



## Experiment 6

**Student Name:** Amit Kumar Sahu

**Branch:** BE-CSE

**Semester:** 6th

**Subject Name:** AP Lab-2

**UID:** 22BCS50073

**Section/Group:** 22BCS\_NTPP\_IOT\_602'A'

**Date of Performance:** 24-02-25

**Subject Code:** 22CSP-351

### 1. Aim: Climbing Stairs

### 2. Objective:

**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?**

### 3. Implementation/Code:

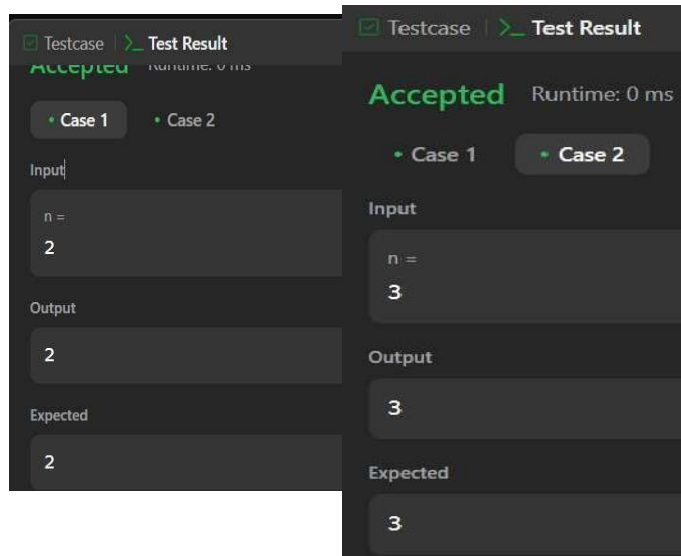
```
classSolution{
    publicintclimbStairs(intn){ if
        (n == 0) return 1;
        if(n == 1)return 1;

        int[]dp=newint[n+1];
        dp[0] = 1;
        dp[1]=1;

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

        returndp[n];
    }
}
```

### 4. Output



## 5. Learning Outcome:

- Understand dynamic programming to solve recurrence-based problems.
- Learn how to use an array to store intermediate results for optimization.
- Recognize the Fibonacci-like nature of the "climbing stairs" problem.
- Implement iterative solutions to reduce redundant calculations.
- Improve problem-solving skills with bottom-up dynamic programming

## Question 2.

### 1. Aim: Maximum Subarray

### 2. Objective:

**Given an integer array num, find the subarray with the largest sum, and return its sum .**

### 3. Implementation/Code:

```
class Solution {
    public int maxSubArray(int[] nums) { int
        maxSum = nums[0];
        int currentSum = nums[0];
```

```

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

```

## 4. Output

**Accepted** Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =  
[-2, 1, -3, 4, -1, 2, 1, -5, 4]

Output

6

Expected

6

**Accepted** Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =  
[1]

Output

1

Expected

1

**Accepted** Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

nums =  
[5, 4, -1, 7, 8]

Output

23

Expected

23

## 5. Learning Outcome:

- ❑ Understand dynamic programming to solve recurrence-based problems.
- ❑ Learn how to use an array to store intermediate results for optimization.
- ❑ Recognize the Fibonacci-like nature of the "climbing stairs" problem.
- ❑ Implement iterative solutions to reduce redundant calculations.
- ❑ Improve problem-solving skills with bottom-up dynamic programming.