Assignment 5

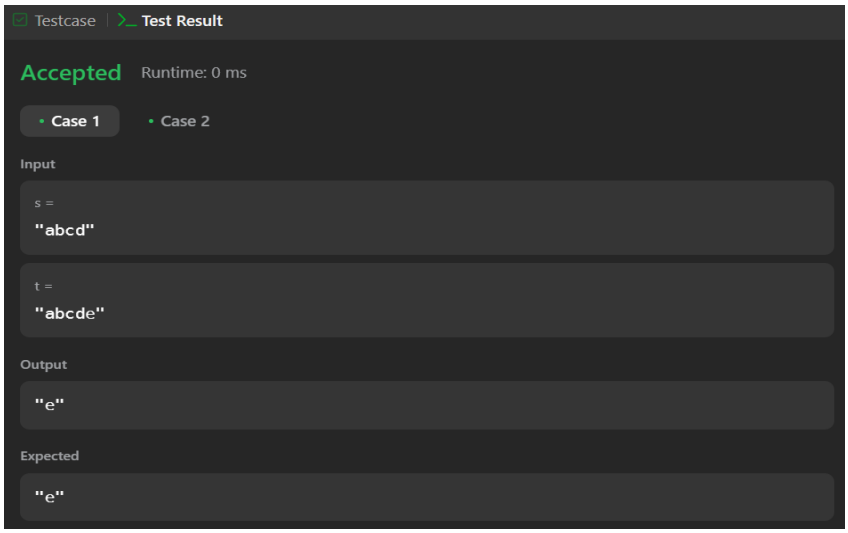
Student Name: Nishant Kumar
UID: 22BCS15009
Branch: CSE
Section/Group: 22BCS_FL_IOT-602 A
Semester: 6th
Date of Performance: 04/03/2025
Subject Name: Advanced Programming Lab - 2
Subject Code: 22CSP-351

Problem 389. Find the Difference

- **Implementation/Code:**

```
class Solution {
public:
    char findTheDifference(string s, string t) {
        char result = 0;
        for (char c : s) result ^= c;
        for (char c : t) result ^= c;
        return result;
    }
};
```

- **Output:**



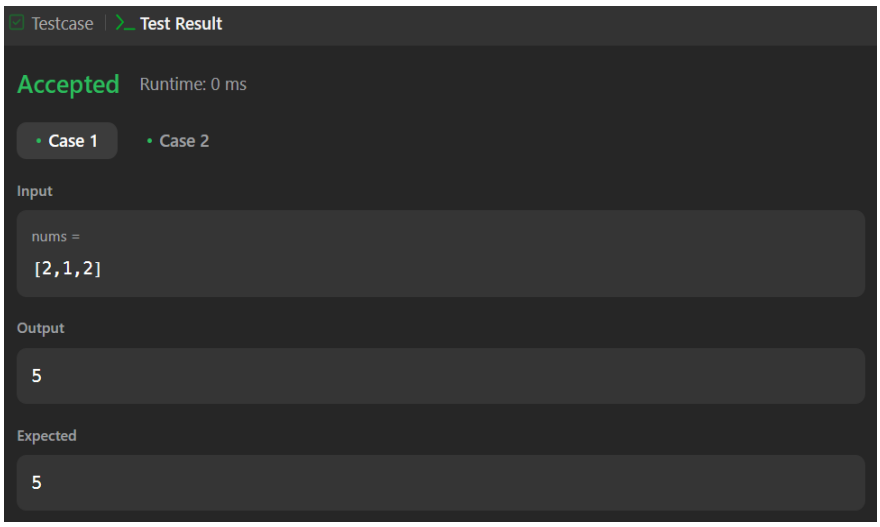
Problem 976. Largest Perimeter Triangle

- **Implementation/Code:**

```
class Solution {
public:
    int largestPerimeter(vector<int>& nums) {
```

```
        sort(nums.begin(), nums.end());
        for (int i = nums.size() - 1; i >= 2; i--) {
            if (nums[i - 1] + nums[i - 2] > nums[i]) {
                return nums[i] + nums[i - 1] + nums[i - 2];
            }
        }
        return 0;
    }
};
```

- **Output:**



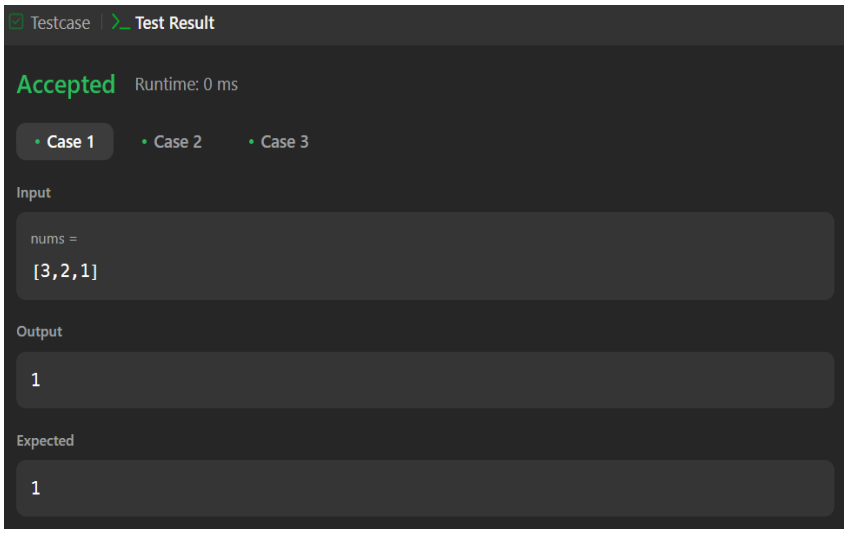
Problem 414. Third Maximum Number

- **Implementation/Code:**

```
class Solution {
public:
    int thirdMax(vector<int>& nums) {
        set<int> s;

        for (int num : nums) {
            s.insert(num);
            if (s.size() > 3) {
                s.erase(s.begin());
            }
        }
        if (s.size() == 3) {
            return *s.begin();
        }
        return *s.rbegin();
    }
};
```

- **Output:**



Problem 451. Sort Characters By Frequency

- **Implementation/Code:**

```
class Solution {
public:
    string frequencySort(string s) {
        unordered_map<char, int> frequencyMap;
        for (char ch : s) {
            ++frequencyMap[ch];
        }
        vector<char> uniqueChars;
        for (auto& keyValue : frequencyMap) {
            uniqueChars.push_back(keyValue.first);
        }
        sort(uniqueChars.begin(), uniqueChars.end(), [&](char a, char b) {
            return frequencyMap[a] > frequencyMap[b];
        });
        string result;
        for (char ch : uniqueChars) {
            result += string(frequencyMap[ch], ch);
        }
        return result;
    }
};
```

- **Output:**

Problem 452. Minimum Number of Arrows to Burst Balloons

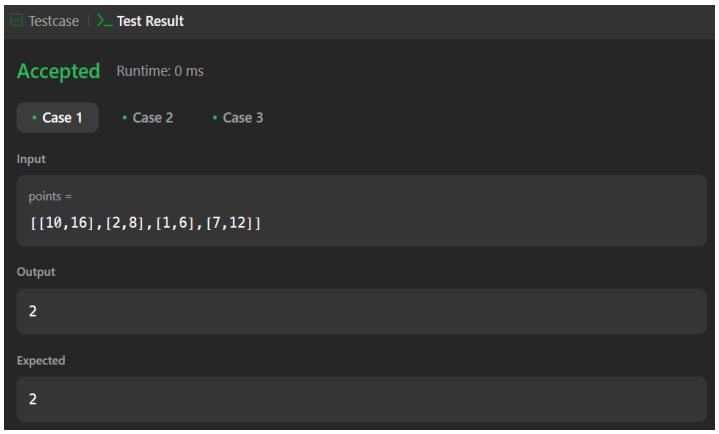
- **Implementation/Code:**

```
class Solution {
public:
    int findMinArrowShots(vector<vector<int>>& points) {
        if (points.empty()) return 0;

        sort(points.begin(), points.end(), [](vector<int>& a, vector<int>& b) {
            return a[1] < b[1];
        });
        int arrows = 1;
        int lastArrow = points[0][1];

        for (int i = 1; i < points.size(); i++) {
            if (points[i][0] > lastArrow) {
                arrows++;
                lastArrow = points[i][1];
            }
        }
        return arrows;
    }
};
```

- **Output:**



Problem 881. Boats to Save People

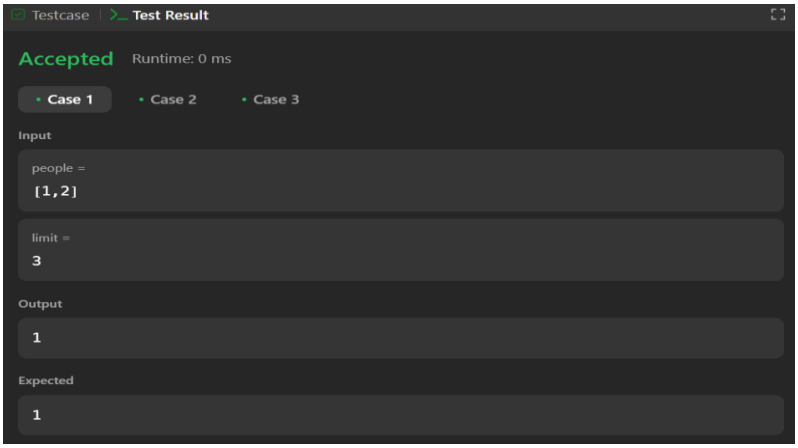
- **Implementation/Code:**

```
class Solution {
public:
    int numRescueBoats(vector<int>& people, int limit) {
        sort(people.begin(), people.end());

        int left = 0, right = people.size() - 1;
        int boats = 0;

        while (left <= right) {
            if (people[left] + people[right] <= limit) {
                left++;
            }
            right--;
            boats++;
        }
        return boats;
    }
};
```

- **Output:**



Problem 973. K Closest Points to Origin

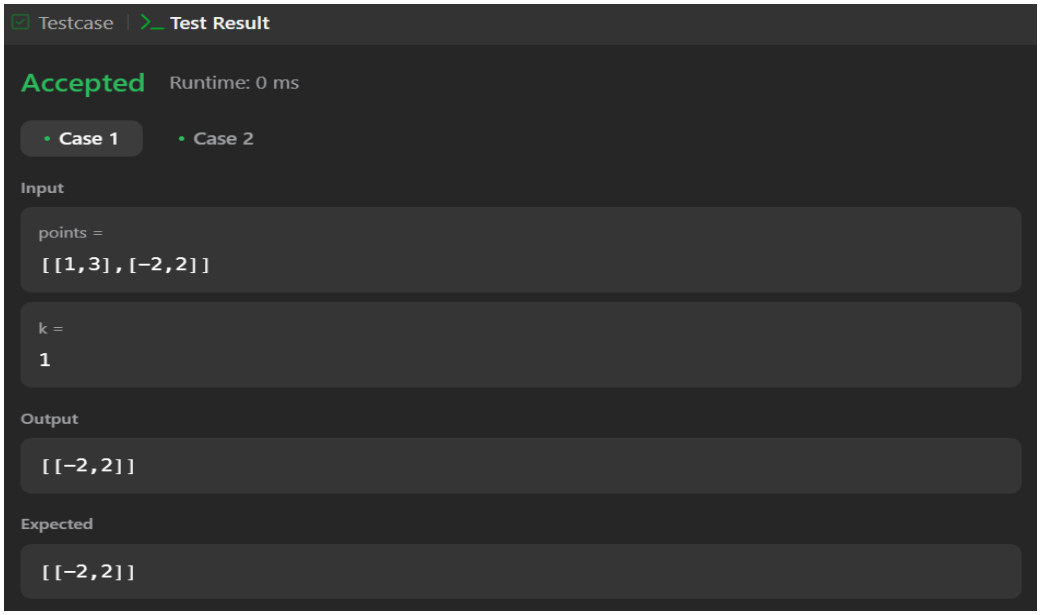
- **Implementation/Code:**

```
class Solution {
public:
    vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {
        priority_queue<pair<int, vector<int>>> maxHeap;

        for (auto& point : points) {
            int x = point[0], y = point[1];
            int dist = x * x + y * y;

            maxHeap.push({dist, point});
            if (maxHeap.size() > k) {
                maxHeap.pop();
            }
        }
        vector<vector<int>> result;
        while (!maxHeap.empty()) {
            result.push_back(maxHeap.top().second);
            maxHeap.pop();
        }
        return result;
    }
};
```

- **Output:**



Problem 1338. Reduce Array Size to The Half

- **Implementation/Code:**

```
class Solution {
public:
    int minSetSize(vector<int>& arr) {
        unordered_map<int, int> freq;
        for (int num : arr) {
            freq[num]++;
        }

        vector<int> freqCounts;
        for (auto& it : freq) {
            freqCounts.push_back(it.second);
        }
        sort(freqCounts.rbegin(), freqCounts.rend());

        int removed = 0, count = 0, half = arr.size() / 2;
        for (int f : freqCounts) {
            removed += f;
            count++;
            if (removed >= half) break;
        }
        return count;
    }
};
```

- **Output:**