**EXPERIMENT - 5**

**Student Name: Aditya Kumar Bharti UID: 22BET10276**

**Branch: BE-IT Section/Group:22BET\_IOT-703(B )**

**Semester: 6th  Subject Code: 22ITP-351**

# PROBLEM-1

**AIM:-**

Merge SortedArray

## CODE:-

class Solution

{ public:

void merge(vector<int>& nums1,

int m, vector<int>& nums2, int n) { for (int j = 0, i = m; j<n; j++){ nums1[i] = nums2[j]; i++;

}

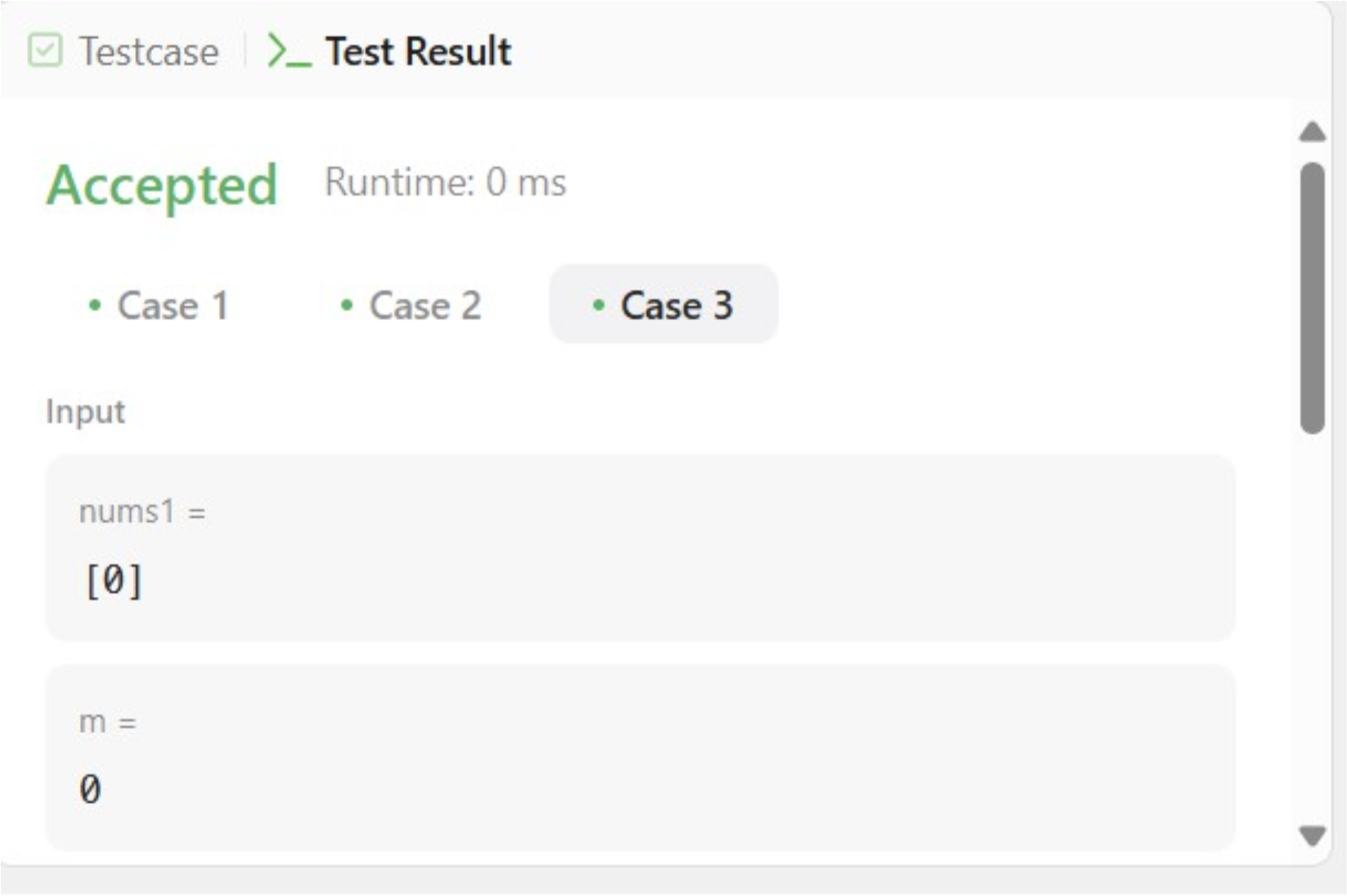
sort(nums1.begin(),nums1.end());

}

};



**OUTPUT:-**



**PROBLEM-2**

**AIM:-**

First Bad Version

## CODE:-

// The API isBadVersion is defined for you.

// bool isBadVersion(int version);

class Solution

{ public:

int firstBadVersion(int

n){ long long l = 1, r = n; long long m, res = n; while(l <= r){ m = l + (r - l) / 2;

if(isBadVersion(m) == 1){

r = m - 1;

res = min(res, m);

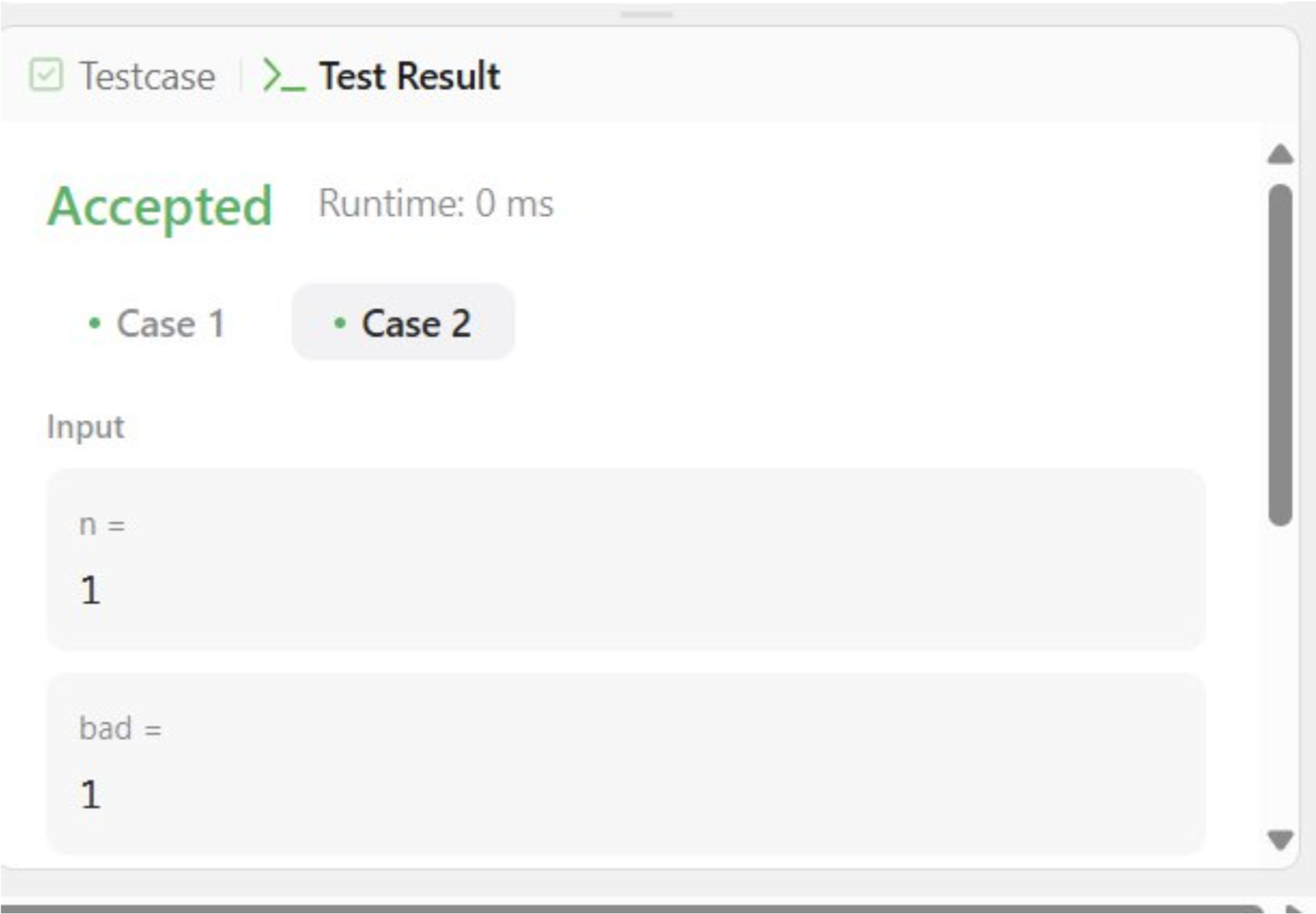
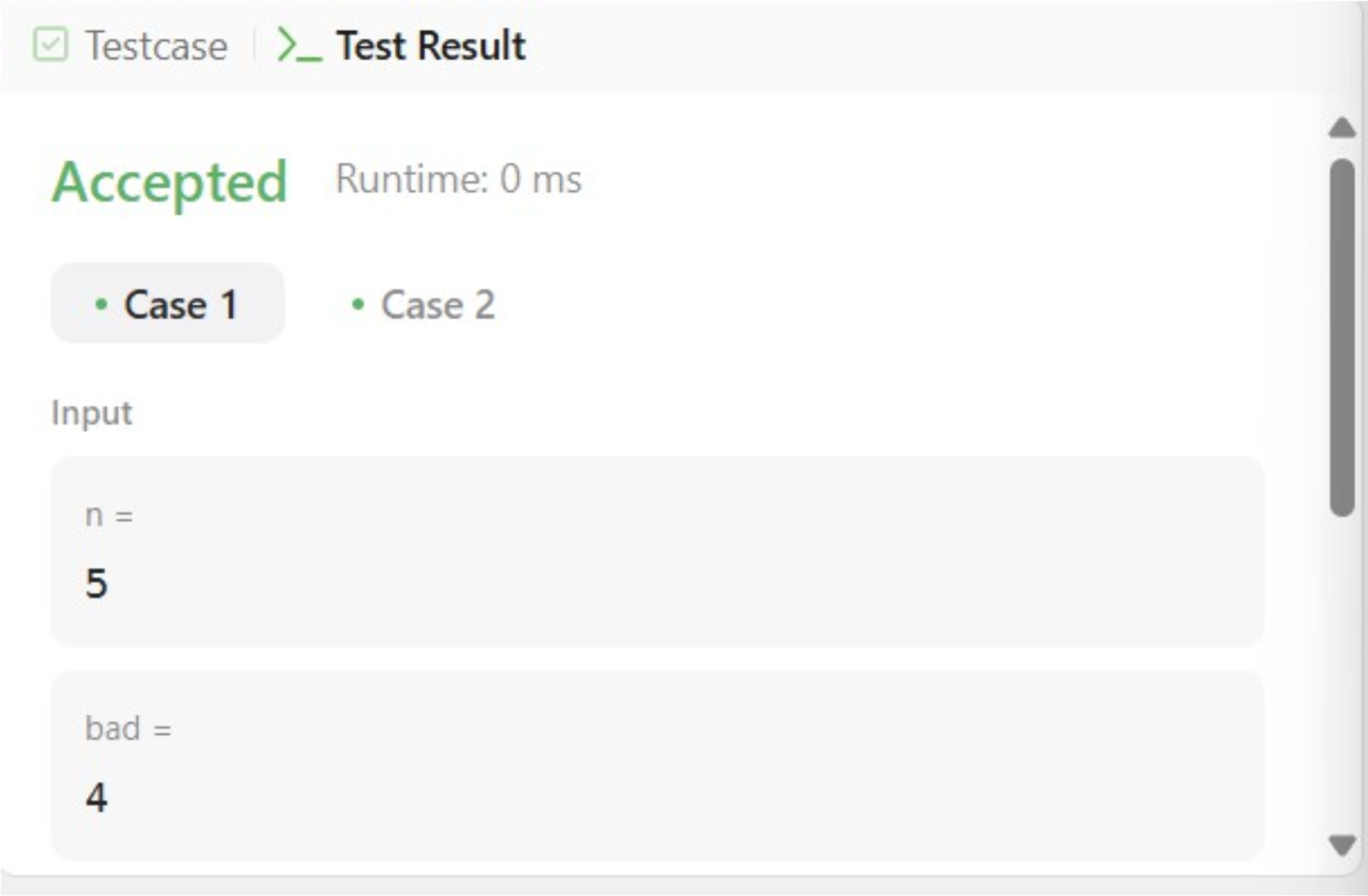
} else {

l = m + 1;

} } return res;

}

};



**OUTPUT:-**

# PROBLEM-3

**AIM:-**

Sort Colors

**CODE:**class Solution { public:

void sortColors(vector<int>& nums) {

int low = 0, mid = 0, high = nums.size()-1; while(mid <= high){

if(nums[mid] == 0){ swap(nums[low], nums[mid]); low++; mid++; } else if(nums[mid] == 1){ mid++; } else{

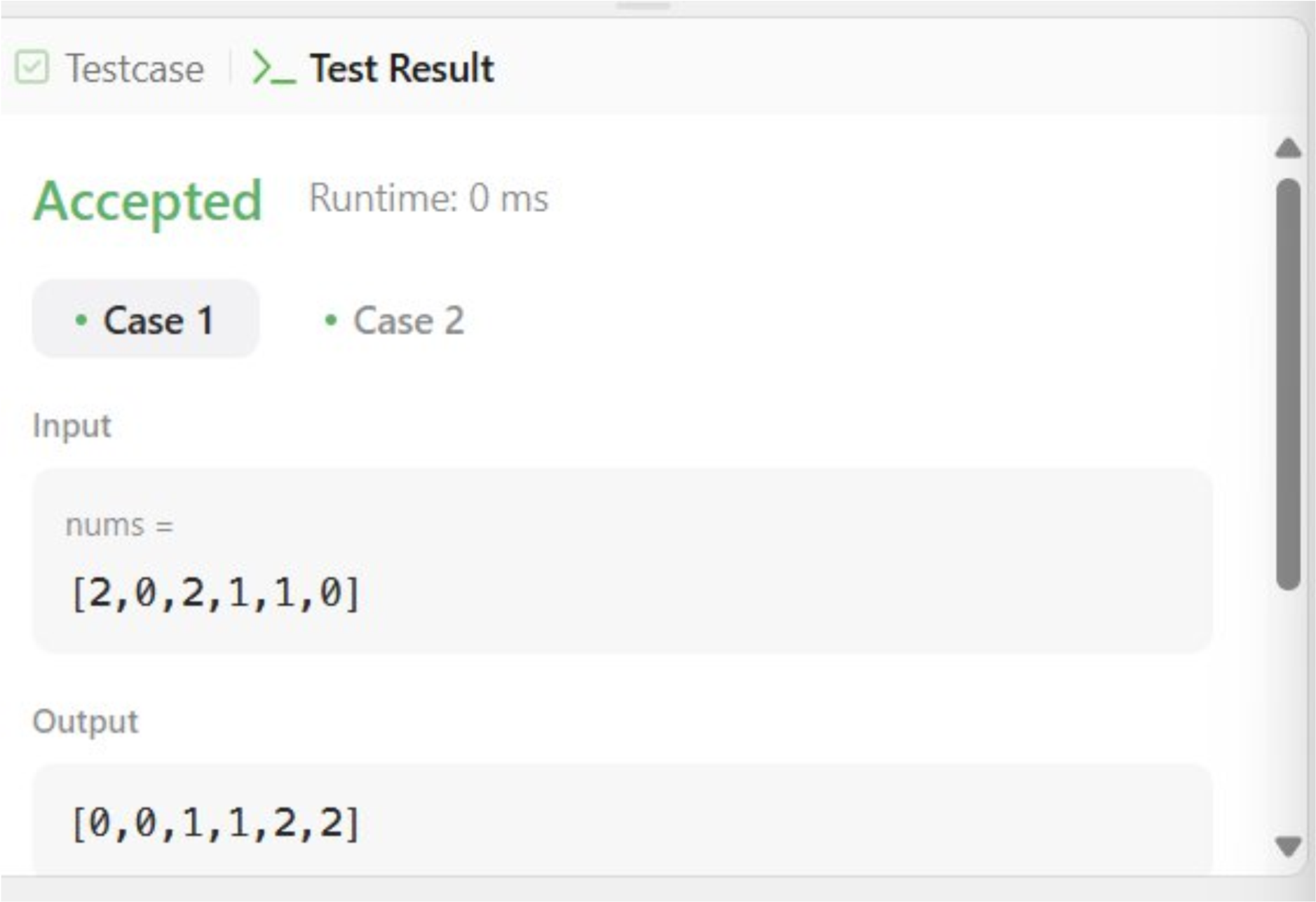
swap(nums[mid], nums[high]); high--;

}

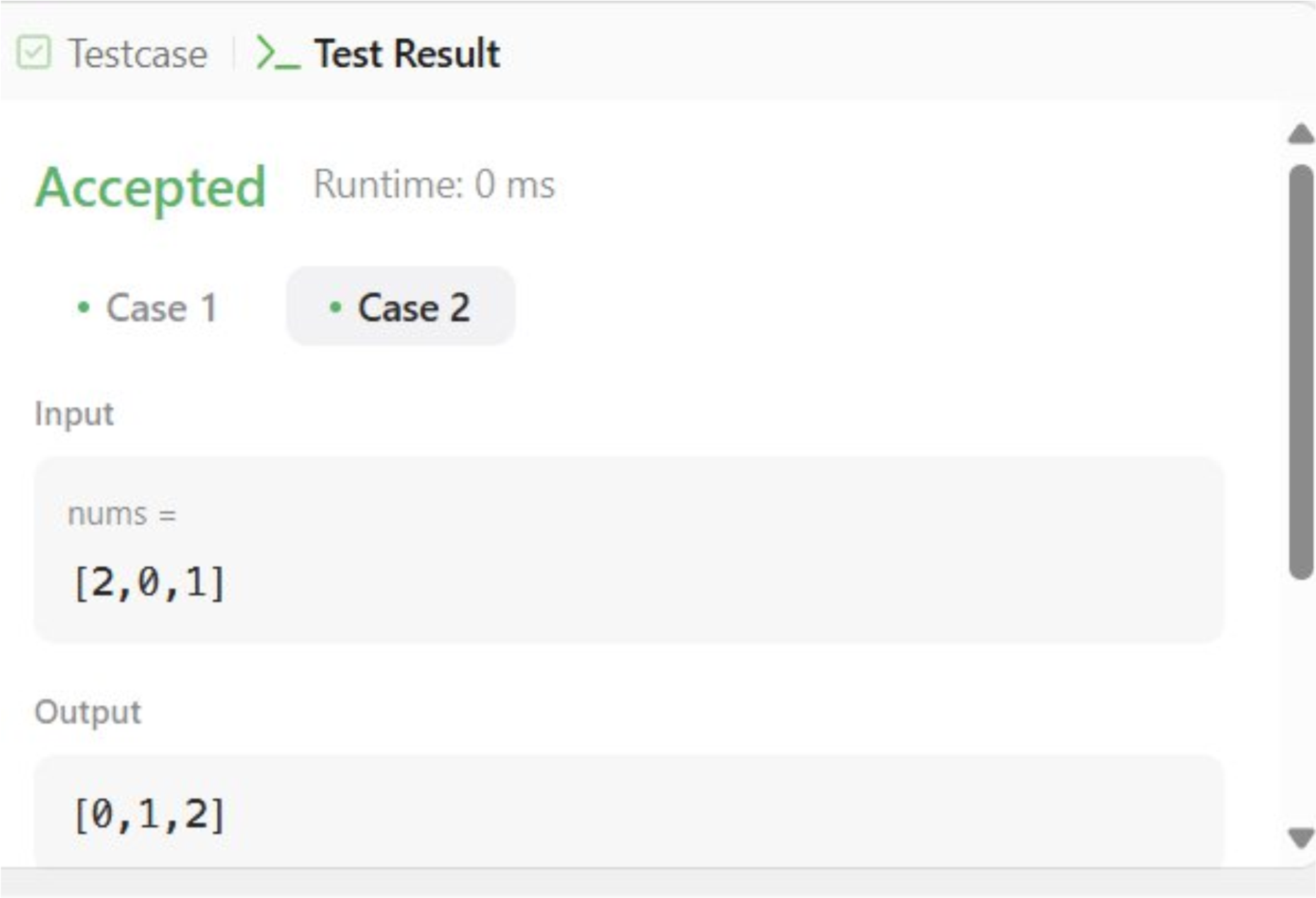
}

}

};



**OUTPUT:-**



# PROBLEM-4

**AIM:-**

Top K Frequency Element

## CODE:-

class Solution { public: vector<int> topKFrequent(vector<int>& nums, int k) { unordered\_map<int, int> ump;

for(int i:

nums){ ump[i]++;

}

priority\_queue<pair<int, int>>pq;

for(auto i: ump){

pq.push({i.second,i.first});

}

vector<int> res;

while(k--){ auto [elem, count] =

pq.top(); res.push\_back(count); pq.pop(); }

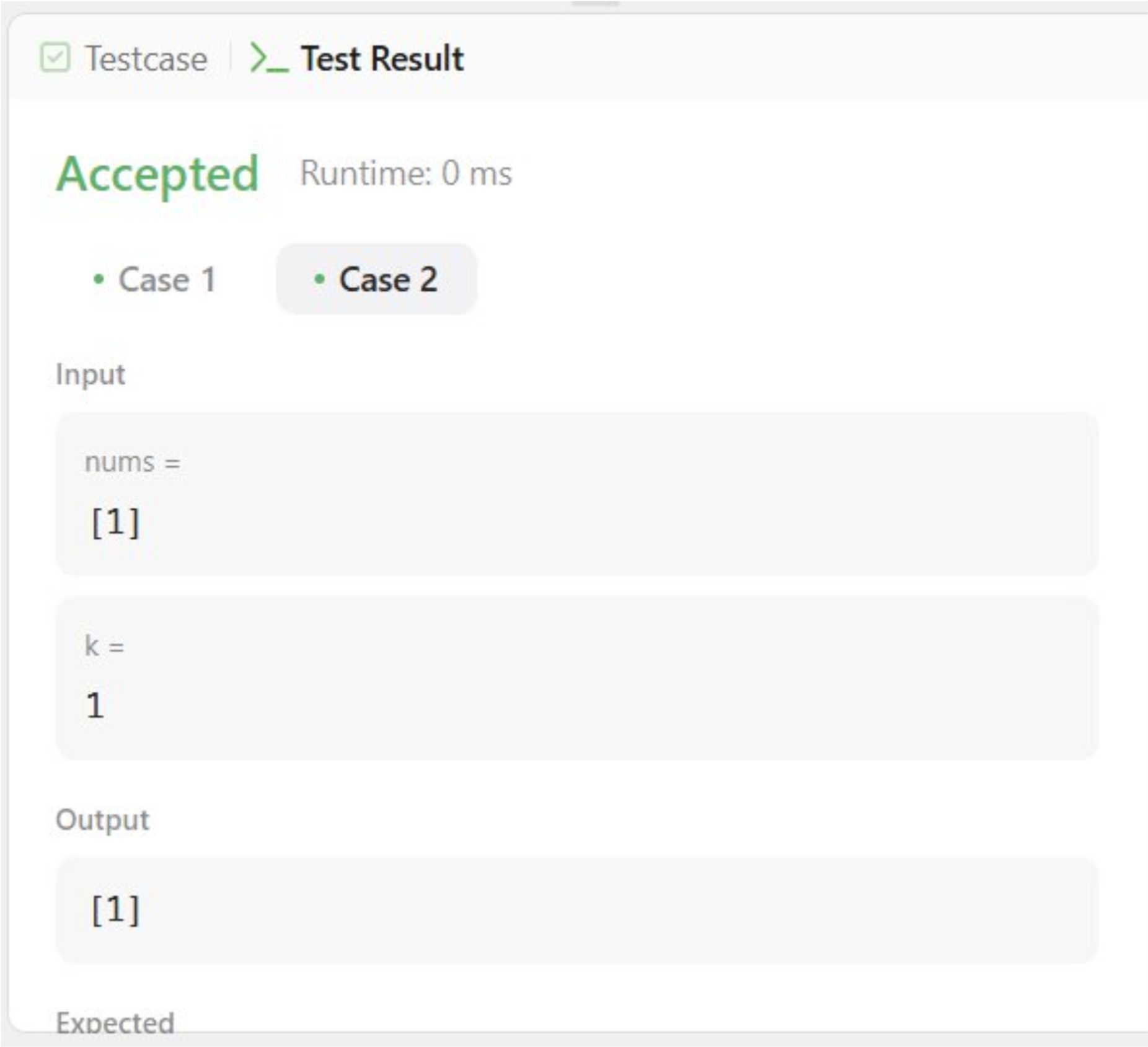
return res;

}

};



**OUTPUT:-**



# PROBLEM-5

**AIM:- Find Peak Element**

## CODE:-

class Solution { public:

void solve(vector<int>&nums,int l,int r, int&ans){ if(l>r || ans>-1) return ; int m=(r-l)/2+l;

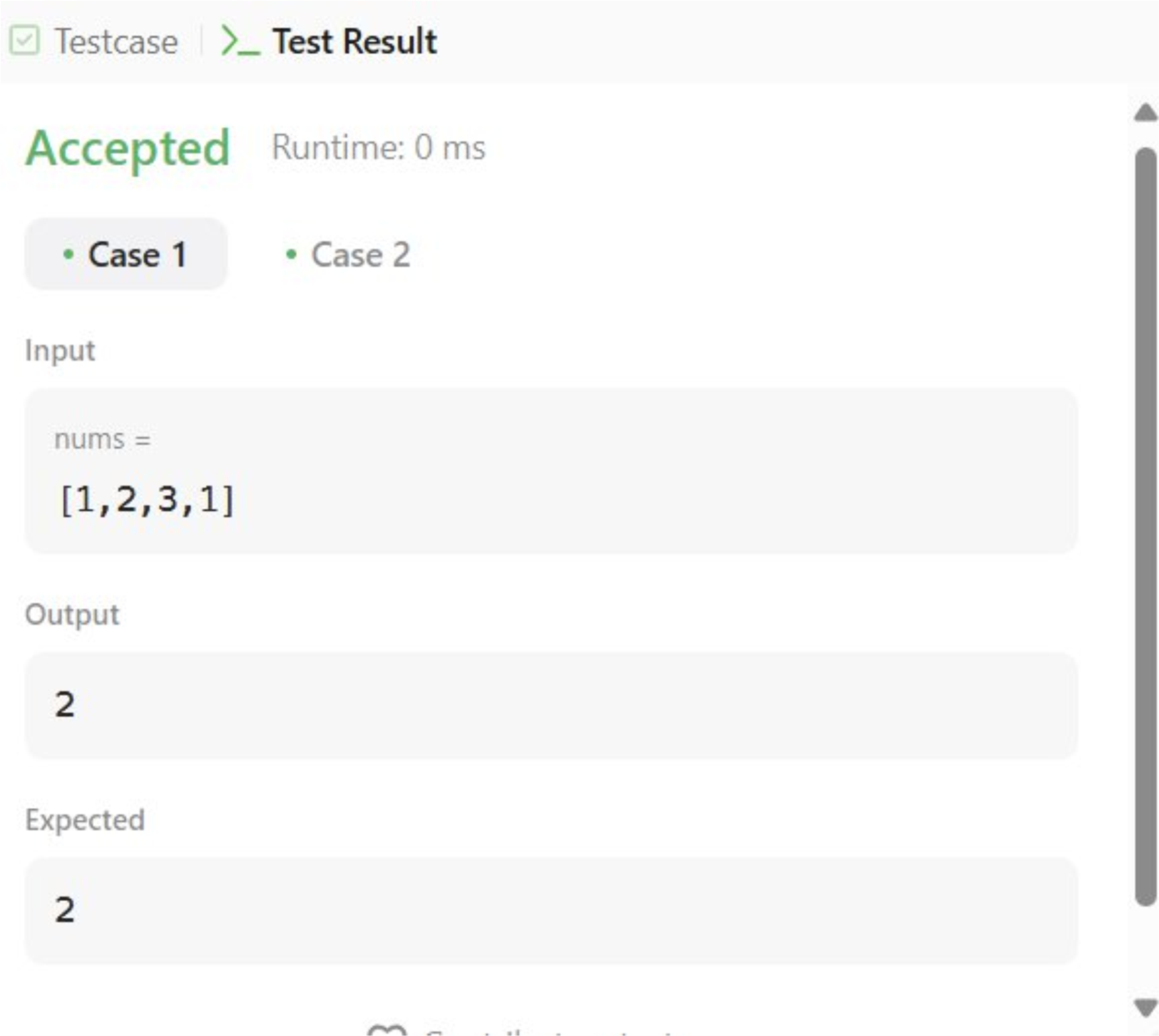
if(!(m-1>=0 && nums[m]<nums[m-1]) &&

!(m+1<nums.size() && nums[m]<nums[m+1])) ans=m; solve(nums, l, m-1, ans); solve(nums, m+1, r, ans); return ;

} int findPeakElement(vector<int>& nums) { int ans=-1, l=0, r=nums.size()-1; solve(nums, l, r, ans); return ans;

}

};



**OUTPUT:-**

