Experiment-10

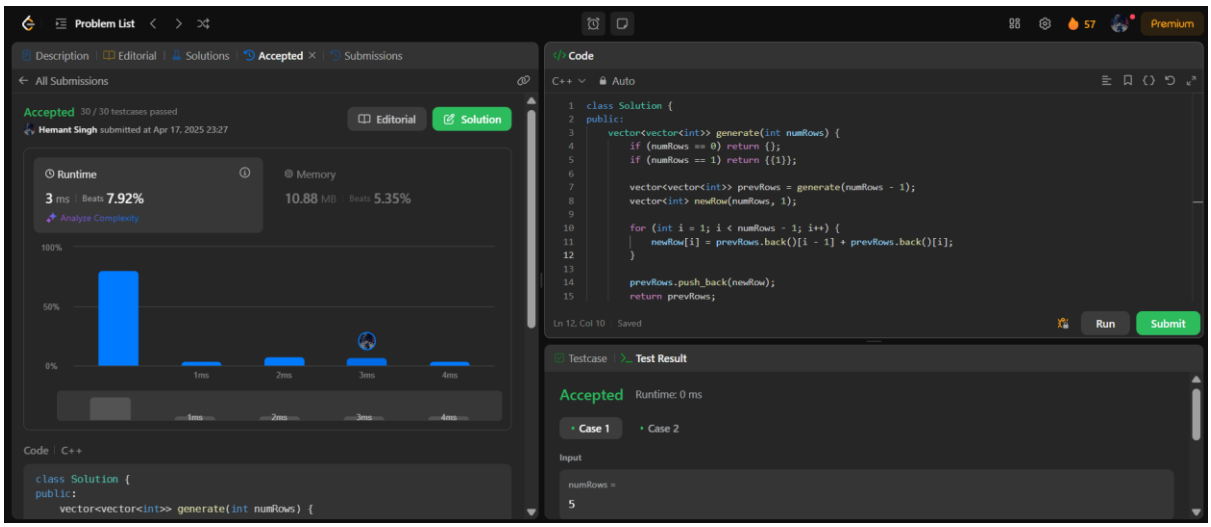
Name:- Hemant Singh **UID:- 22BCS12820**

Q1. https://leetcode.com/problems/pascals-triangle/submissions/1609779949/

Code:

```
class Solution {
public:
    vector<vector<int>> generate(int numRows) {
        if (numRows == 0) return {};
        if (numRows == 1) return {{1}};
        vector<vector<int>> prevRows = generate(numRows - 1);
        vector<int> newRow(numRows, 1);
        for (int i = 1; i < numRows - 1; i++) {
            newRow[i] = prevRows.back()[i - 1] + prevRows.back()[i];
        }
        prevRows.push_back(newRow);
        return prevRows;
    }
};
```

Output:-

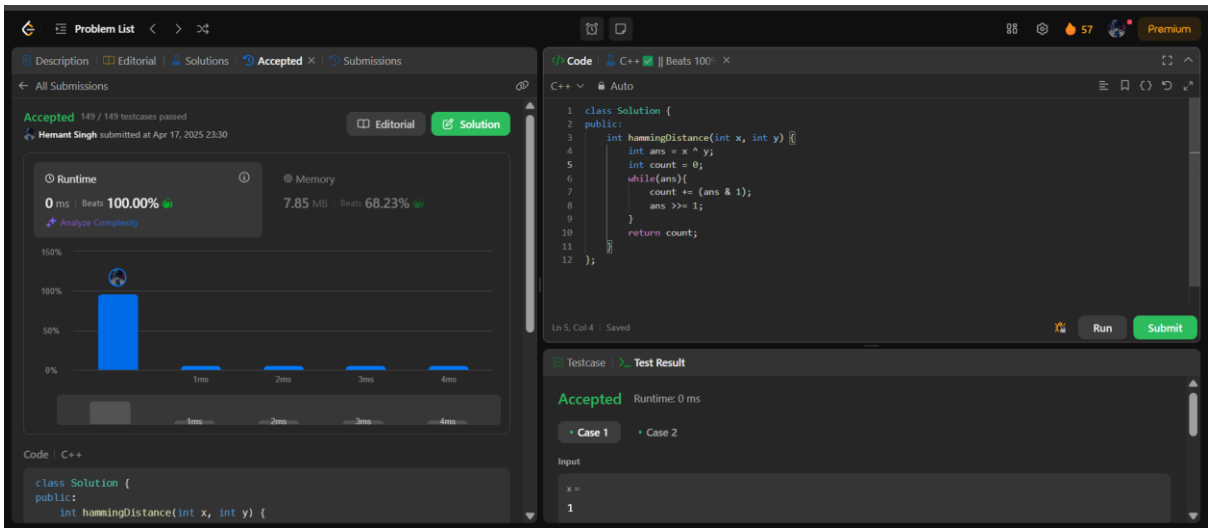


Q2. https://leetcode.com/problems/hamming-distance/submissions/1609783026/

Code:

```
class Solution {
public:
    int hammingDistance(int x, int y) {
        int ans = x ^ y;
        int count = 0;
        while(ans){
            count += (ans & 1);
            ans >>= 1;
        }
        return count;
    }
};
```

Output:

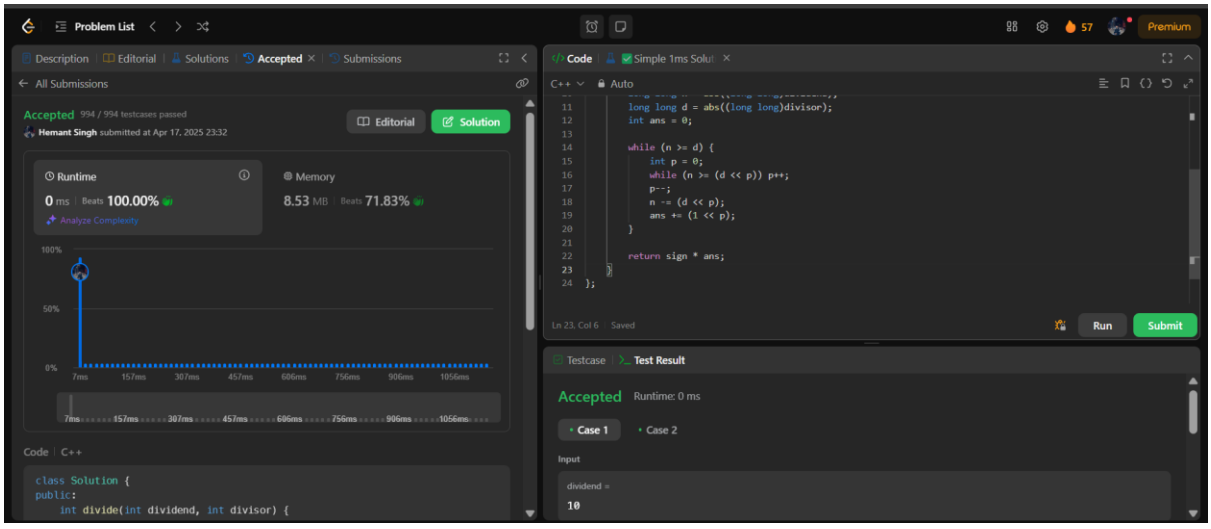


Q3. https://leetcode.com/problems/divide-two-integers/

Code:

```
class Solution {
public:
    int divide(int dividend, int divisor) {
        if (dividend == divisor) return 1;
        if (dividend == INT_MIN && divisor == -1) return INT_MAX;
        if (divisor == 1) return dividend;
        int sign = (dividend < 0) ^ (divisor < 0) ? -1 : 1;
        long long n = abs((long long)dividend);
        long long d = abs((long long)divisor);
        int ans = 0;
        while (n >= d) {
            int p = 0;
            while (n >= (d << p)) p++;
            p--;
            n -= (d << p);
            ans += (1 << p);
        }
        return sign * ans;
    }
};
```

Output:

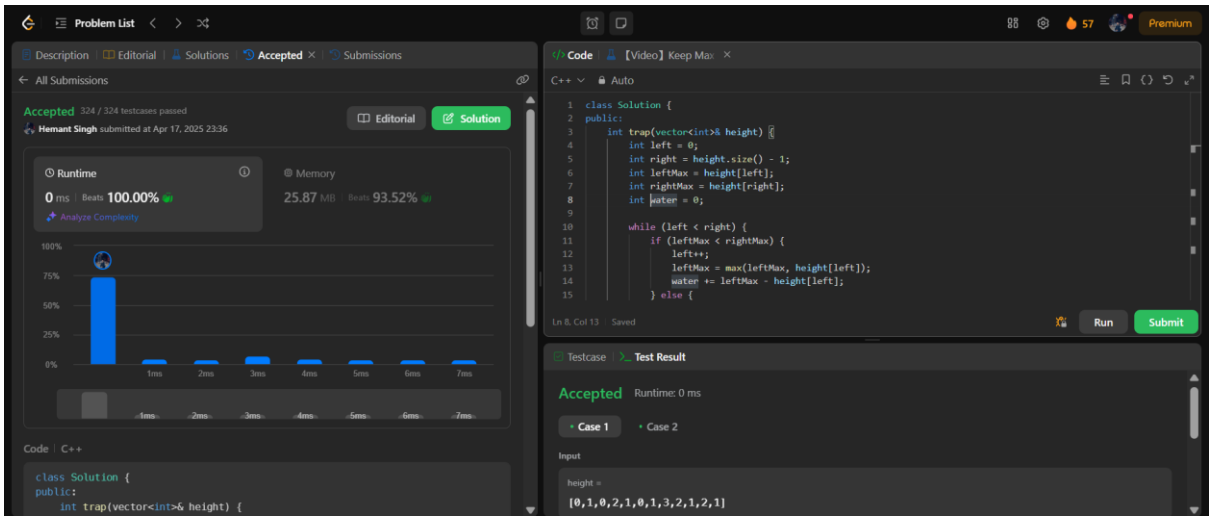


Q4. https://leetcode.com/problems/trapping-rain-water/submissions/1609788466/

Code:

```
class Solution {
public:
    int trap(vector<int>& height) {
        int left = 0;
        int right = height.size() - 1;
        int leftMax = height[left];
        int rightMax = height[right];
        int water = 0;
        while (left < right) {
            if (leftMax < rightMax) {
                left++;
                leftMax = max(leftMax, height[left]);
                water += leftMax - height[left];
            } else {
                right--;
                rightMax = max(rightMax, height[right]);
                water += rightMax - height[right];
            }}
        return water; }};
```

Output:



Q5. https://leetcode.com/problems/maximum-number-of-tasks-you-can-assign/submissions/1609794465/?envType=problem-list-v2&envId=greedy

Code:

```
class Solution {
    int s;
    vector<int> ts, ws;
public:
    int maxTaskAssign(vector<int>& tasks, vector<int>& workers, int pills, int strength) {
        sort(tasks.begin(), tasks.end());
        sort(workers.begin(), workers.end());
        int n = tasks.size();
        int m = workers.size();
        int left = 0;
        int right = min(n, m);
        int answer = 0;
        while (left <= right) {
            int mid = (left + right)/2;
            multiset<int> workersSet(workers.end() - mid, workers.end());
            int pillCountRemaining = pills;
            for (int i = mid-1; i>=0; --i) {
                auto it = prev(workersSet.end());
                if (*it < tasks[i]) {
                    if (pillCountRemaining == 0) break;
                    it = workersSet.lower_bound(tasks[i] - strength);
                    if (it == workersSet.end()) break;
                    pillCountRemaining--;
                }
                workersSet.erase(it);
            }
            if (workersSet.empty()) {
                answer = mid;
                left = mid + 1;
            } else {
```

```
            right = mid - 1;
        }
    }
    return answer;
  }
};
```

Output: