EXPERIMENT-7

Student Name: Shubham Sharma **UID:**22BET10358
Branch: BE -IT **Section/Group:**22BET_IOT-703(A)
Semester: 6th **Subject Code:** 22ITP-351

PROBLEM-1

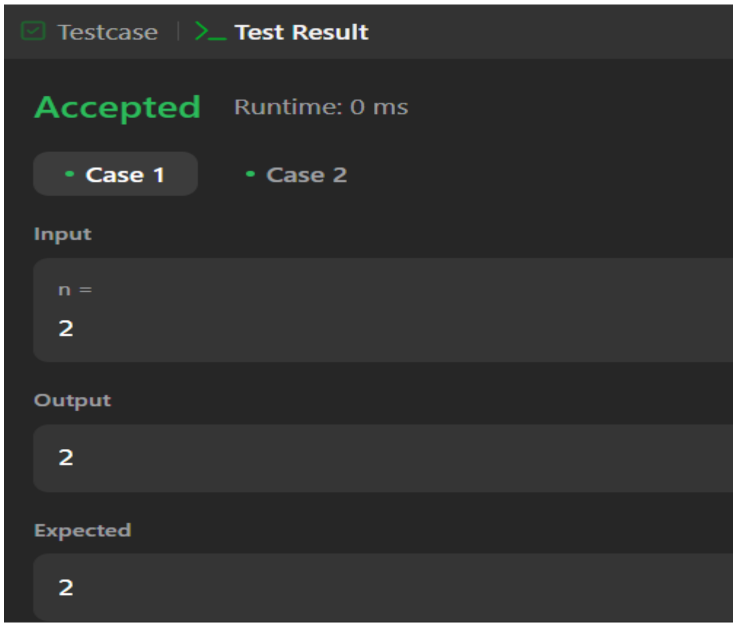
AIM:-

Climbing Stairs

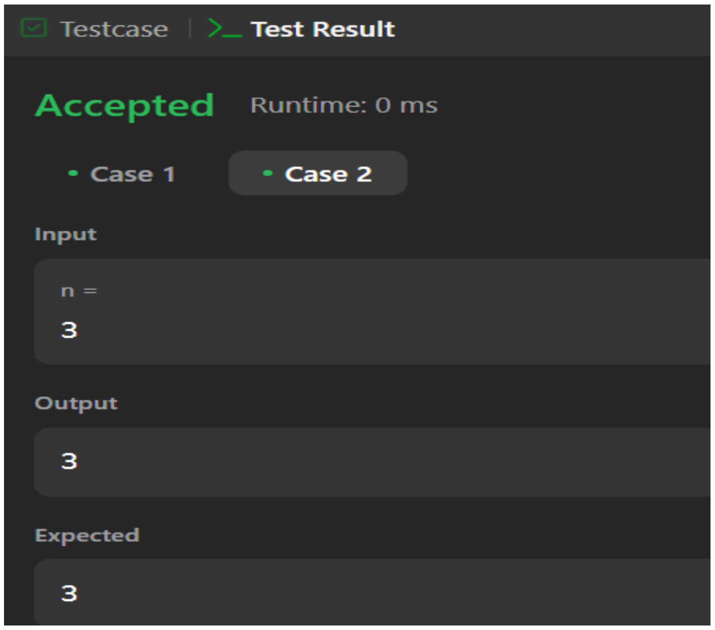
CODE:-

```
class Solution {
    public int climbStairs(int n) {
        if (n <= 3) return n;

        int prev1 = 3;
        int prev2 = 2;
        int cur = 0;

        for (int i = 3; i < n; i++) {
            cur = prev1 + prev2;
            prev2 = prev1;
            prev1 = cur;
        }

        return cur;
    }
}
```

OUTPUT:-



```
Testcase | Test Result
Accepted   Runtime: 0 ms
Case 1   Case 2
Input
n =
2
Output
2
Expected
2
```



```
Testcase | Test Result
Accepted   Runtime: 0 ms
Case 1   Case 2
Input
n =
3
Output
3
Expected
3
```

PROBLEM-2

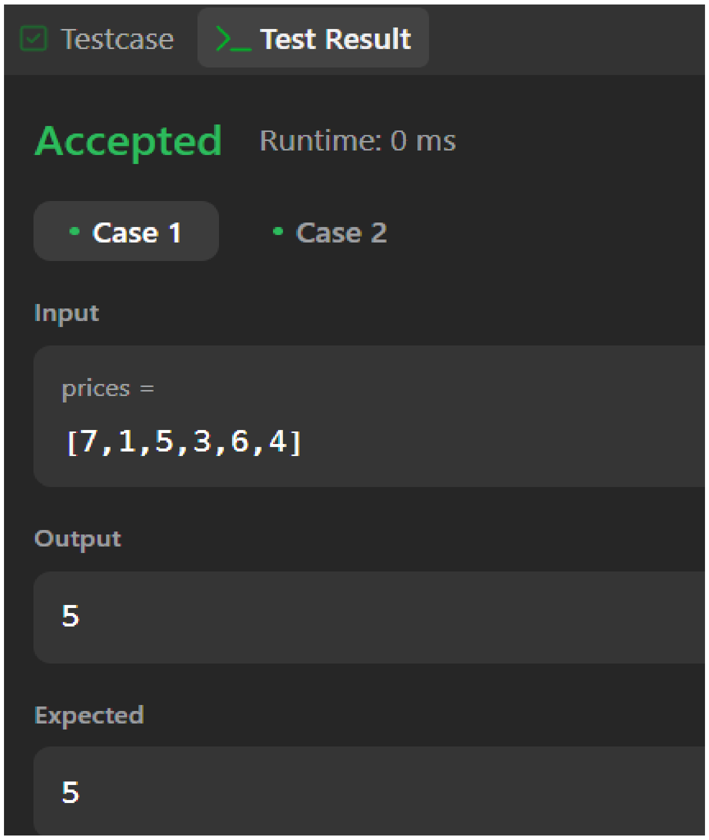
AIM:-

Best Time to Buy and Sell a Stock

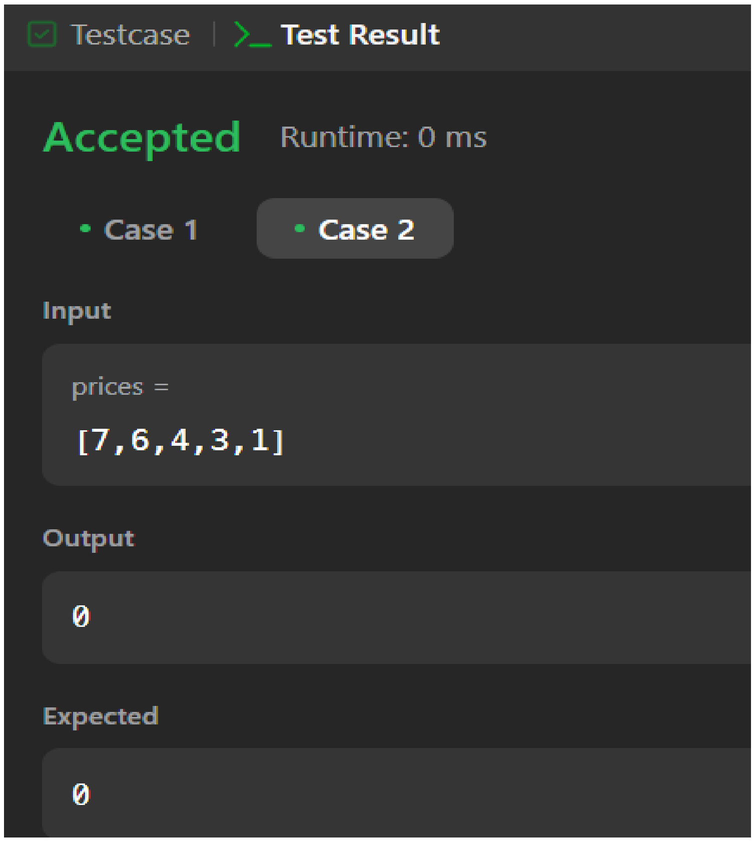
CODE:-

```
class Solution {
    public int maxProfit(int[] prices) {
        int buyPrice = prices[0];
        int profit = 0;

        for (int i = 1; i < prices.length; i++) {
            if (buyPrice > prices[i]) {
                buyPrice = prices[i];
            }

            profit = Math.max(profit, prices[i] - buyPrice);
        }

        return profit;
    }
}
```

OUTPUT:-



```
Accepted    Runtime: 0 ms
Input
prices =
[7,1,5,3,6,4]
Output
5
Expected
5
```



```
Accepted    Runtime: 0 ms
Input
prices =
[7,6,4,3,1]
Output
0
Expected
0
```

PROBLEM-3

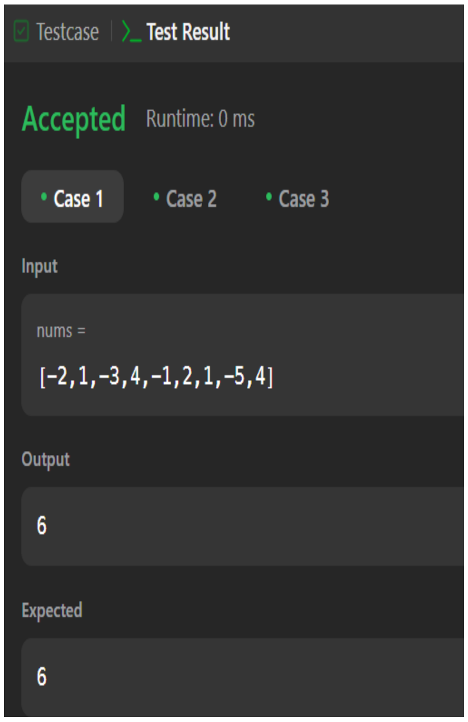
AIM:-

Maximum Subarray

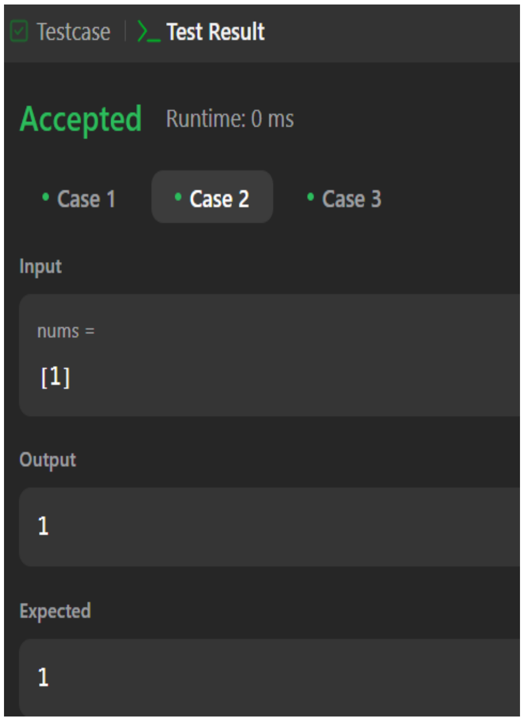
CODE:-

```
class Solution {
    public int maxSubArray(int[] nums) {
        int res = nums[0];
        int total = 0;

        for (int n : nums) {
            if (total < 0) {
                total = 0;
            }

            total += n;
            res = Math.max(res, total);
        }

        return res;
    }
}
```

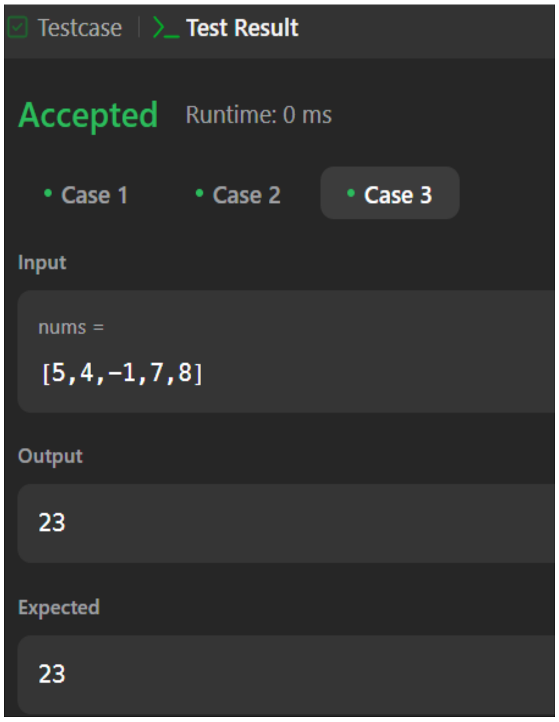
OUTPUT:-



```
Accepted   Runtime: 0 ms
Input
nums =
[-2,1,-3,4,-1,2,1,-5,4]
Output
6
Expected
6
```



```
Accepted   Runtime: 0 ms
Input
nums =
[1]
Output
1
Expected
1
```



```
Accepted   Runtime: 0 ms
Input
nums =
[5,4,-1,7,8]
Output
23
Expected
23
```

PROBLEM-4

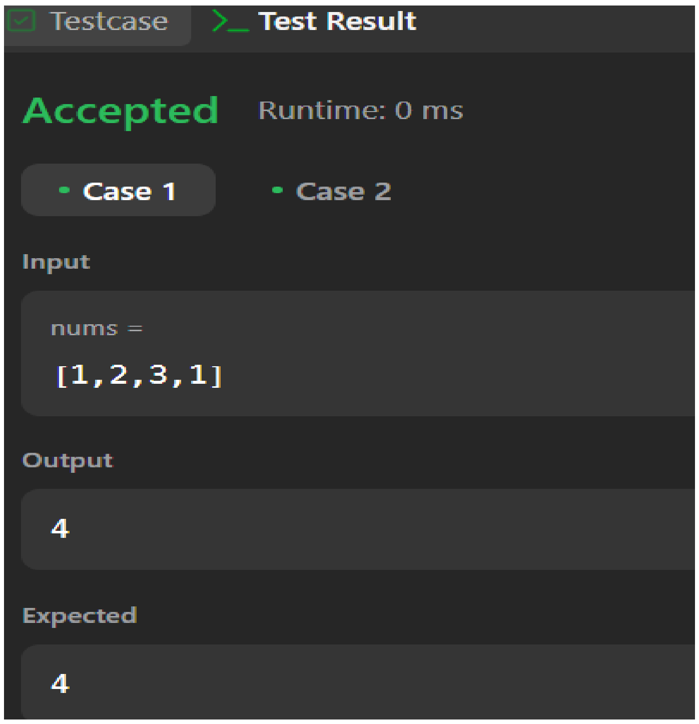
AIM:-

House Robber

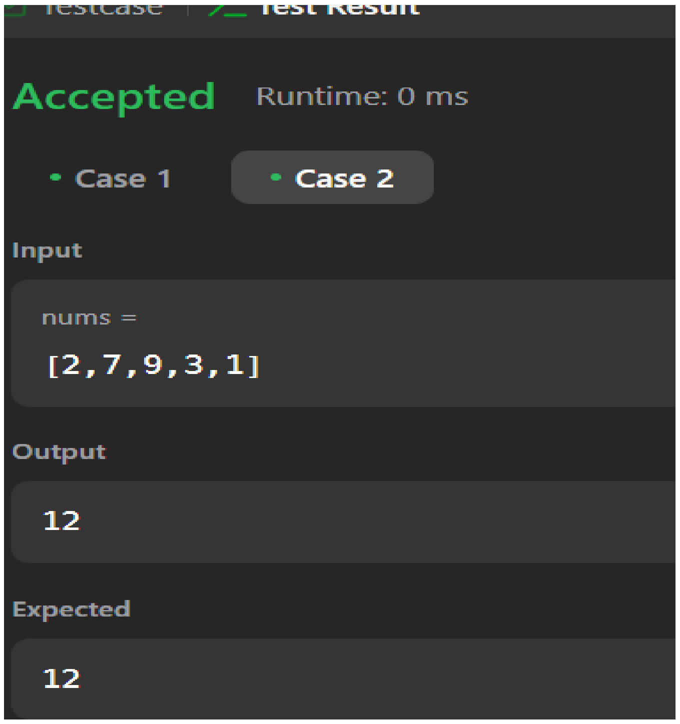
CODE:-

```
class Solution {
    public int rob(int[] nums) {
        int n = nums.length;

        if (n == 1) {
            return nums[0];
        }

        int[] dp = new int[n];

        dp[0] = nums[0];
        dp[1] = Math.max(nums[0], nums[1]);

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

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

OUTPUT:-



```
Accepted    Runtime: 0 ms

Input
nums =
[1,2,3,1]

Output
4

Expected
4
```



```
Accepted    Runtime: 0 ms

Input
nums =
[2,7,9,3,1]

Output
12

Expected
12
```

PROBLEM-5

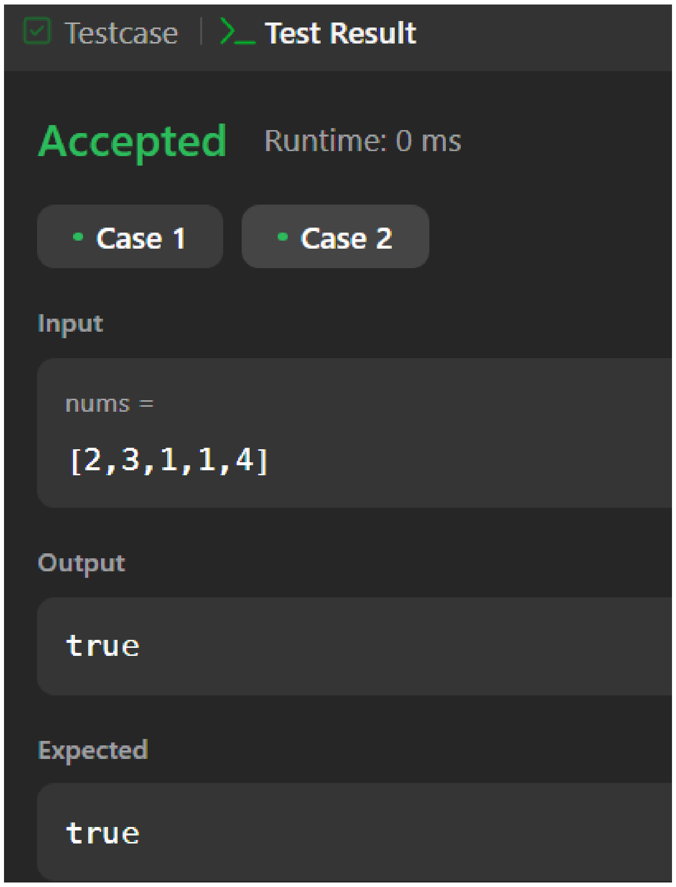
AIM:-

Jump Game

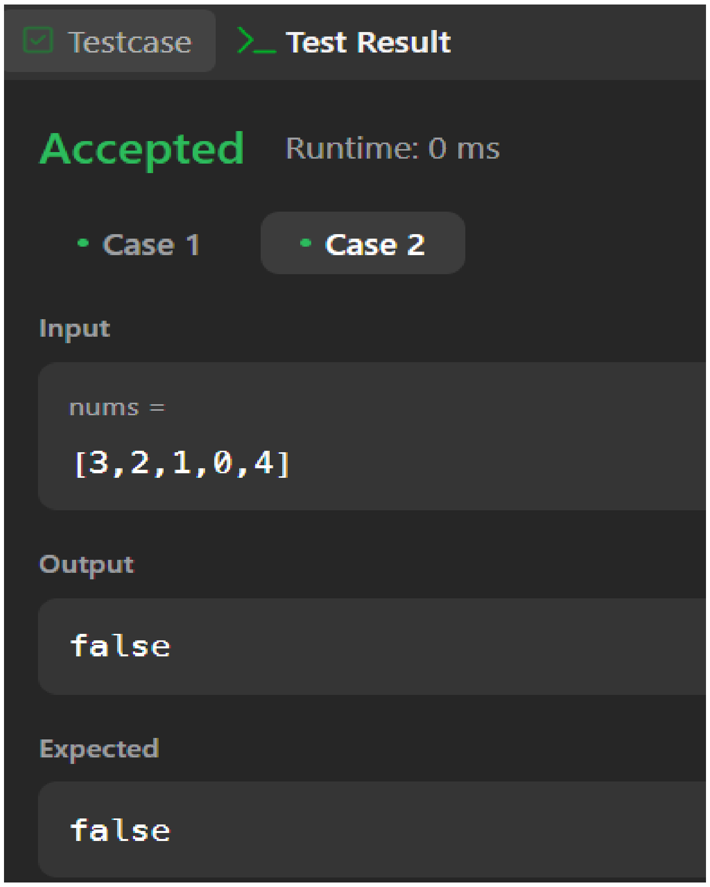
CODE:-

```
class Solution {
    public boolean canJump(int[] nums) {
        int goal = nums.length - 1;

        for (int i = nums.length - 2; i >= 0; i--) {
            if (i + nums[i] >= goal) {
                goal = i;
            }
        }

        return goal == 0;
    }
}
```

OUTPUT:-



```
Testcase | Test Result

Accepted   Runtime: 0 ms

• Case 1   • Case 2

Input
nums =
[2,3,1,1,4]

Output
true

Expected
true
```



```
Testcase   Test Result

Accepted   Runtime: 0 ms

• Case 1   • Case 2

Input
nums =
[3,2,1,0,4]

Output
false

Expected
false
```

PROBLEM-6

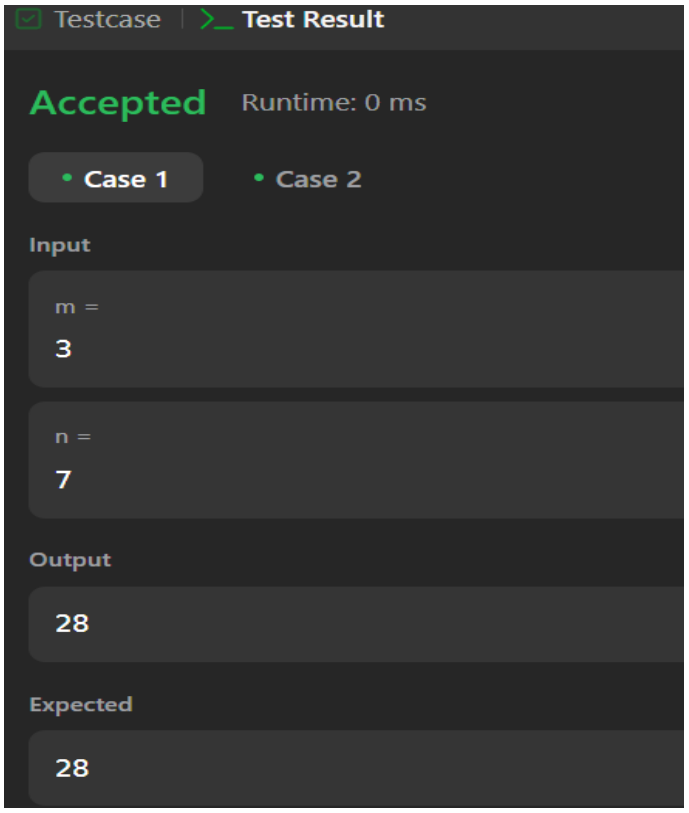
AIM:-

Unique Paths

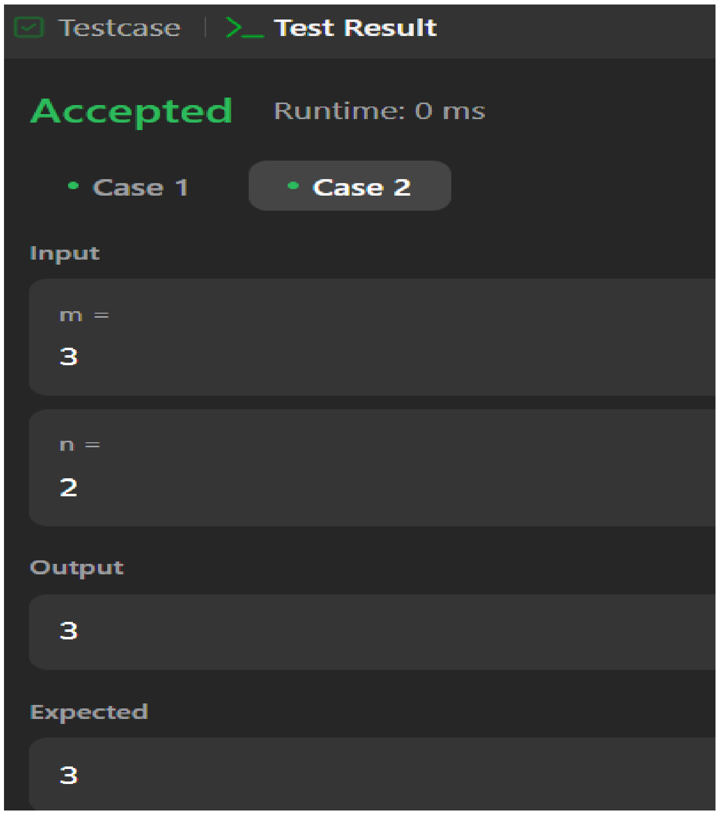
CODE:-

```
class Solution {
    public int uniquePaths(int m, int n) {
        int[] aboveRow = new int[n];
        Arrays.fill(aboveRow, 1);

        for (int row = 1; row < m; row++) {
            int[] currentRow = new int[n];
            Arrays.fill(currentRow, 1);
            for (int col = 1; col < n; col++) {
                currentRow[col] = currentRow[col - 1] + aboveRow[col];
            }
            aboveRow = currentRow;
        }

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

OUTPUT:-



```
Testcase | >_ Test Result
Accepted   Runtime: 0 ms
• Case 1   • Case 2
Input
m =
3
n =
7
Output
28
Expected
28
```



```
Testcase | >_ Test Result
Accepted   Runtime: 0 ms
• Case 1   • Case 2
Input
m =
3
n =
2
Output
3
Expected
3
```

PROBLEM-7

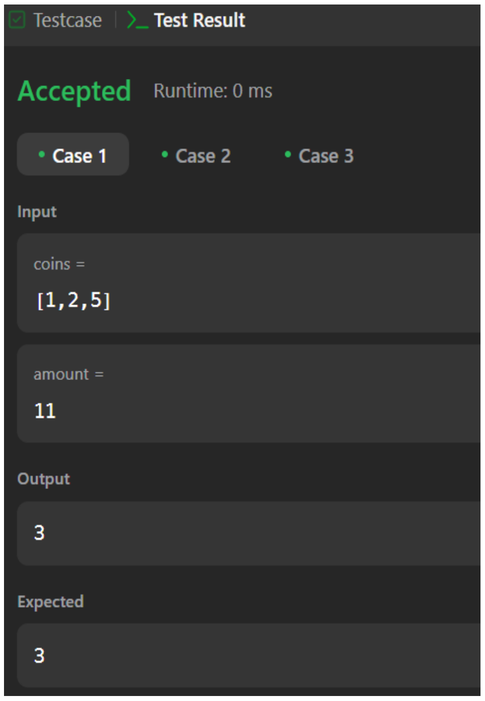
AIM:-

Coin Change

CODE:-

```
class Solution {
    public int coinChange(int[] coins, int amount) {
        int[] minCoins = new int[amount + 1];
        Arrays.fill(minCoins, amount + 1);
        minCoins[0] = 0;

        for (int i = 1; i <= amount; i++) {
            for (int j = 0; j < coins.length; j++) {
                if (i - coins[j] >= 0) {
                    minCoins[i] = Math.min(minCoins[i], 1 + minCoins[i - coins[j]]);
                }
            }
        }

        return minCoins[amount] != amount + 1 ? minCoins[amount] : -1;
    }
}
```

OUTPUT:-

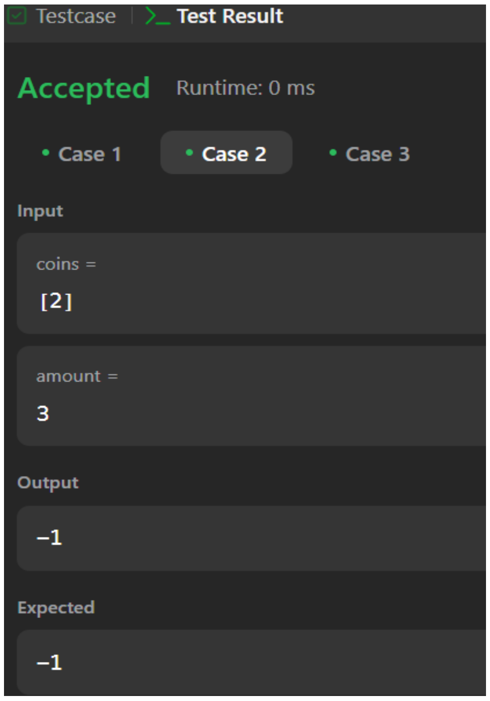


Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

coins =
```
[1,2,5]
```
amount =
```
11
```
Output
```
3
```
Expected
```
3
```

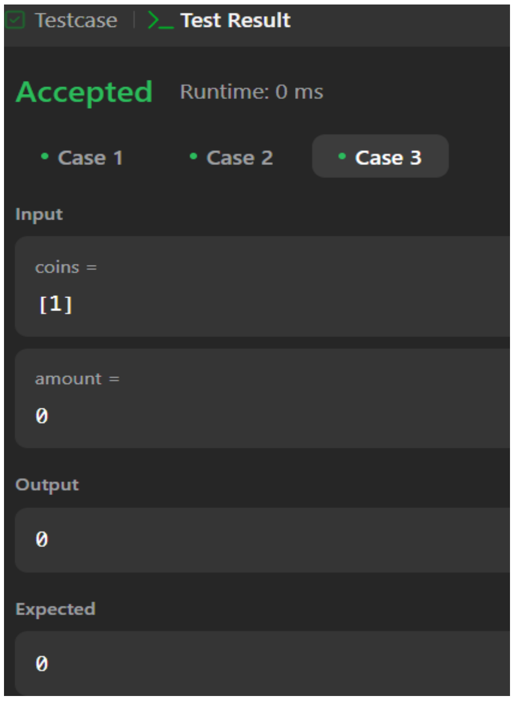


Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

coins =
```
[2]
```
amount =
```
3
```
Output
```
-1
```
Expected
```
-1
```



Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

coins =
```
[1]
```
amount =
```
0
```
Output
```
0
```
Expected
```
0
```