

NAME- Ashish kumar singh

SECTION- 22BCS-IOT-614/B

UID- 22BCS16892

Question – 1:

❖ HOUSE ROBBER

CODE:

```
class Solution {
public:
    int rob(vector<int>& nums) {
        int n = nums.size();
        if (n == 0) return 0;
        if (n == 1) return nums[0];

        vector<int> dp(n, 0);
        dp[0] = nums[0];
        dp[1] = max(nums[0], nums[1]);

        for (int i = 2; i < n; i++) {
            dp[i] = max(dp[i - 1], nums[i] + dp[i - 2]);
        }

        return dp[n - 1];
    }
};
```

SCREENSHOT:

The screenshot shows a LeetCode submission interface for the 'House Robber' problem. The submission is accepted, with a runtime of 0 ms and a memory usage of 10.58 MB. The code is written in C++ and implements a dynamic programming solution. The test result shows that the code passes all test cases, including Case 1 and Case 2. The input for Case 2 is [1,2,3,1] and the output is 4.

Accepted

0 ms | Best 100.00% | Memory 10.58 MB | Beats 61.79%

Runtime

Accepted

Runtime: 0 ms

Case 1 | Case 2

Input

nums = [1,2,3,1]

Output

4

Expected

4

Question – 2:

❖ 70. Climbing Stairs

CODE:

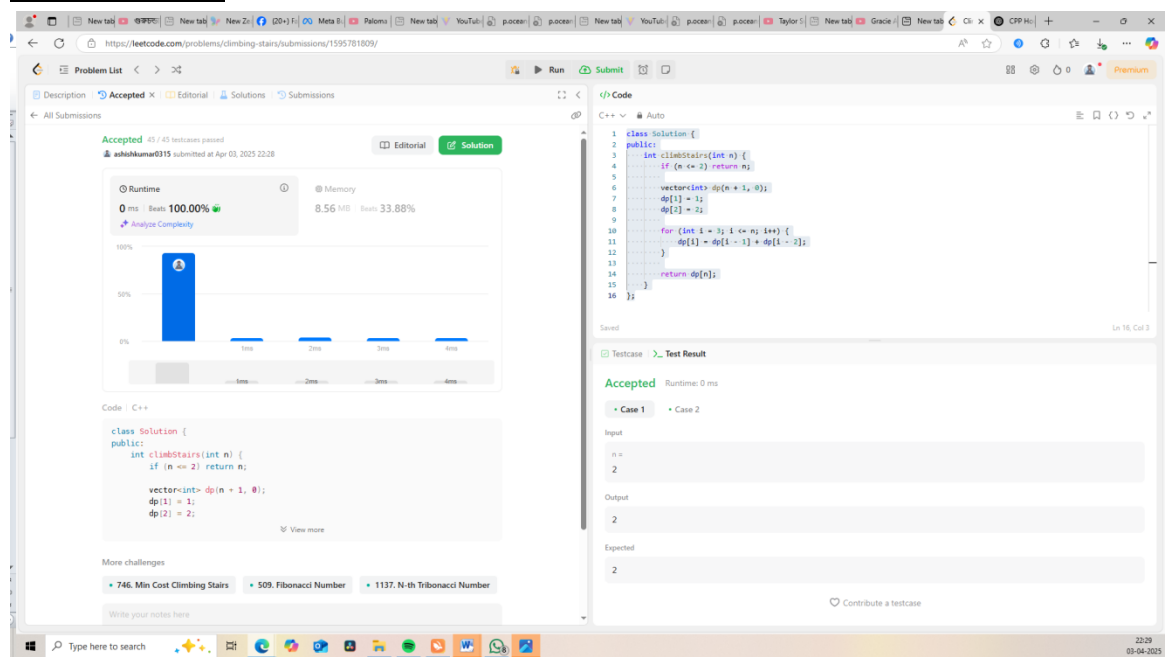
```
class Solution {
public:
    int climbStairs(int n) {
        if (n <= 2) return n;

        vector<int> dp(n + 1, 0);
        dp[1] = 1;
        dp[2] = 2;

        for (int i = 3; i <= n; i++) {
            dp[i] = dp[i - 1] + dp[i - 2];
        }

        return dp[n];
    }
};
```

SCREENSHOT:



Question – 3:

62. Unique Paths

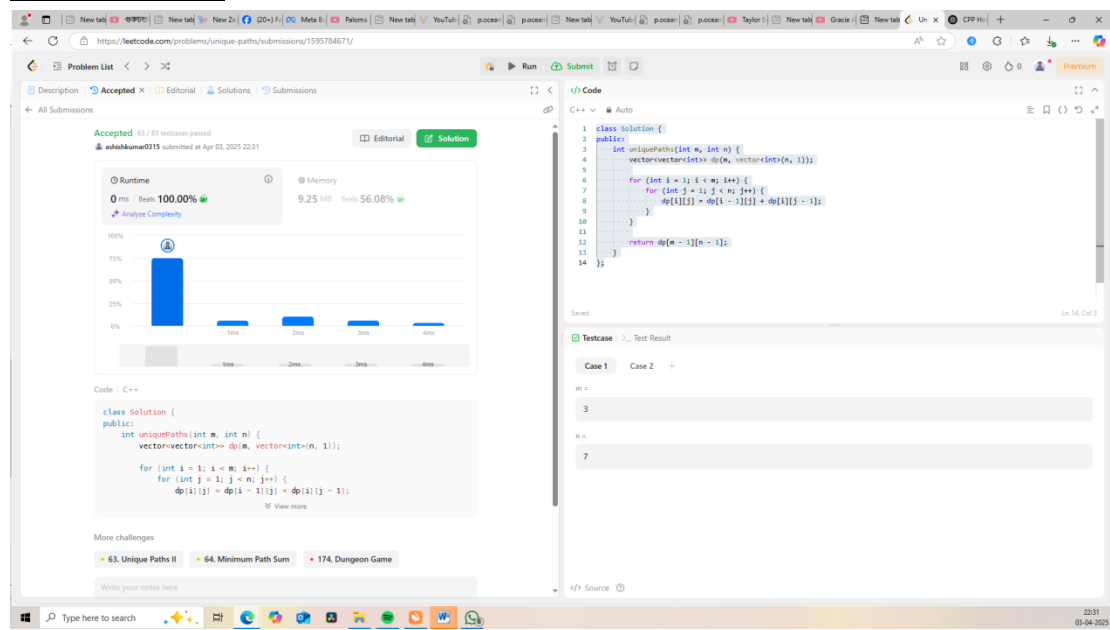
CODE:

```
class Solution {
public:
    int uniquePaths(int m, int n) {
        vector<vector<int>> dp(m, vector<int>(n, 1));

        for (int i = 1; i < m; i++) {
            for (int j = 1; j < n; j++) {
                dp[i][j] = dp[i - 1][j] + dp[i][j - 1];
            }
        }

        return dp[m - 1][n - 1];
    }
};
```

SCREENSHOT:



Question – 4:

❖ 53. Maximum Subarray

CODE:

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

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

        return maxSum;
    }
};
```

SCREENSHOT:

The screenshot displays the LeetCode interface for the 'Maximum Subarray' problem. The top section shows the problem description and a bar chart indicating the solution's performance: 0 ms runtime, 100.00% speed, and 71.73 MB memory usage. The code editor on the right contains the C++ solution. The test result at the bottom shows 'Accepted' with a runtime of 0 ms. The input array is [-2, 1, -3, 4, -1, 2, 1, -5, 4] and the expected output is 6.

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

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

        return maxSum;
    }
};
```