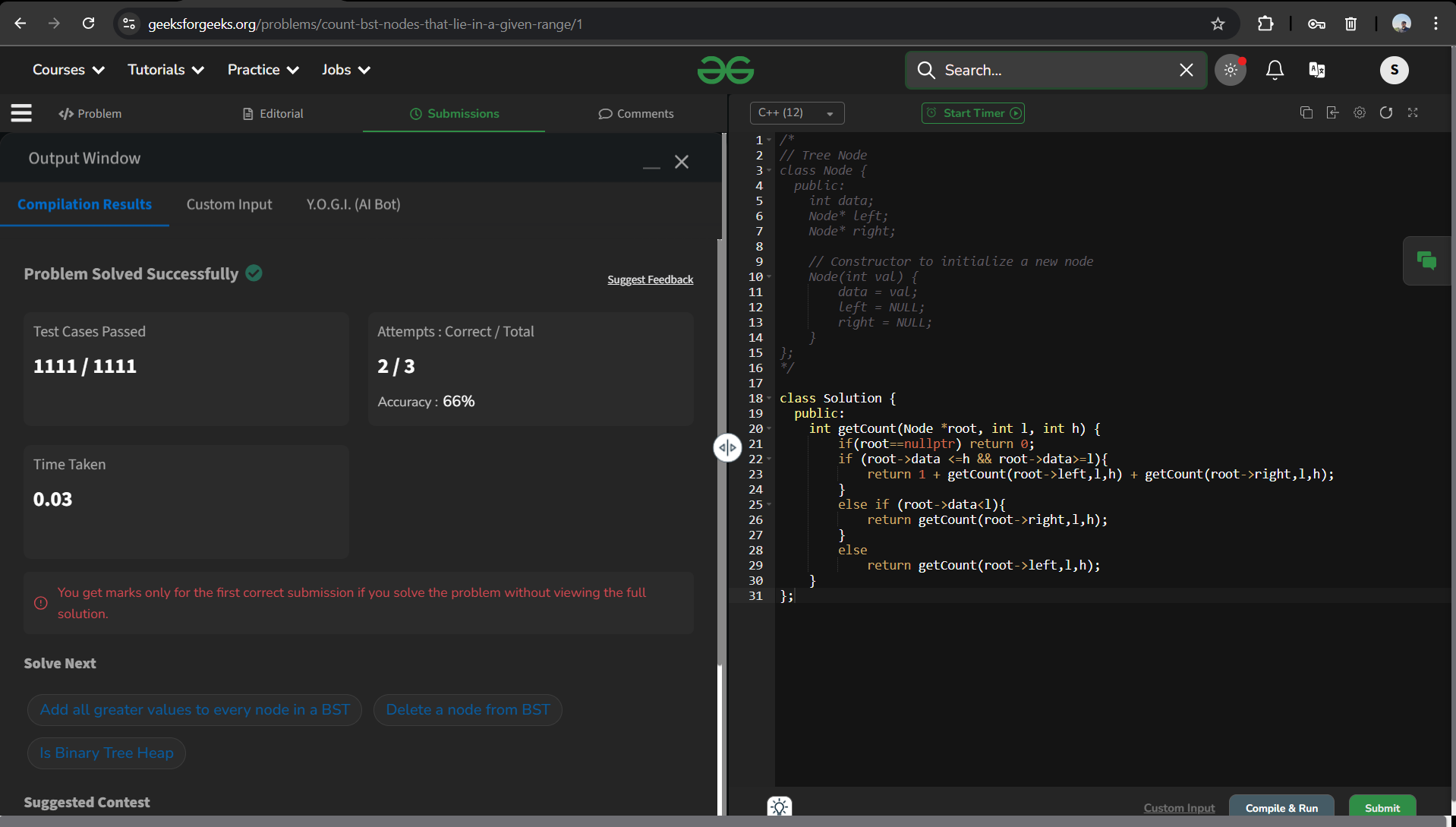
else

return getCount(root->left,l,h);

}

};

Output:



Problem-5: longest common prefix

Code:

// User function template for C++

class Solution {

public:

string longestCommonPrefix(vector<string> arr) {

// your code here

if(arr.size()==0) return 0;

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

char c=arr[0][i];

for(int j=0;j<arr.size();j++){

if(i=arr[i].length() || arr[[j][i]!=c]){

return arr[0].substr(0,i);

}

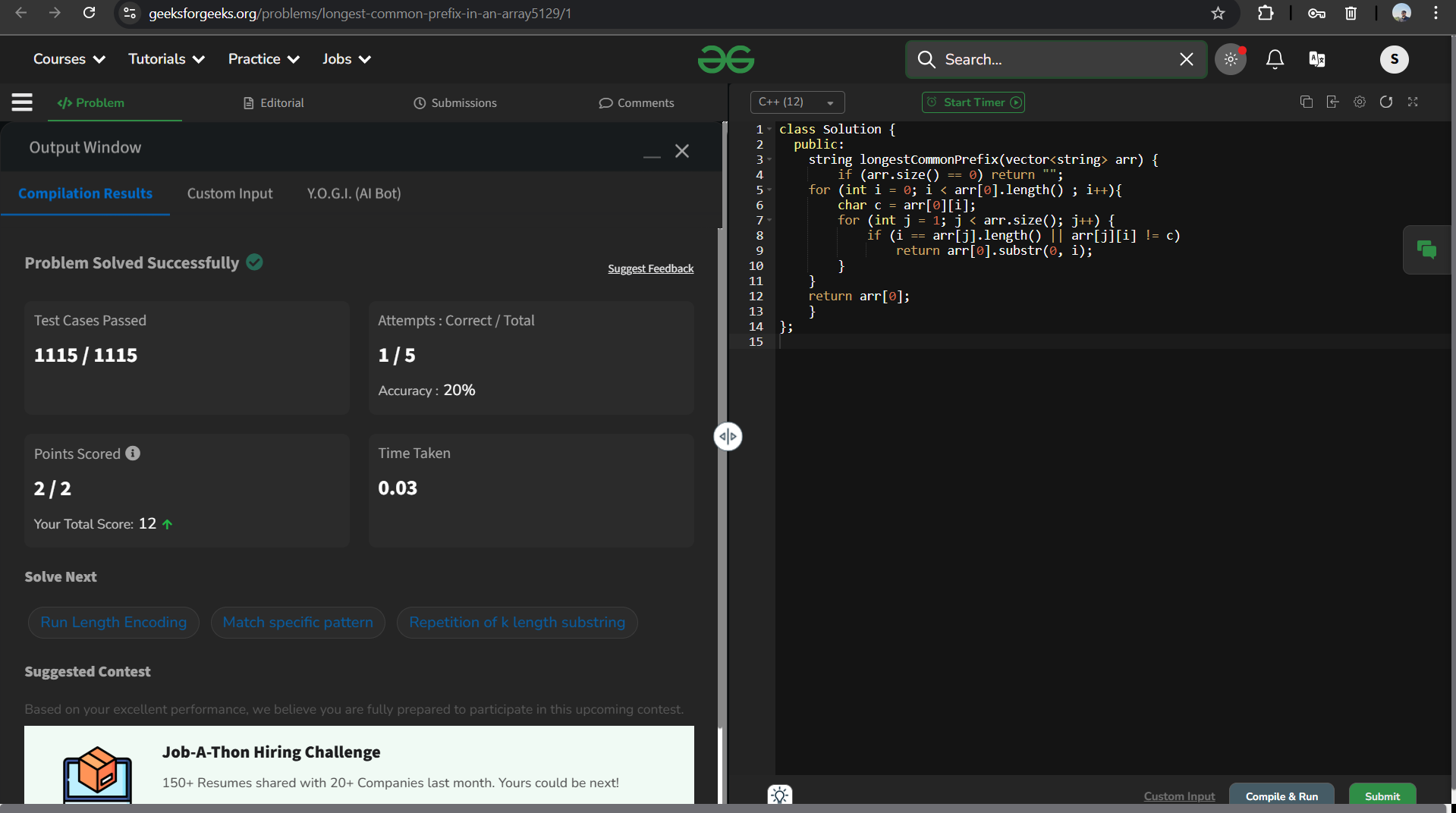
}

}

}

};

Output:



Problem-6:

Code:

class Solution {

public:

bool areKAnagrams(string &s1, string &s2, int k) {

// code here

{

if (s1.length() != s2.length()) {

return false;

}

unordered\_map<char, int> map;

for (int i = 0; i < s1.length(); i++) {

char ch = s1[i];

map[ch]++;

}

for (int i = 0; i < s2.length(); i++) {

char ch = s2[i];

if (map[ch] > 0) {

map[ch]--;

}

}

int count = 0;

for (auto it = map.begin(); it != map.end(); it++) {

count += it->second;

}

if (count > k)

return false;

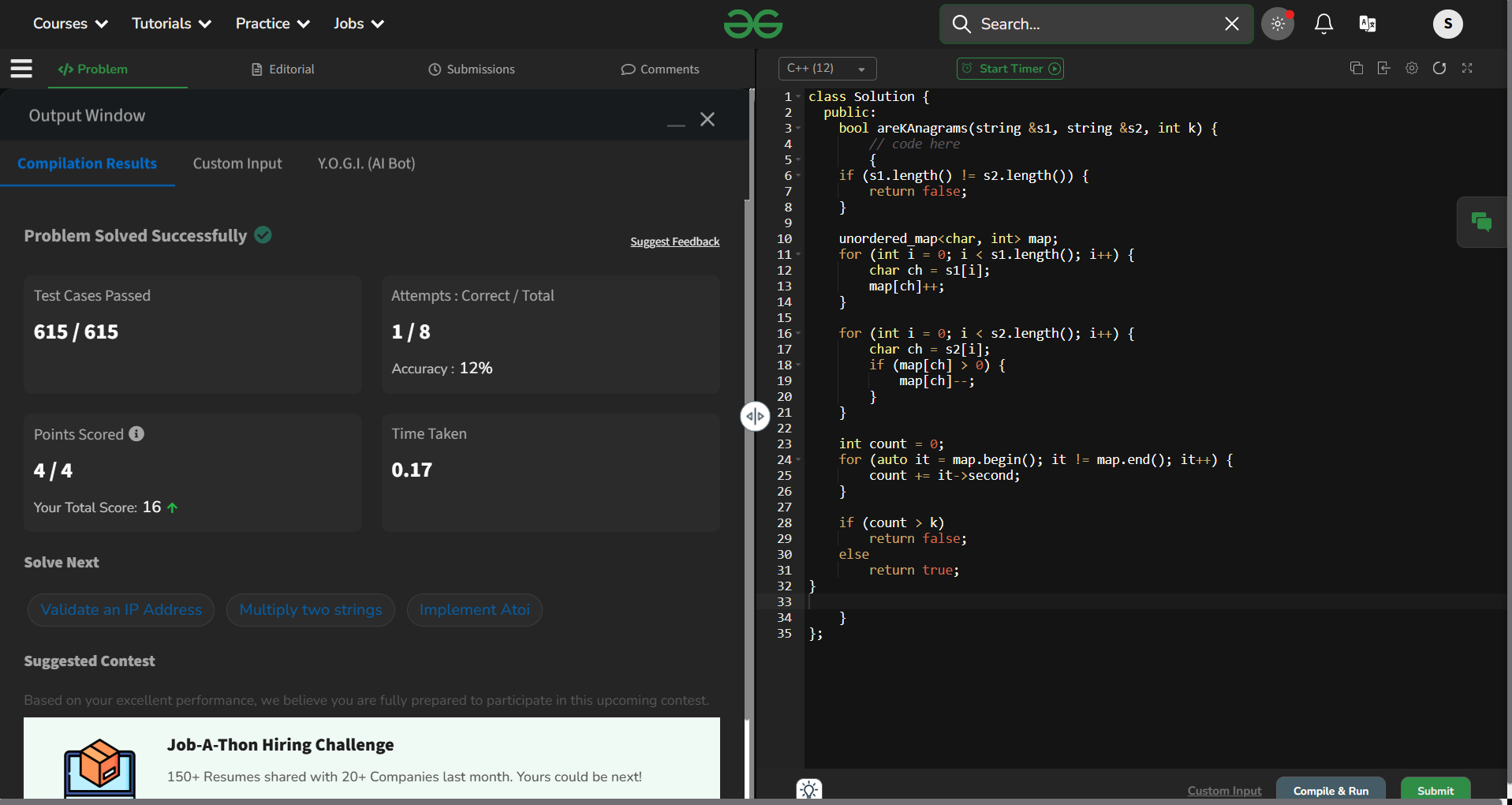
else

return true;

}

}

};



Problem-7: generate binary string

Code:

class Solution {

public:

void helper(string s, int index, vector<string>& result) {

if (index == s.length()) {

result.push\_back(s);

return;

}

if (s[index] == '?') {

s[index] = '0';

helper(s, index + 1, result);

s[index] = '1';

helper(s, index + 1, result);

s[index] = '?'; // backtrack

} else {

helper(s, index + 1, result);

}

}

vector<string> generate\_binary\_string(string s) {

vector<string> result;

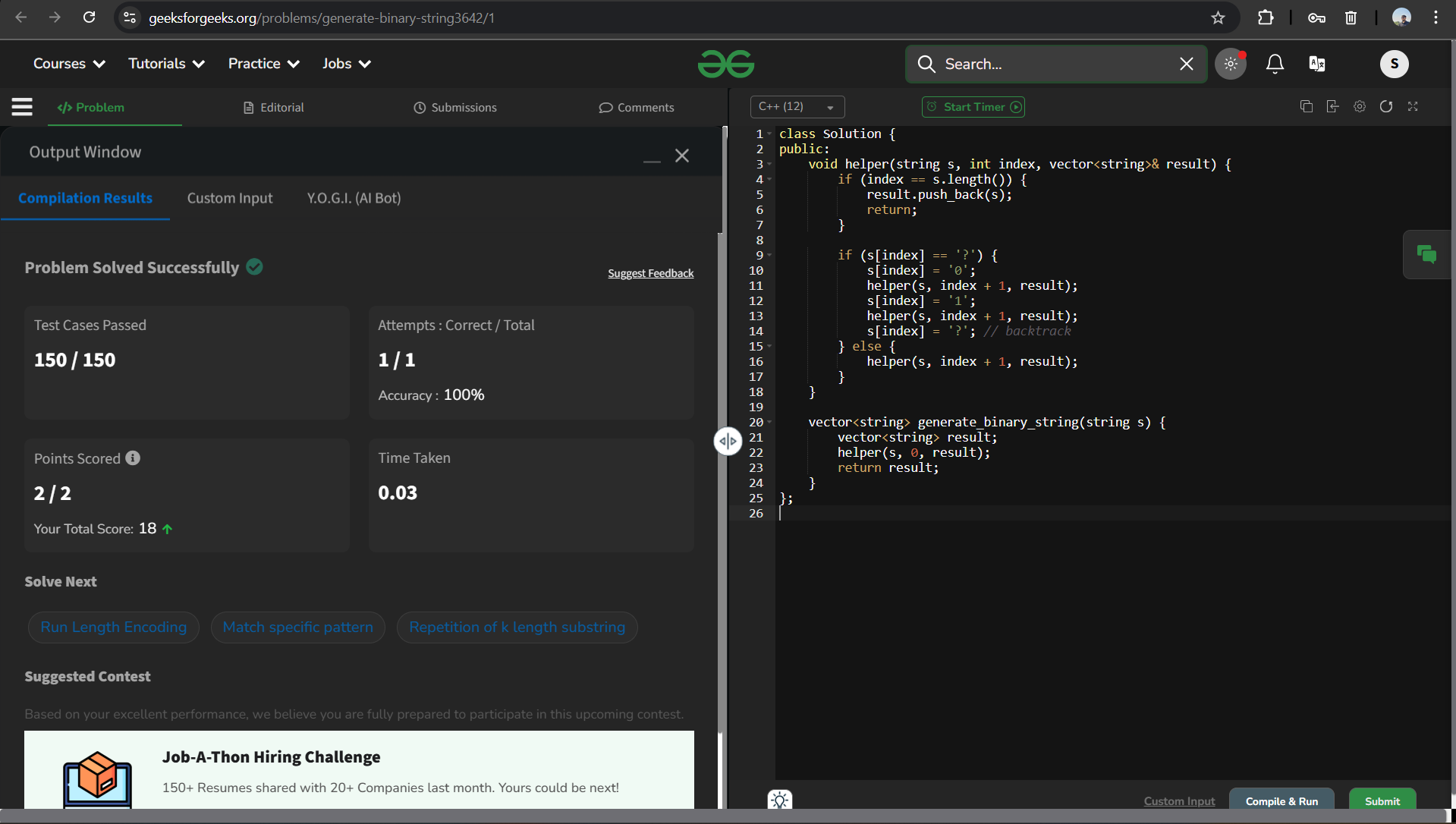
helper(s, 0, result);

return result;

}

};

Output:



Problem -8:

class Solution {

public:

vector<long long> numOfSubsets(int Arr[], int N) {

// code

int max=Arr[0],min=INT\_MAX,count\_min=0,count\_max=0;

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

if(Arr[i]>max){

max=Arr[i];

}

if(Arr[i]<min){

min=Arr[i];

}

}

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

if(Arr[i]==min){

count\_min++;

}

if(Arr[i]==max){

count\_max++;

}

}

long long subset1=pow(2, count\_min) - 1;

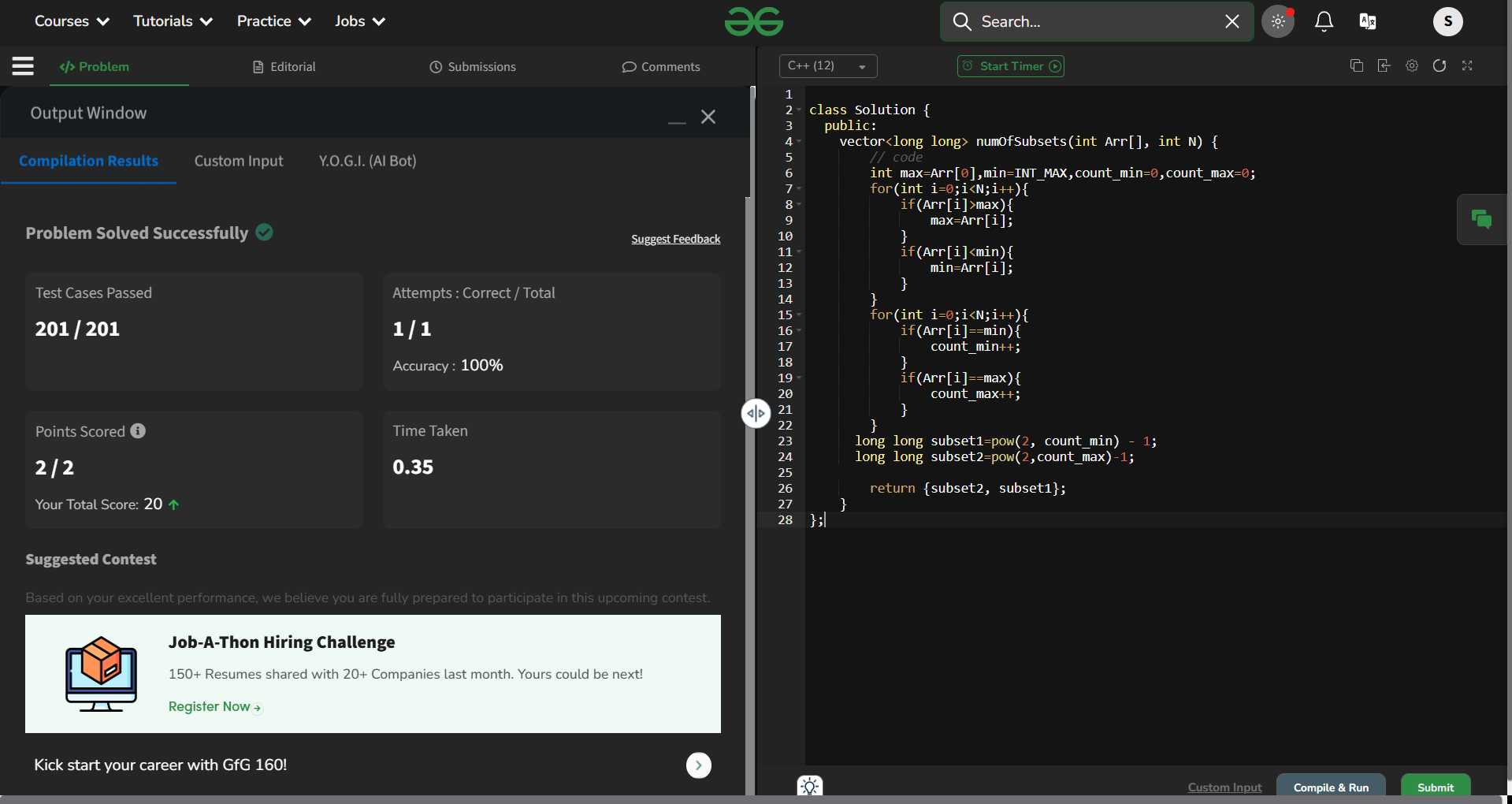
long long subset2=pow(2,count\_max)-1;

return {subset2, subset1};

}

};

Output:



Problem-9: subarray with given sum

Code:

class Solution {

public:

vector<int> subarraySum(vector<int> &arr, int target) {

int n=arr.size();

int j=0,sum=0;

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

sum+=arr[i];

while(sum>target){

sum-=arr[j++];

}

if(sum==target){

return {j+1,i+1};

}

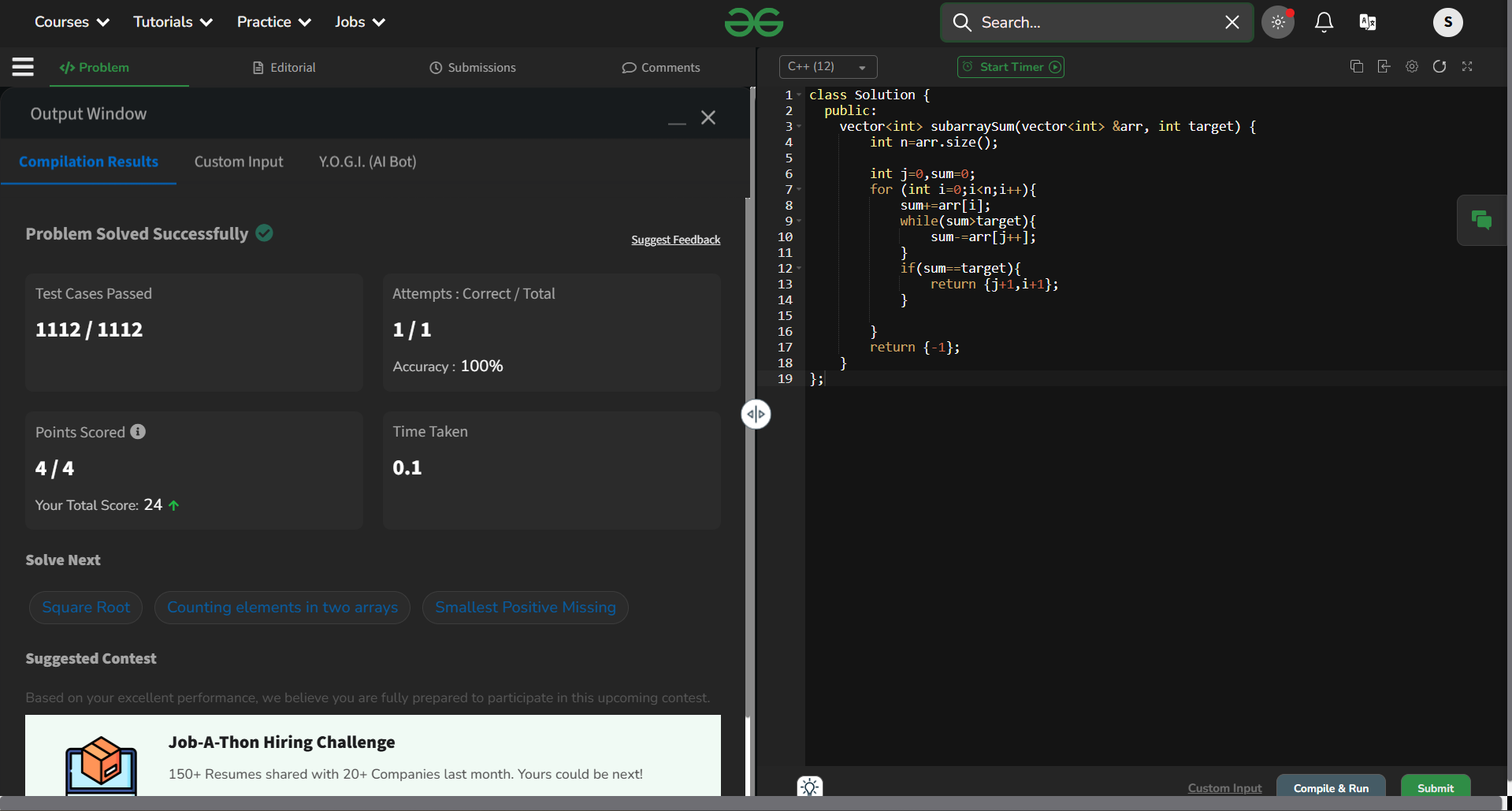
}

return {-1};

}

};

Output:



Problem-10: Maximum index

Code:

class Solution {

public:

int maxIndexDiff(vector<int>& arr) {

int n=arr.size();

vector<int> LMin(n), RMax(n);

LMin[0]=arr[0];

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

LMin[i]=min(arr[i], LMin[i-1]);

RMax[n-1]=arr[n-1];

for (int j=n-2;j>=0;j--)

RMax[j]=max(arr[j],RMax[j+1]);

int i= 0,j=0,maxDiff=-1;

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

if (LMin[i]<=RMax[j]) {

maxDiff=max(maxDiff,j-i);

j++;

} else {

i++;

}

}

return maxDiff;

}

};

Output:

