**Problem - 4**

1. **Aim of the Experiment :**

You are climbing a staircase. It takes n steps to reach the top.

Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

1. **Objective of the Experiment :**

To find the distinct ways to climb stairs .

1. **Explanation :**

Since the problem contains an optimal substructure and has overlapping sub-problems, it can be solved using bottom-up approach by dynamic programming (We can use brute force method also but it will be time consuming according to the time complexity of the algorithm).

One can reach the ith step in one of the two ways :

Take one step from (i – 1) th step.

Take two steps from (i – 2) th step.

1. **Algorithm :**

**Step 1:** Initialise a dp[] array of size N + 1.

**Step 2:** Assign dp[0] and dp[1] to 1.

**Step 3:** Iterate a loop from 2 till N and for each iteration:

dp[i] = dp[i – 1] + dp[i – 2]

**Step 4:** Return the value of dp[N]

1. **Code :**

class Solution {

public:

    int climbStairs(int N)

    {

        int dp[N+1];

        dp[0] = 1;

        dp[1] = 1;

        for (int i = 2; i <= N; i++){

            dp[i] = dp[i-1] + dp[i-2];

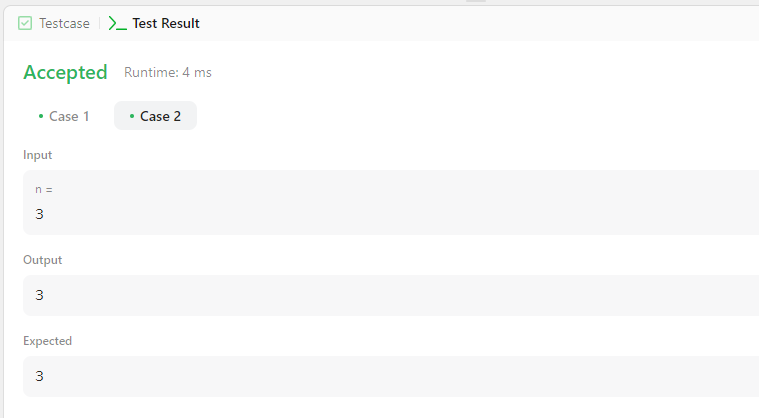
        }

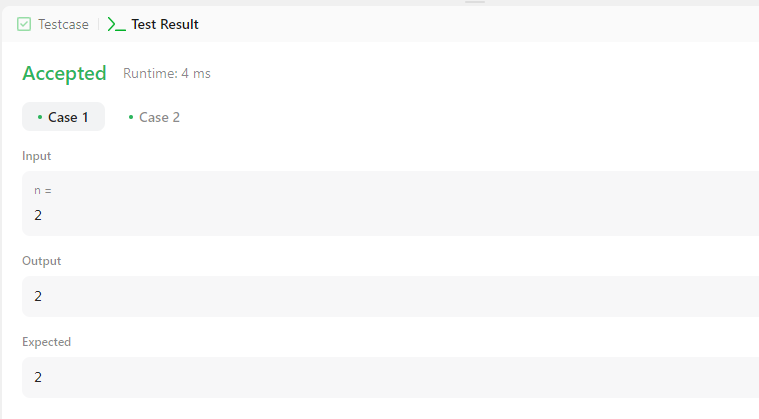
        return dp[N];

    }

};

1. **Output:**

****

****

**Learning outcomes (What I have learnt):**

1. **Learned what is dynamic programming and how it is useful in C++.**
2. **Learned how to do competitive coding which approach is better with less time and space complexity.**