Kth Smallest Array

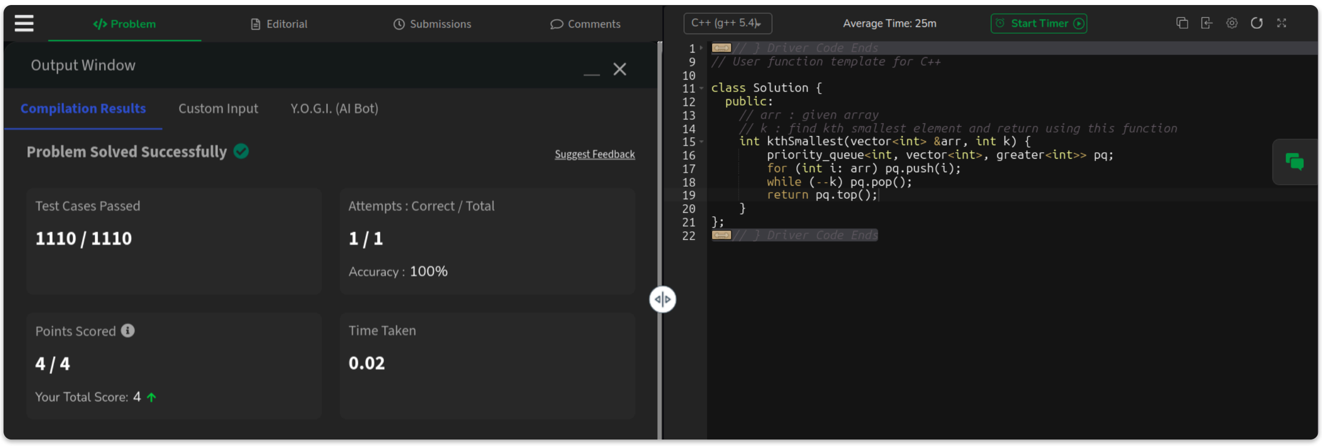
```
#include<bits/stdc++.h>
using namespace std;

#define FASTIO ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0)

void solve(vector<int> &arr, int k);

int main(){
    FASTIO;
    int n,k; cin>>n;
    vector<int> arr (n,0);
    for (int i=0; i<n; i++) cin>>arr[i];
    cin>>k;
    auto t1 = std::chrono::system_clock::now();
    solve(arr,k);
    auto t2 = std::chrono::system_clock::now();
    cout<<"-----------"<<endl;
    auto diff = t2-t1; cout<<"Time: "<<diff.count()/1e6<<" ms"<<endl;
    return 1;
}

void solve(vector<int> &arr, int k){
    priority_queue<int> pq;
    for (int i: arr) pq.push(i);
    while (--k) pq.pop();
    cout<<pq.top()<<endl;
}
```

Test case 1

```
6
7 10 4 3 20 15
3
```

Output

```
7
```



Problem Solved Successfully

Test Cases Passed
1110 / 1110

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
4 / 4
Your Total Score: 4

Time Taken
0.02

```
// } Driver Code Ends
// User function template for C++

class Solution {
  public:
    // arr : given array
    // k : find kth smallest element and return using this function
    int kthSmallest(vector<int> &arr, int k) {
        priority_queue<int, vector<int>, greater<int>> pq;
        for (int i: arr) pq.push(i);
        while (--k) pq.pop();
        return pq.top();
    }
};
// } Driver Code Ends
```

Time Complexity: $O(N * log(K))$
Space Complexity: $O(N)$ for auxiliary min heap

Minimize Heights II

```
class Solution {
  public:
    int getMinDiff(vector<int> &arr, int k) {
        sort(arr.begin(), arr.end());
        int n=arr.size();
        int a=arr[0]+k, b=arr[n-1]-k, mi=0, mx=0;
        int res = arr[n-1]-arr[0];
        for (int i=0; i<n-1; i++){
            mi=min(a,arr[i+1]-k); mx=max(b,arr[i]+k);
            if (mi<0) continue;
            res = min(res,mx-mi);
        }
        return res;
    }
};
```

Test case 1

```
3
3 9 12 16 20
```

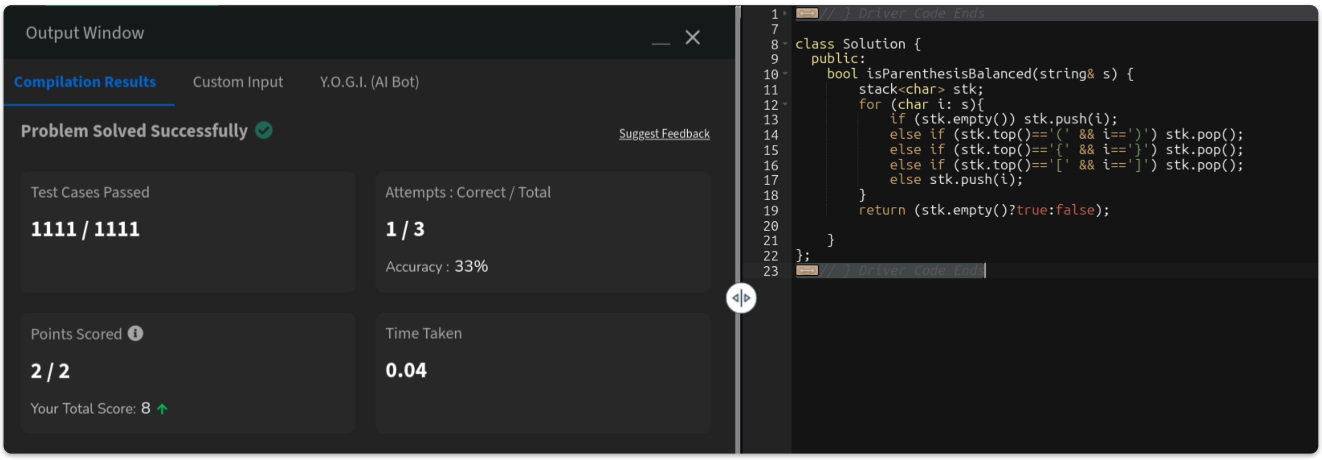
Output

```
11
```

Time Complexity: $O(N * log(N))$
Space Complexity: $O(1)$

Parenthesis Checker

```cpp
class Solution {
  public:
    bool isParenthesisBalanced(string& s) {
        stack<char> stk;
        for (char i: s){
            if (stk.empty()) stk.push(i);
            else if (stk.top()=='(' && i==')') stk.pop();
            else if (stk.top()=='{' && i=='}') stk.pop();
            else if (stk.top()=='[' && i==']') stk.pop();
            else stk.push(i);
        }
        return (stk.empty()?true:false);

    }
};
```



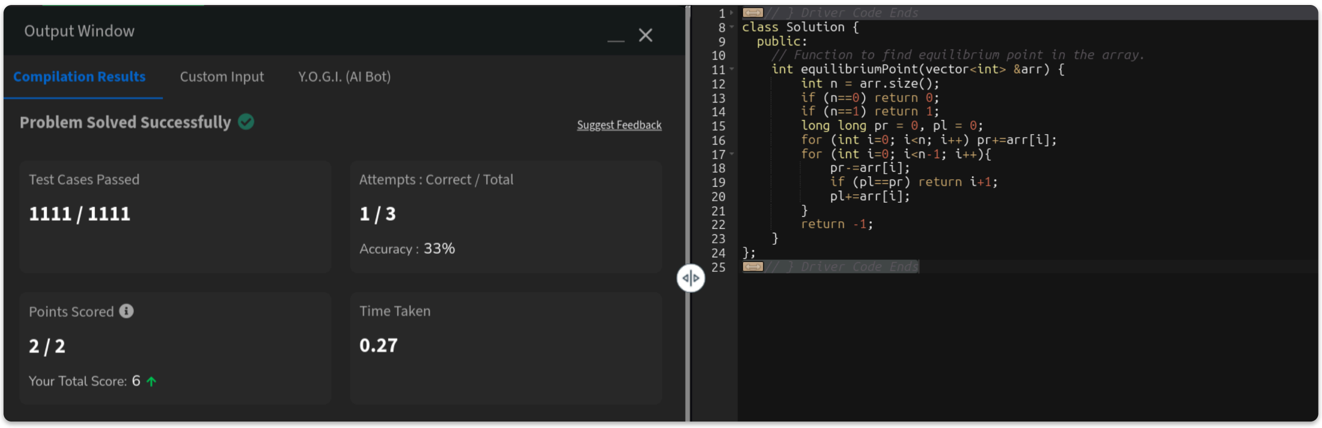
```
Output Window

Compilation Results    Custom Input    Y.O.G.I. (AI Bot)

Problem Solved Successfully                                Suggest Feedback

Test Cases Passed                  Attempts : Correct / Total
1111 / 1111                        1 / 3
                                   Accuracy : 33%

Points Scored                      Time Taken
2 / 2                              0.04
Your Total Score: 8
```

```cpp
// } Driver Code Ends

class Solution {
  public:
    bool isParenthesisBalanced(string& s) {
        stack<char> stk;
        for (char i: s){
            if (stk.empty()) stk.push(i);
            else if (stk.top()=='(' && i==')') stk.pop();
            else if (stk.top()=='{' && i=='}') stk.pop();
            else if (stk.top()=='[' && i==']') stk.pop();
            else stk.push(i);
        }
        return (stk.empty()?true:false);

    }
};
// } Driver Code Ends
```

Time Complexity: $O(N)$
Space Complexity: $O(N)$

Equilibrium Point

```cpp
class Solution {
  public:
    // Function to find equilibrium point in the array.
    int equilibriumPoint(vector<int> &arr) {
        int n = arr.size();
        if (n==0) return 0;
        if (n==1) return 1;
        long long pr = 0, pl = 0;
        for (int i=0; i<n; i++) pr+=arr[i];
        for (int i=0; i<n-1; i++){
            pr-=arr[i];
            if (pl==pr) return i+1;
            pl+=arr[i];
        }
        return -1;
    }
};
```



Output Window

Compilation Results · Custom Input · Y.O.G.I. (AI Bot)

Problem Solved Successfully · Suggest Feedback

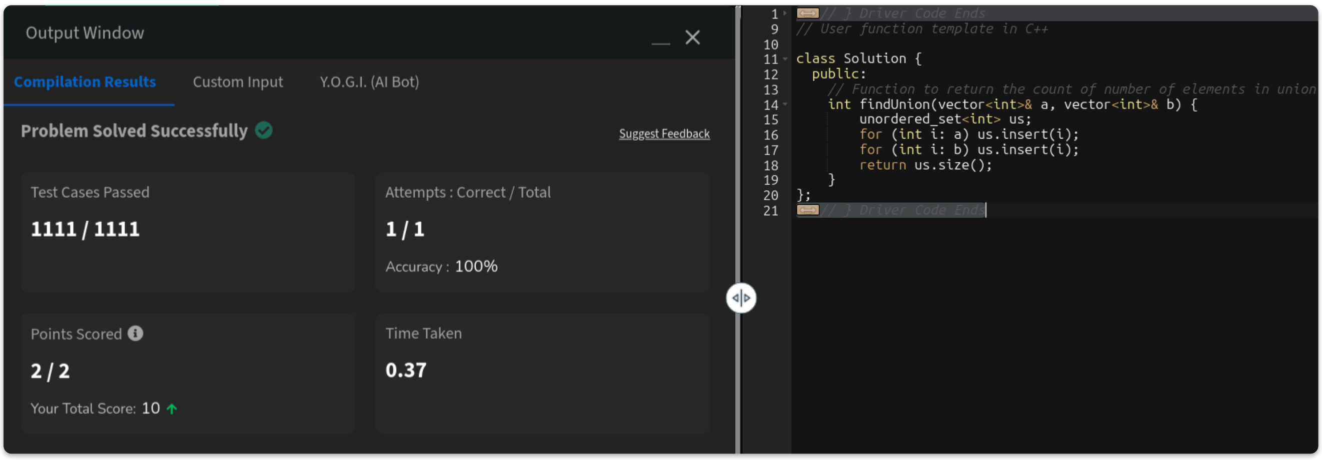
```
Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 3
Accuracy : 33%

Points Scored
2 / 2
Your Total Score: 6

Time Taken
0.27
```

```
// } Driver Code Ends
class Solution {
  public:
    // Function to find equilibrium point in the array.
    int equilibriumPoint(vector<int> &arr) {
        int n = arr.size();
        if (n==0) return 0;
        if (n==1) return 1;
        long long pr = 0, pl = 0;
        for (int i=0; i<n; i++) pr+=arr[i];
        for (int i=0; i<n-1; i++){
            pr-=arr[i];
            if (pl==pr) return i+1;
            pl+=arr[i];
        }
        return -1;
    }
};
// } Driver Code Ends
```

Time Complexity: $O(N)$
Space Complexity: $O(1)$

Union of two arrays with repeated elements

```
class Solution {
  public:
    // Function to return the count of number of elements in union of two
arrays.
    int findUnion(vector<int>& a, vector<int>& b) {
        unordered_set<int> us;
        for (int i: a) us.insert(i);
        for (int i: b) us.insert(i);
        return us.size();
    }
};
```



Output Window

Compilation Results · Custom Input · Y.O.G.I. (AI Bot)

Problem Solved Successfully · Suggest Feedback

```
Test Cases Passed
1111 / 1111

Attempts : Correct / Total
1 / 1
Accuracy : 100%

Points Scored
2 / 2
Your Total Score: 10

Time Taken
0.37
```

```
// } Driver Code Ends
// User function template in C++

class Solution {
  public:
    // Function to return the count of number of elements in union
    int findUnion(vector<int>& a, vector<int>& b) {
        unordered_set<int> us;
        for (int i: a) us.insert(i);
        for (int i: b) us.insert(i);
        return us.size();
    }
};
// } Driver Code Ends
```

Time Complexity: $O(m + n)$
Space Complexity: $O(m + n)$