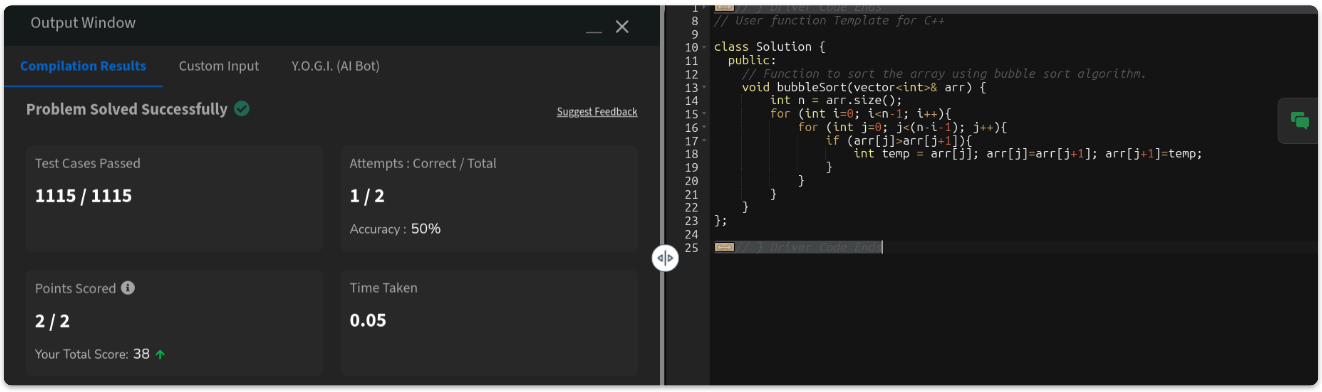
Bubble Sort

```
class Solution {
  public:
    // Function to sort the array using bubble sort algorithm.
    void bubbleSort(vector<int>& arr) {
        int n = arr.size();
        for (int i=0; i<n-1; i++){
            for (int j=0; j<(n-i-1); j++){
                if (arr[j]>arr[j+1]){
                    int temp = arr[j]; arr[j]=arr[j+1]; arr[j+1]=temp;
                }
            }
        }
    }
};
```

Output:



Output Window

Problem Solved Successfully

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 2
Accuracy : 50%

Points Scored
2 / 2
Your Total Score: 38

Time Taken
0.05

```
// User function Template for C++

class Solution {
  public:
    // Function to sort the array using bubble sort algorithm.
    void bubbleSort(vector<int>& arr) {
        int n = arr.size();
        for (int i=0; i<n-1; i++){
            for (int j=0; j<(n-i-1); j++){
                if (arr[j]>arr[j+1]){
                    int temp = arr[j]; arr[j]=arr[j+1]; arr[j+1]=temp;
                }
            }
        }
    }
};

// } Driver Code Ends
```

Time Complexity: $O(N^2)$
Space Complexity: $O(1)$

Quick Sort

Output

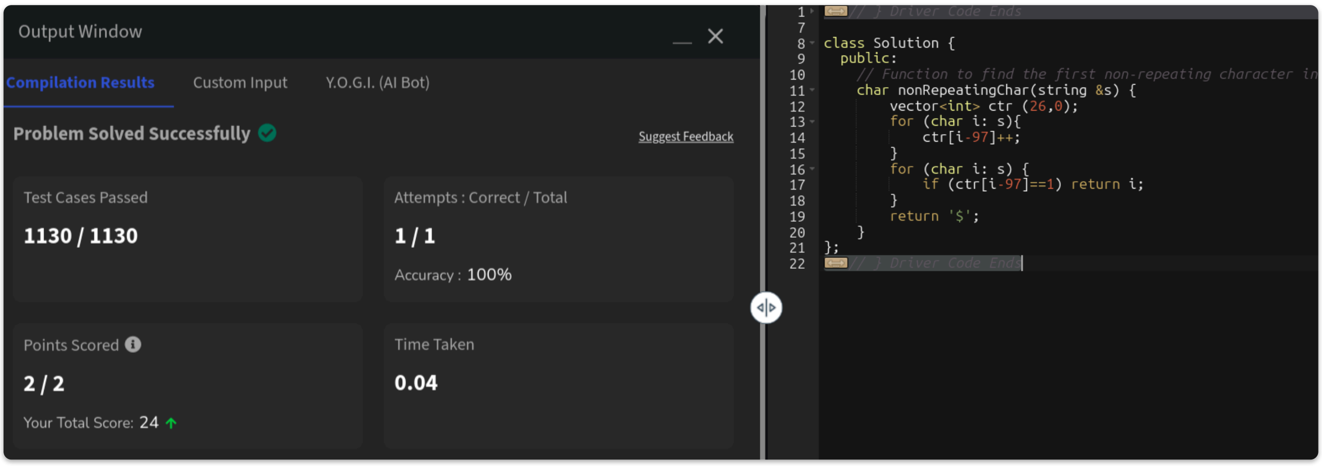
Time Complexity: $O(N*log(N))$ (Worst case $O(N^2)$ when in non increasing order)
Space Complexity: $O(1)$

First non repeating character

```
class Solution {
  public:
```

```cpp
    // Function to find the first non-repeating character in a string.
    char nonRepeatingChar(string &s) {
        vector<int> ctr (26,0);
        for (char i: s){
            ctr[i-97]++;
        }
        for (char i: s) {
            if (ctr[i-97]==1) return i;
        }
        return '$';
    }
};
```

Output



Output Window

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

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1130 / 1130

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

Points Scored
2 / 2
Your Total Score: 24

Time Taken
0.04

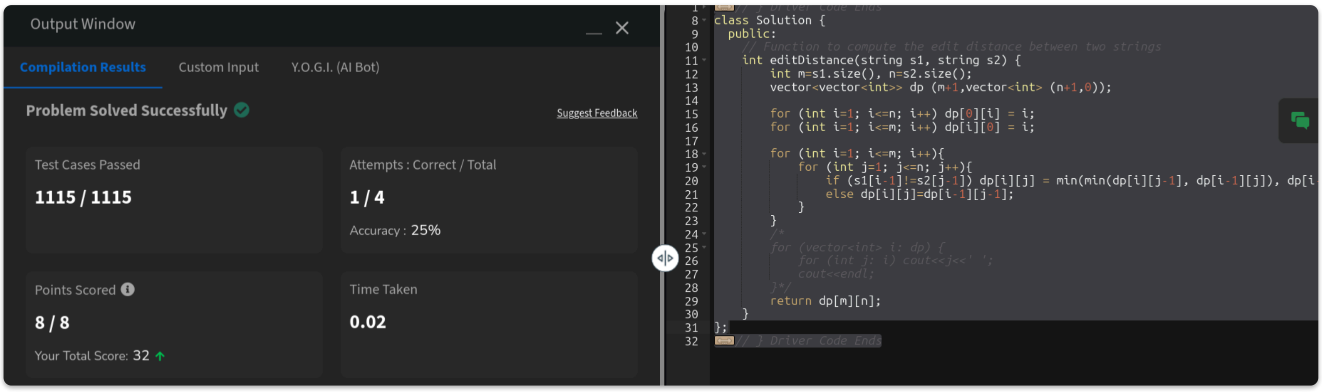
```cpp
// } Driver Code Ends

class Solution {
  public:
    // Function to find the first non-repeating character in
    char nonRepeatingChar(string &s) {
        vector<int> ctr (26,0);
        for (char i: s){
            ctr[i-97]++;
        }
        for (char i: s) {
            if (ctr[i-97]==1) return i;
        }
        return '$';
    }
};
// } Driver Code Ends
```

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

Edit Distance

```cpp
class Solution {
  public:
    // Function to compute the edit distance between two strings
    int editDistance(string s1, string s2) {
        int m=s1.size(), n=s2.size();
        vector<vector<int>> dp (m+1,vector<int> (n+1,0));

        for (int i=1; i<=n; i++) dp[0][i] = i;
        for (int i=1; i<=m; i++) dp[i][0] = i;

        for (int i=1; i<=m; i++){
            for (int j=1; j<=n; j++){
                if (s1[i-1]!=s2[j-1]) dp[i][j] = min(min(dp[i][j-1], dp[i-
1][j]), dp[i-1][j-1])+1;
                else dp[i][j]=dp[i-1][j-1];
            }
        }
        /*
        for (vector<int> i: dp) {
            for (int j: i) cout<<j<<' ';
            cout<<endl;
        }*/
        return dp[m][n];
    }
};
```

Output



Output Window

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

Problem Solved Successfully

Suggest Feedback

Test Cases Passed
1115 / 1115

Attempts : Correct / Total
1 / 4
Accuracy : 25%

Points Scored
8 / 8
Your Total Score: 32

Time Taken
0.02

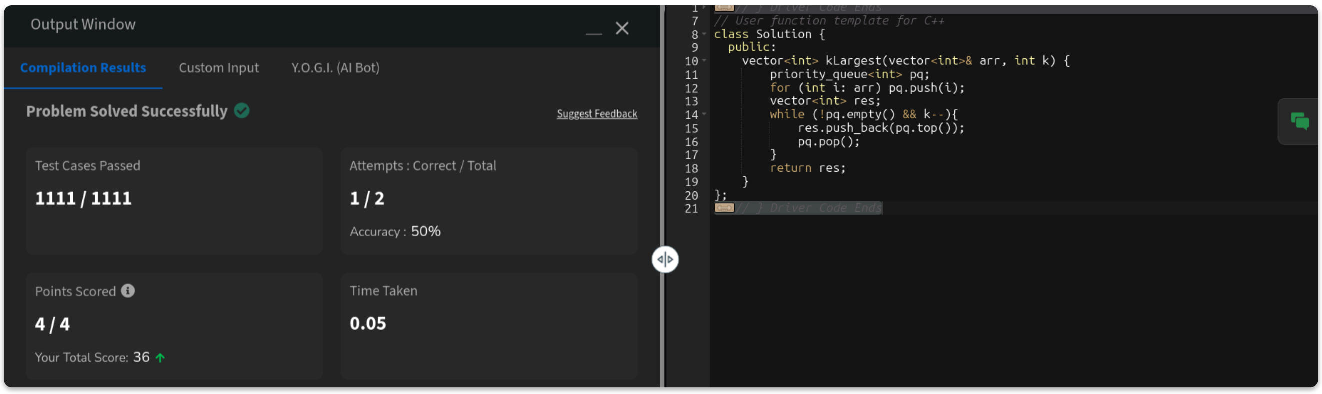
```
// } Driver Code Ends
class Solution {
  public:
    // Function to compute the edit distance between two strings
    int editDistance(string s1, string s2) {
        int m=s1.size(), n=s2.size();
        vector<vector<int>> dp (m+1,vector<int> (n+1,0));

        for (int i=1; i<=n; i++) dp[0][i] = i;
        for (int i=1; i<=m; i++) dp[i][0] = i;

        for (int i=1; i<=m; i++){
            for (int j=1; j<=n; j++){
                if (s1[i-1]!=s2[j-1]) dp[i][j] = min(min(dp[i][j-1], dp[i-1][j]), dp[i
                else dp[i][j]=dp[i-1][j-1];
            }
        }
        /*
        for (vector<int> i: dp) {
            for (int j: i) cout<<j<<' ';
            cout<<endl;
        }*/
        return dp[m][n];
    }
};
// } Driver Code Ends
```

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

K largest elements

```
class Solution {
  public:
    vector<int> kLargest(vector<int>& arr, int k) {
        priority_queue<int> pq;
        for (int i: arr) pq.push(i);
        vector<int> res;
        while (!pq.empty() && k--){
            res.push_back(pq.top());
            pq.pop();
        }
        return res;
    }
};
```

Output:


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 / 2
Accuracy : 50%

Points Scored
4 / 4
Your Total Score: 36

Time Taken
0.05

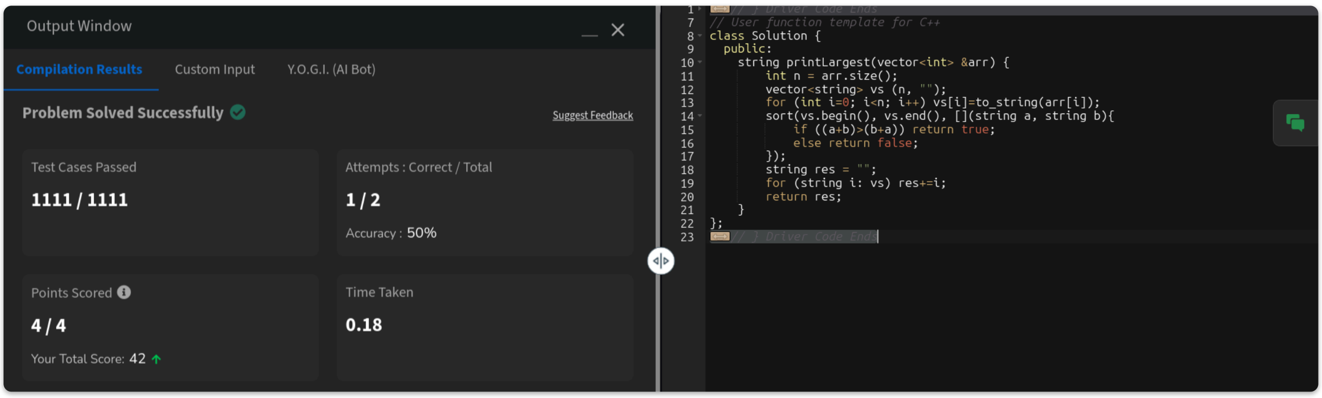
```
// } Driver Code Ends
// User function template for C++
class Solution {
  public:
    vector<int> kLargest(vector<int>& arr, int k) {
        priority_queue<int> pq;
        for (int i: arr) pq.push(i);
        vector<int> res;
        while (!pq.empty() && k--){
            res.push_back(pq.top());
            pq.pop();
        }
        return res;
    }
};
// } Driver Code Ends
```

Time Complexity: $O(k * log(n))$
Space Complexity: $O(n)$

Form the largest number

```
class Solution {
  public:
    string printLargest(vector<int> &arr) {
        int n = arr.size();
        vector<string> vs (n, "");
        for (int i=0; i<n; i++) vs[i]=to_string(arr[i]);
        sort(vs.begin(), vs.end(), [](string a, string b){
            if ((a+b)>(b+a)) return true;
            else return false;
        });
        string res = "";
        for (string i: vs) res+=i;
        return res;
    }
};
```

Output:


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 / 2
Accuracy : 50%

Points Scored
4 / 4
Your Total Score: 42

Time Taken
0.18

```
// } Driver Code Ends
// User function template for C++
class Solution {
  public:
    string printLargest(vector<int> &arr) {
        int n = arr.size();
        vector<string> vs (n, "");
        for (int i=0; i<n; i++) vs[i]=to_string(arr[i]);
        sort(vs.begin(), vs.end(), [](string a, string b){
            if ((a+b)>(b+a)) return true;
            else return false;
        });
        string res = "";
        for (string i: vs) res+=i;
        return res;
    }
};
// } Driver Code Ends
```

Time Complexity: $O(n * log(n))$
Space Complexity: $O(n)$