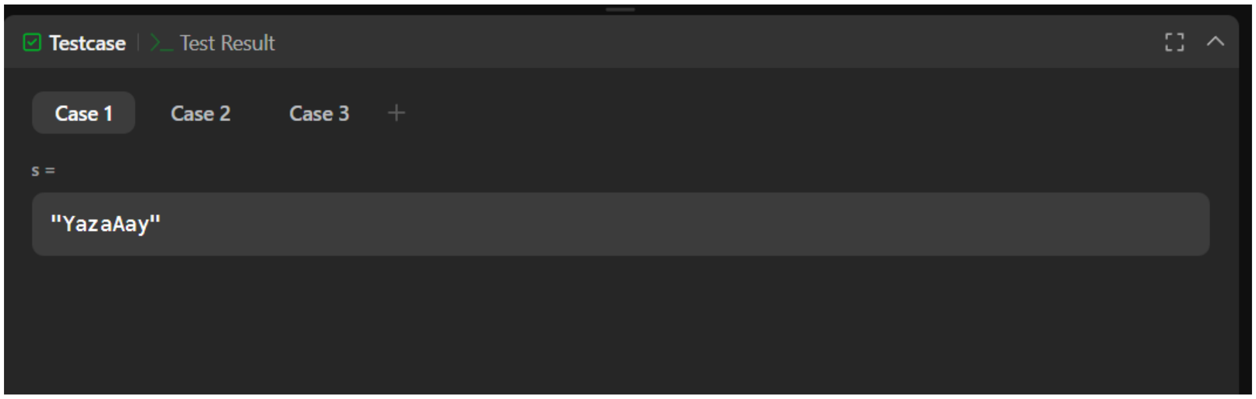
Name : Kumad Mahajan
UID : 22BCS13821

1. Longest Nice Substring:

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        int n=s.length();
        if (s.length()<2) {
            return "";
        }
        bool lower[26]={false};
        bool upper[26]={false};
        for(char c:s){
            if(islower(c)){
                lower[c-'a']=true;
            }
            else{
                upper[c-'A']=true;
            }
        }
        for(int i=0;i<n;i++){
            char c=s[i];
            if(islower(c)&&!upper[c-'a']){
                string left=longestNiceSubstring(s.substr(0,i));
                string right=longestNiceSubstring(s.substr(i+1));
                return left.length()>=right.length()?left:right;
            }
            if(isupper(c)&&!lower[c-'A']){
                string left=longestNiceSubstring(s.substr(0,i));
                string right=longestNiceSubstring(s.substr(i+1));
                return left.length()>=right.length()?left:right;
            }
        }
        return s;
    }
};
```

OUTPUT:



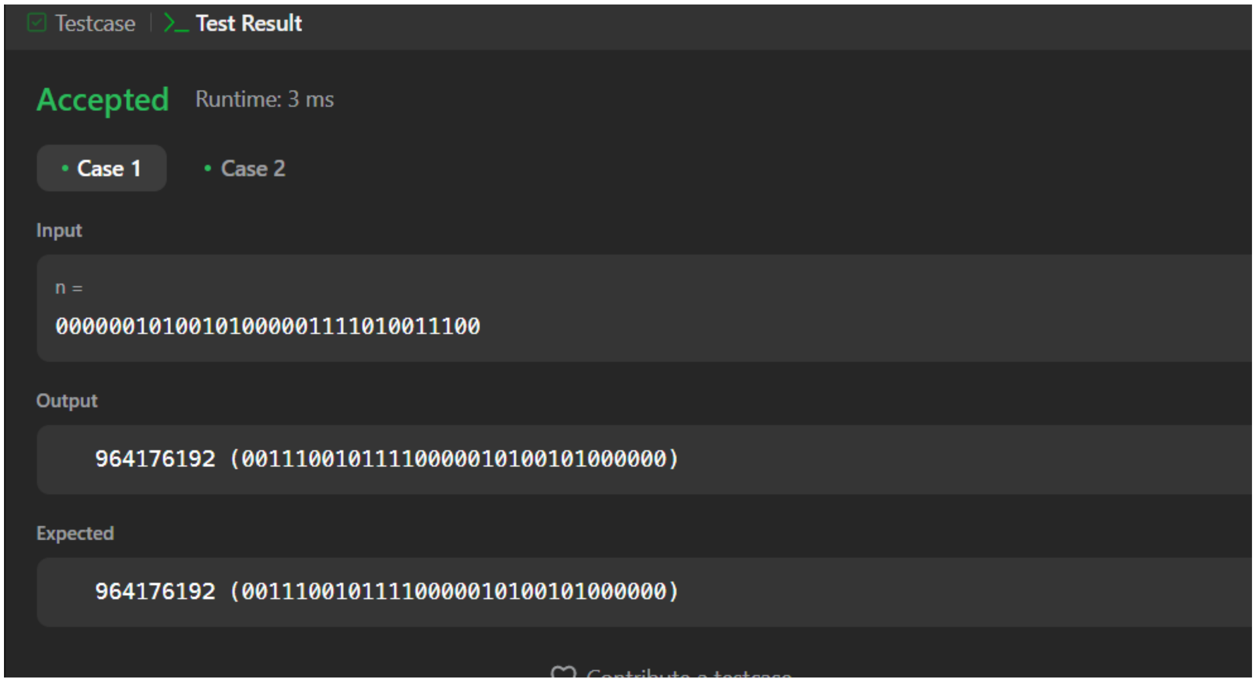
```
s =
"YazaAay"
```

2. Reverse Bits:

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;
        for (int i = 0; i < 32; i++) {
            result = (result << 1) | (n & 1);
            n >>= 1;
        }
        return result;
    }
};
```

Name : Kumad Mahajan
UID : 22BCS13821

OUTPUT:



Accepted Runtime: 3 ms

Input

```
n =
00000010100101000001111010011100
```

Output

```
964176192 (00111001011110000010100101000000)
```

Expected

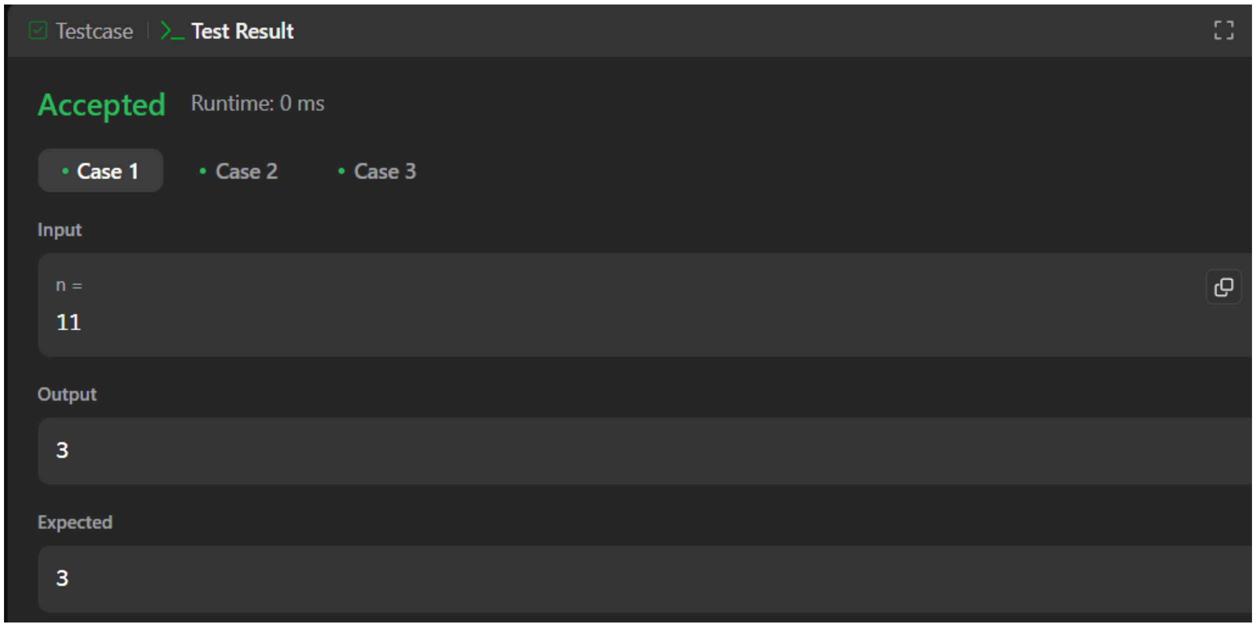
```
964176192 (00111001011110000010100101000000)
```

3. Number of 1 Bits:

```
class Solution {
public:
    int hammingWeight(int n) {
        int count = 0;
        while (n) {
            n &= (n - 1);
            count++;
        }
        return count;
    }
};
```

OUTPUT:

Name : Kumad Mahajan
UID : 22BCS13821


Testcase | >_ Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

```
n =
11
```

Output

```
3
```

Expected

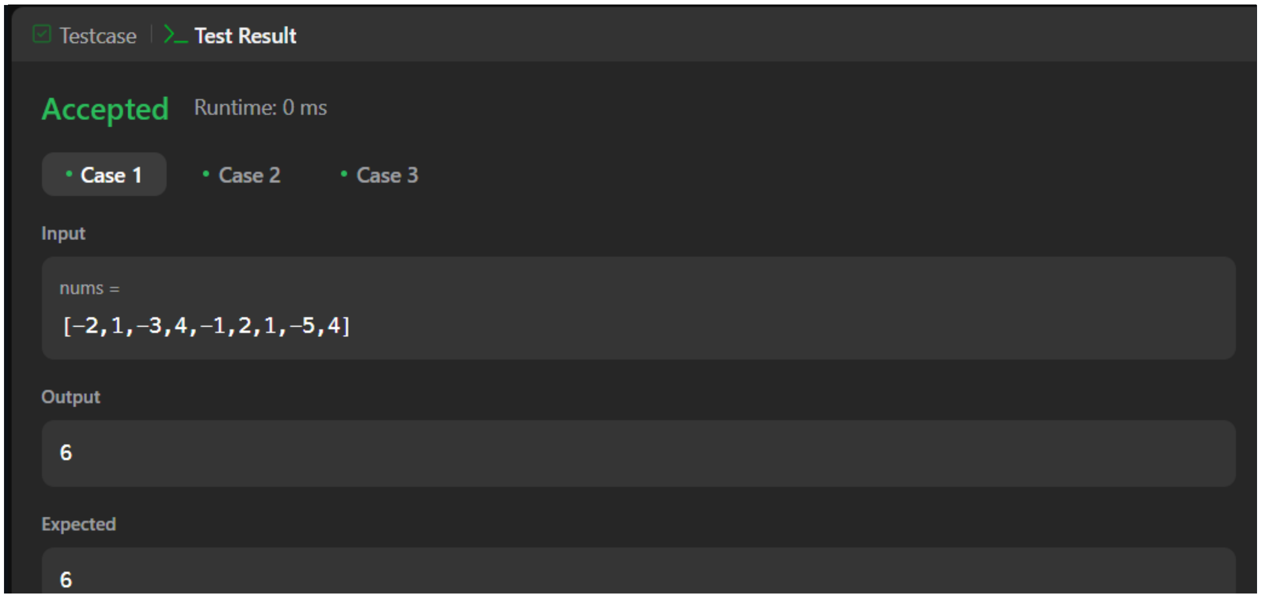
```
3
```

4.Maximum Subarray:

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum = nums[0];
        int currentSum = nums[0];

        for (int i = 1; i < nums.size(); i++) {
            currentSum = max(nums[i], currentSum + nums[i]);
            maxSum = max(maxSum, currentSum);
        }

        return maxSum;
    }
};
```

OUTPUT:

Name : Kumad Mahajan
UID : 22BCS13821



Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
nums =
[-2,1,-3,4,-1,2,1,-5,4]
```

Output

```
6
```

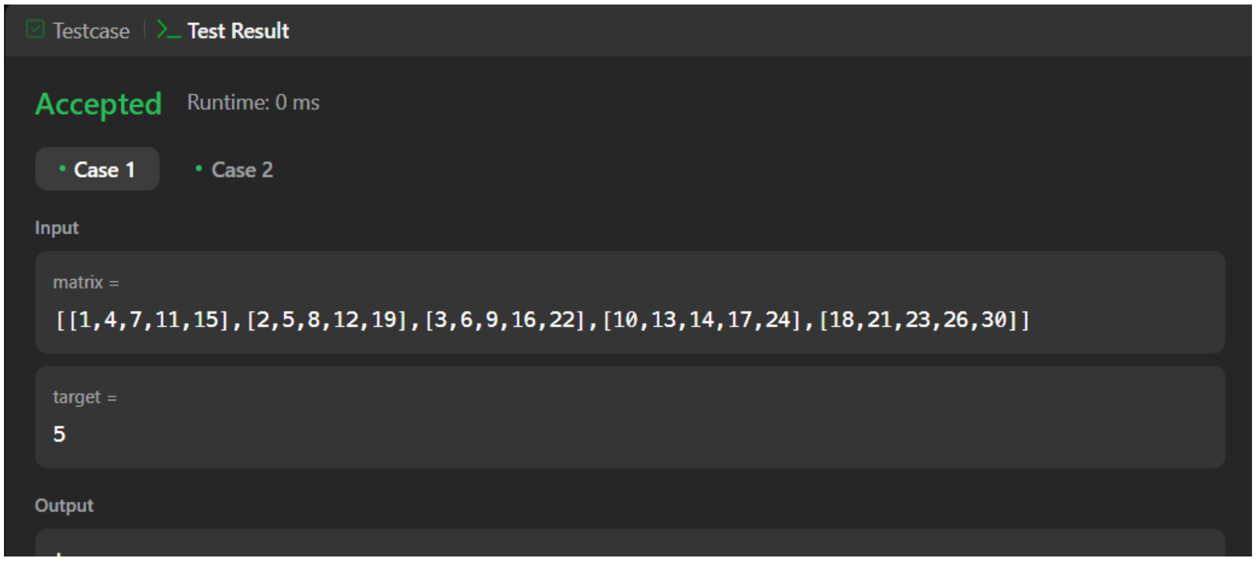
Expected

```
6
```

5.Search a 2D Matrix II:

```cpp
class Solution {
public:
    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        int m = matrix.size();
        int n = matrix[0].size();

        int row = 0, col = n - 1;

        while (row < m && col >= 0) {
            if (matrix[row][col] == target) return true;
            else if (matrix[row][col] > target) col--;
            else row++;
        }

        return false;
    }
};
```

OUTPUT:



Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

```
matrix =
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]
```

```
target =
5
```

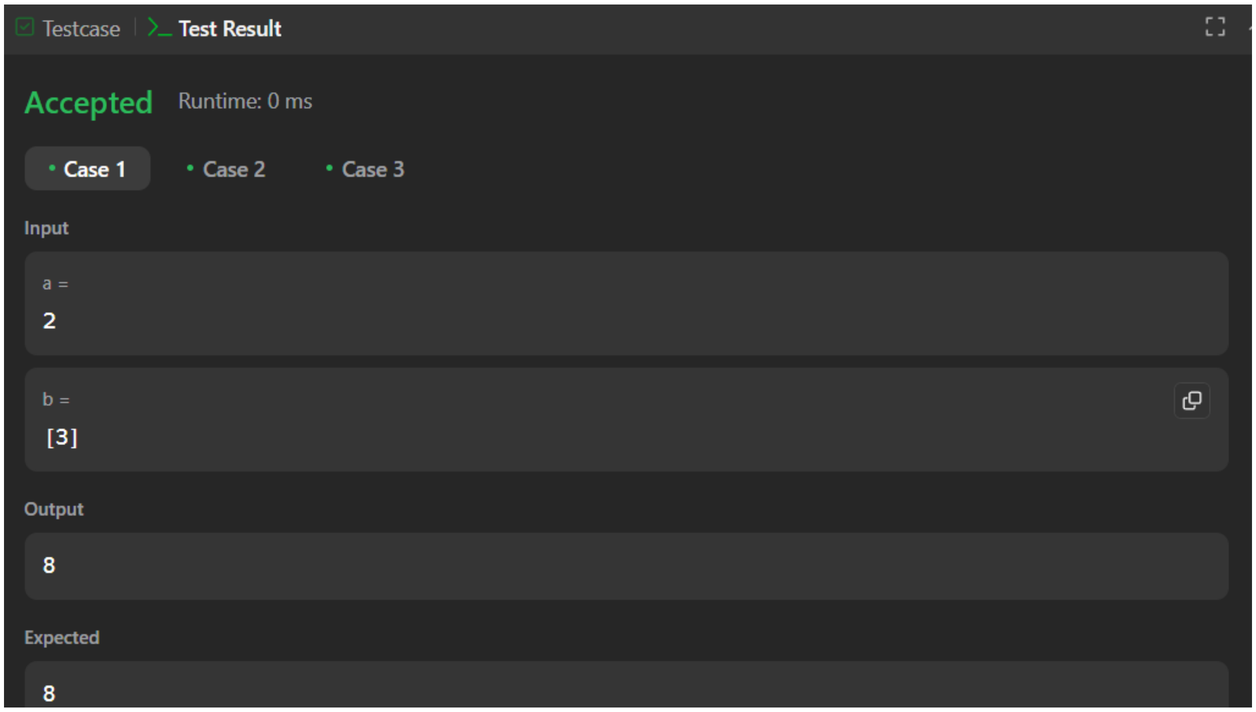
Output

Name : Kumad Mahajan
UID : 22BCS13821

6. Super Pow:

```
class Solution {
public:
    const int MOD = 1337;
    int modExp(int x, int y) {
        int result = 1;
        x %= MOD;
        while (y > 0) {
            if (y % 2 == 1) result = (result * x) % MOD;
            x = (x * x) % MOD;
            y /= 2;
        }
        return result;
    }
    int superPow(int a, vector<int>& b) {
        int result = 1;
        a %= MOD;
        for (int digit : b) {
            result = (modExp(result, 10) * modExp(a, digit)) % MOD;
        }
        return result;
    }
};
```

OUTPUT:



Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
a =
2
b =
[3]
```

Output

```
8
```

Expected

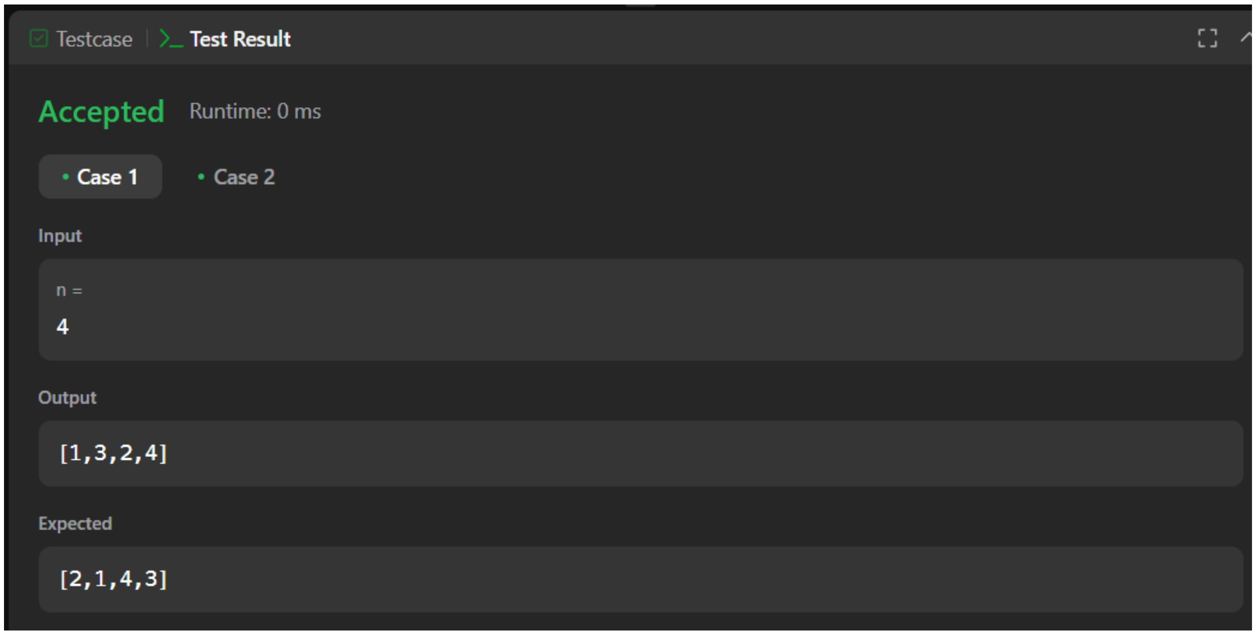
```
8
```

7.Beautiful Array:

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> result;
        if (n == 1) return {1};
        vector<int> left = beautifulArray((n + 1) / 2);
        vector<int> right = beautifulArray(n / 2);
        for (int x : left) result.push_back(2 * x - 1);
        for (int x : right) result.push_back(2 * x);
        return result;
    }
};
```

OUTPUT:

Name : Kumad Mahajan
UID : 22BCS13821


Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

n =
4

Output

```
[1,3,2,4]
```

Expected

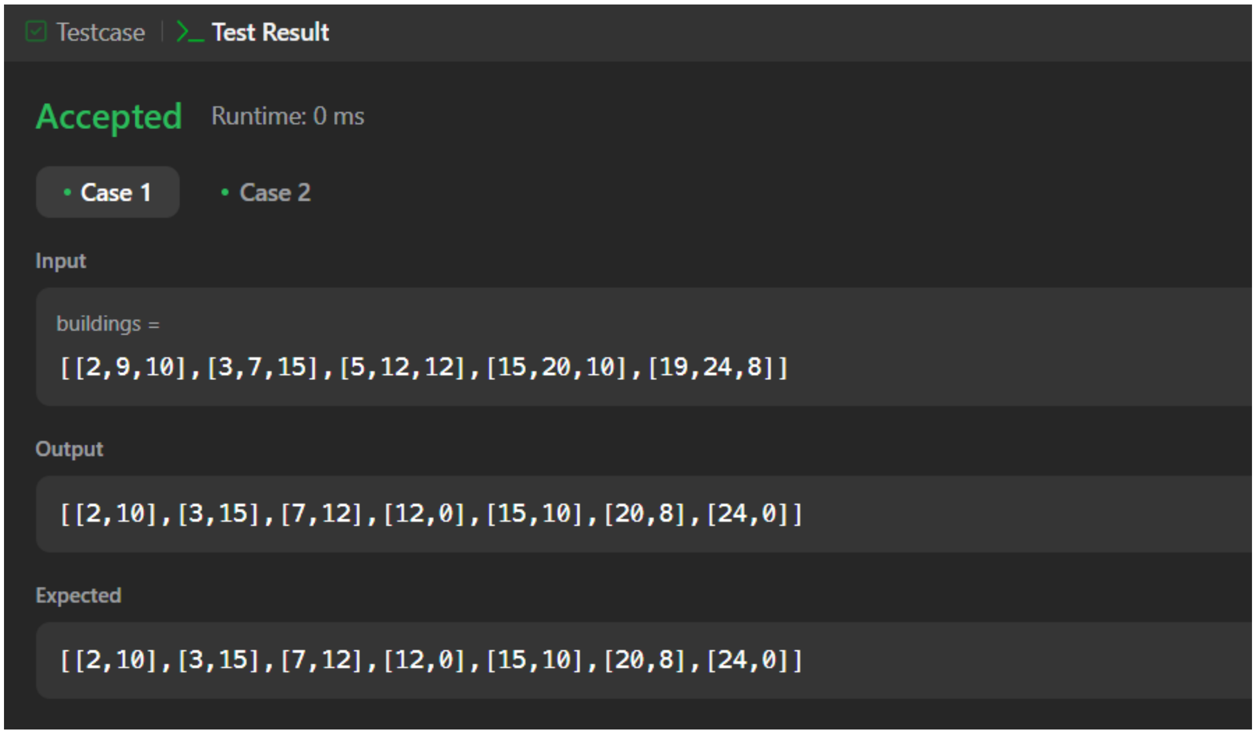
```
[2,1,4,3]
```

8. The Skyline Problem:

```
#include <vector>
#include <queue>
#include <set>
using namespace std;
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        vector<vector<int>> result;
        for (const auto& b : buildings) {
            events.emplace_back(b[0], -b[2]);
            events.emplace_back(b[1], b[2]);
        }

        sort(events.begin(), events.end(), [](const pair<int, int>& a, const pair<int, int>& b) {
            return a.first < b.first || (a.first == b.first && a.second < b.second);
        });
        multiset<int> heights = {0};
        int prevMax = 0;
        for (const auto& [x, h] : events) {
            if (h < 0) {
                heights.insert(-h);
            } else {
                heights.erase(heights.find(h));
            }
            int currMax = *heights.rbegin();
            if (currMax != prevMax) {
                result.push_back({x, currMax});
                prevMax = currMax;
            }
        }
        return result;
    }
};
```

OUTPUT:

Name : Kumad Mahajan
UID : 22BCS13821


Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

buildings =

```
[[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]
```

Output

```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```

Expected

```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```

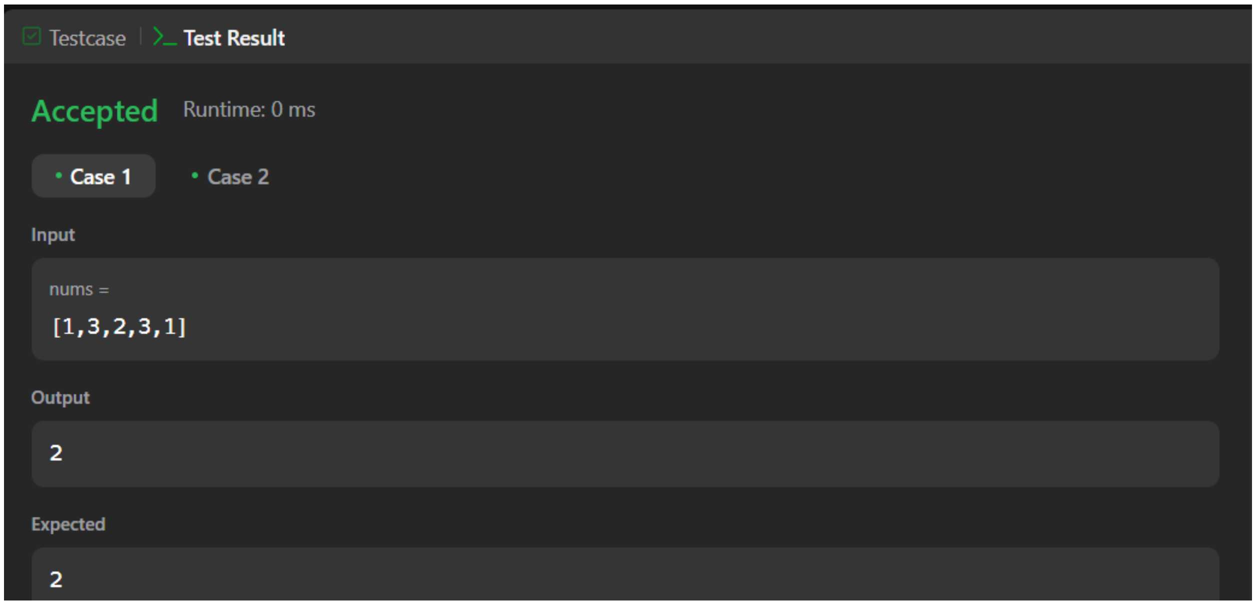
9. Reverse Pairs:

```
#include <vector>
using namespace std;
class Solution {
public:
    int mergeAndCount(vector<int>& nums, int left, int mid, int right) {
        int count = 0, j = mid + 1;
        for (int i = left; i <= mid; i++) {
            while (j <= right && nums[i] > 2LL * nums[j]) {
                j++;
            }
            count += (j - (mid + 1));
        }
        vector<int> temp;
        int i = left, k = mid + 1;
        while (i <= mid && k <= right) {
            if (nums[i] <= nums[k]) {
                temp.push_back(nums[i++]);
            } else {
                temp.push_back(nums[k++]);
            }
        }
        while (i <= mid) temp.push_back(nums[i++]);
        while (k <= right) temp.push_back(nums[k++]);
        for (int i = left; i <= right; i++) {
            nums[i] = temp[i - left];
        }
        return count;
    }
    int mergeSortAndCount(vector<int>& nums, int left, int right) {
        if (left >= right) return 0;
        int mid = left + (right - left) / 2;
        int count = mergeSortAndCount(nums, left, mid) + mergeSortAndCount(nums, mid + 1, right);
        count += mergeAndCount(nums, left, mid, right);
        return count;
    }
    int reversePairs(vector<int>& nums) {
        return mergeSortAndCount(nums, 0, nums.size() - 1);
    }
```

Name : Kumad Mahajan
UID : 22BCS13821

```
};
```

OUTPUT:


Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2

Input

```
nums =
[1,3,2,3,1]
```

Output

```
2
```

Expected

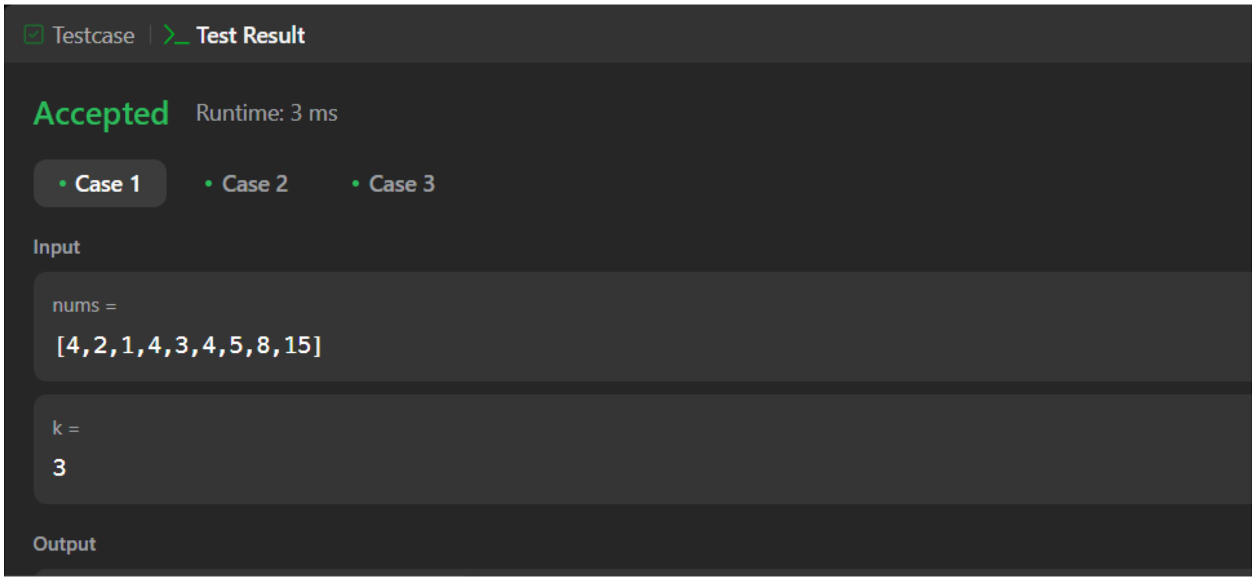
```
2
```

10. Longest Increasing Subsequence II:

```cpp
#include <vector>
#include <algorithm>
using namespace std;
class SegmentTree {
    vector<int> tree;
    int size;
public:
    SegmentTree(int n) {
        size = n;
        tree.assign(4 * n, 0);
    }
    void update(int index, int value, int node = 1, int left = 0, int right = 1e5) {
        if (left == right) {
            tree[node] = value;
            return;
        }
        int mid = (left + right) / 2;
        if (index <= mid)
            update(index, value, 2 * node, left, mid);
        else
            update(index, value, 2 * node + 1, mid + 1, right);

        tree[node] = max(tree[2 * node], tree[2 * node + 1]);
    }
    int query(int L, int R, int node = 1, int left = 0, int right = 1e5) {
        if (L > R) return 0;
        if (L == left && R == right) return tree[node];

        int mid = (left + right) / 2;
        return max(query(L, min(R, mid), 2 * node, left, mid),
                   query(max(L, mid + 1), R, 2 * node + 1, mid + 1, right));
    }
};
class Solution {
public:
    int lengthOfLIS(vector<int>& nums, int k) {
        SegmentTree seg(1e5 + 1);
        int maxLength = 1;

        for (int num : nums) {
            int bestPrev = seg.query(max(0, num - k), num - 1);
            int newLength = bestPrev + 1;
            seg.update(num, newLength);
            maxLength = max(maxLength, newLength);
        }
        return maxLength;
    }
};
```

Name : Kumad Mahajan
UID : 22BCS13821

OUTPUT:



Accepted Runtime: 3 ms

Case 1 Case 2 Case 3

Input

nums =

```
[4,2,1,4,3,4,5,8,15]
```

k =

```
3
```

Output