Experiment 7

Student Name: Armaan Siag
UID: 22BET10322
Branch: Information Technology
Section/Group: 22BET_IOT-703/A
Semester: 6th
Subject Code: 22ITP-351

Problem 1

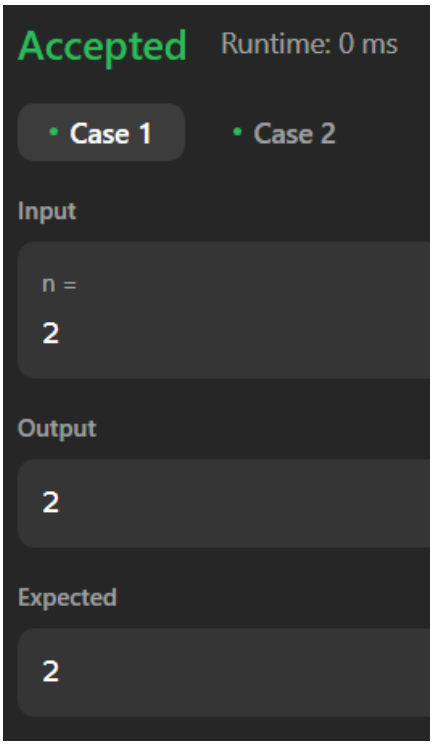
Aim:

Climbing Stairs

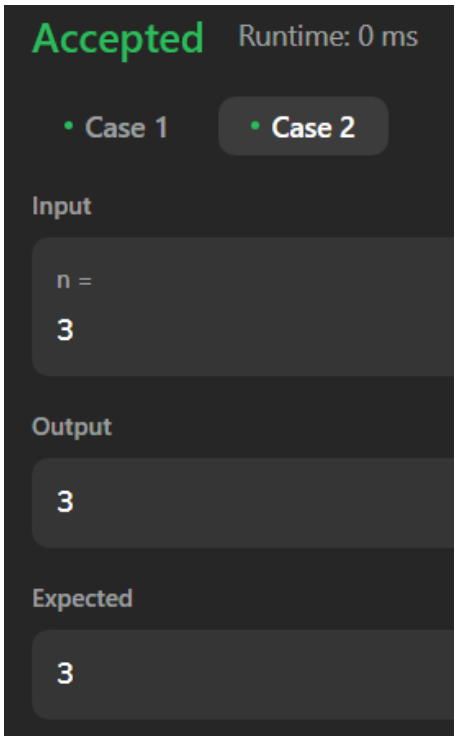
Code:

```
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:



Case 1



Case 2

Problem 2

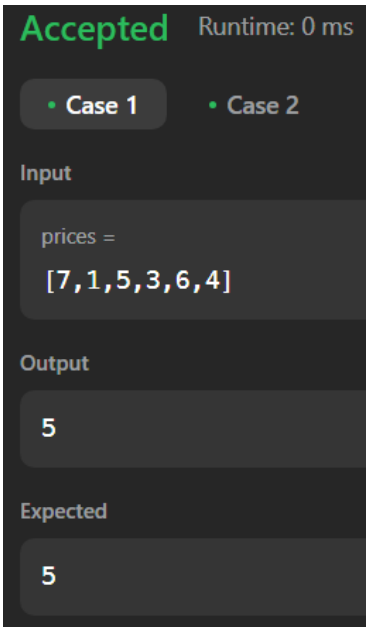
Aim:

Best Time to Buy and Sell a Stock

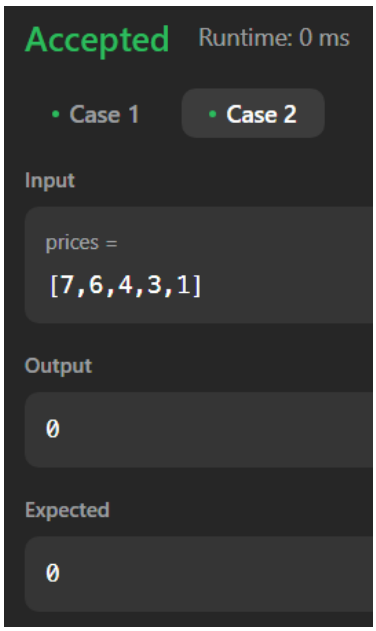
Code:

```cpp
class Solution {
public:
    int maxProfit(std::vector<int>& prices) {
        int buy = prices[0];
        int profit = 0;
        for (int i = 1; i < prices.size(); i++) {
            if (prices[i] < buy) {
                buy = prices[i];
            } else if (prices[i] - buy > profit) {
                profit = prices[i] - buy;
            }
        }
        return profit;
    }
};
```

Output:



Test Case 1



Test Case 2

Problem 3

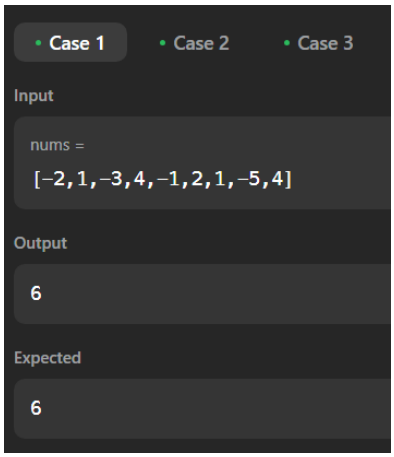
Aim:

Maximum Subarray

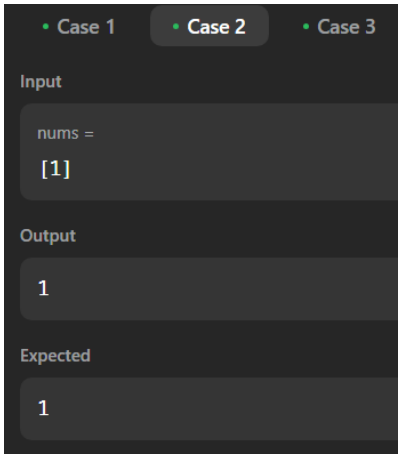
Code:

```cpp
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int res = nums[0];
        int total = 0;

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

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

        return res;
    }
};
```

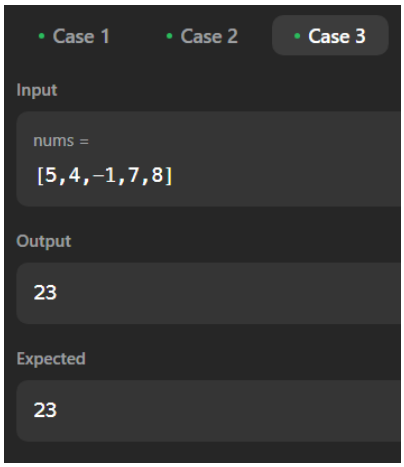
Output:



Case 1



Case 2



Case 3

Problem 4

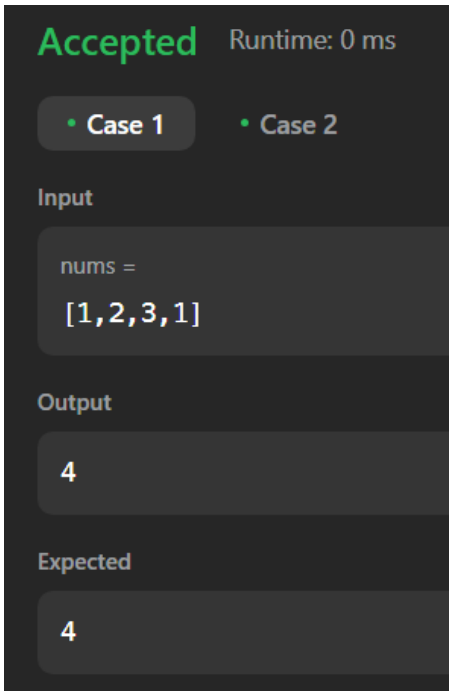
Aim:

House Robber

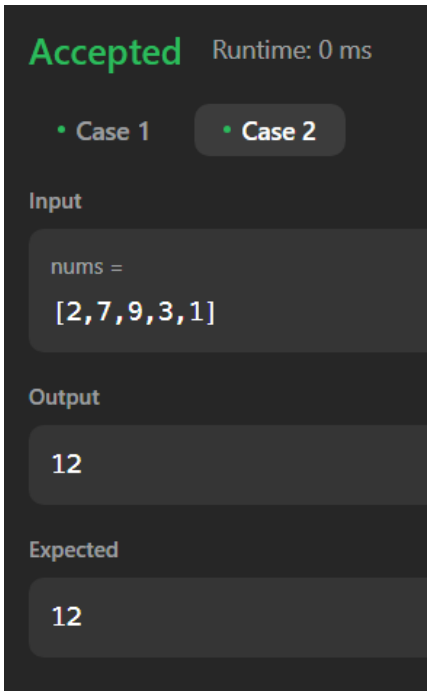
Code:

```cpp
class Solution {
public:
    int rob(vector<int>& nums) {
        int n = nums.size();

        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];
    }
};
```

Output:



Case 1



Case 2

Problem 5

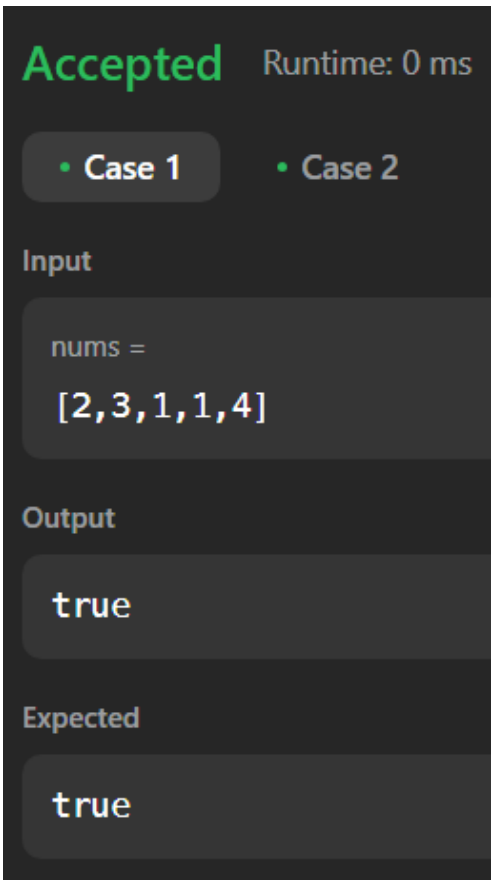
Aim:

Jump Game

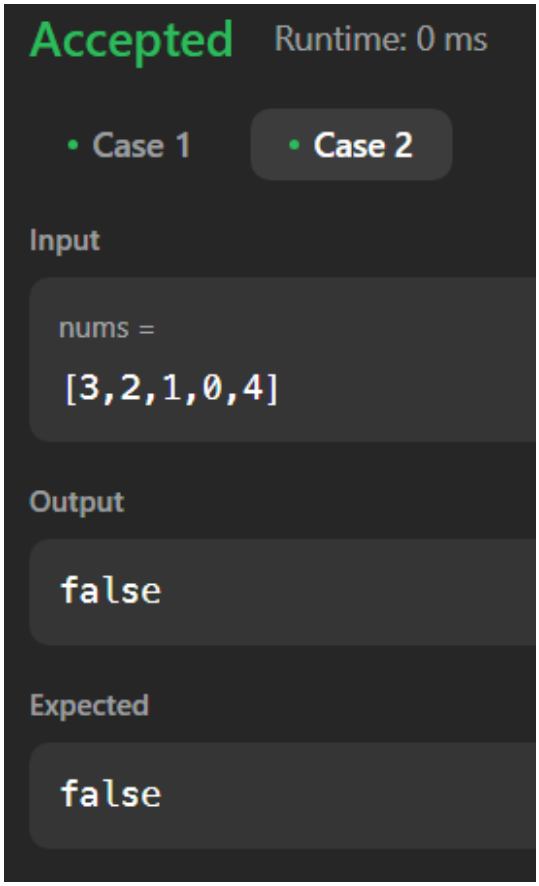
Code:

```cpp
class Solution {
public:
    bool canJump(vector<int>& nums) {
        int goal = nums.size() - 1;

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

        return goal == 0;
    }
};
```

Output:



Case 1



Case 2

Problem 6

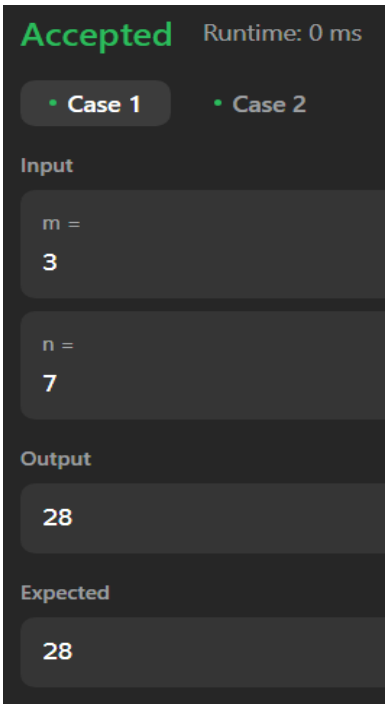
Aim:

Unique Paths

Code:

```
class Solution {
public:
    int uniquePaths(int m, int n, int i = 0, int j = 0) {
        if(i >= m || j >= n) return 0
        if(i == m-1 && j == n-1) return 1;
        return uniquePaths(m, n, i+1, j) + uniquePaths(m, n, i, j+1);
    }
};
```

Output:



Case 1

Case 2

Problem 7

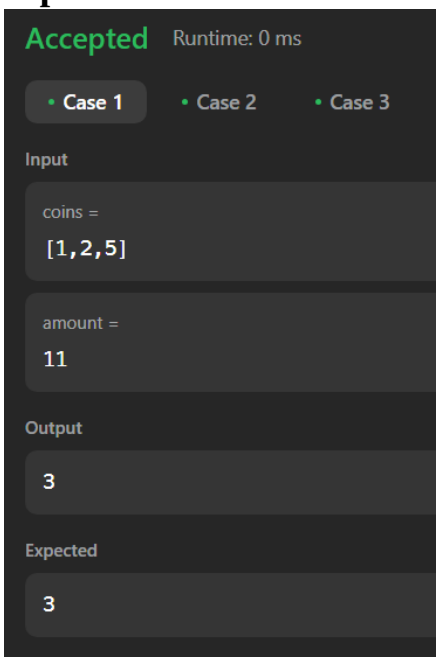
Aim:

Coin Change

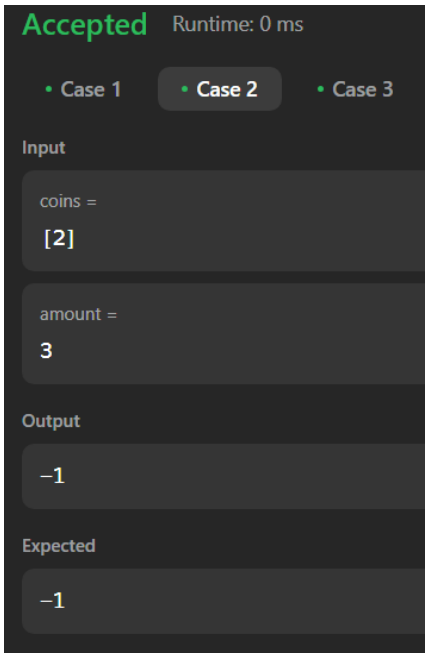
Code:

```cpp
class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {
        vector<int> minCoins(amount + 1, amount + 1);
        minCoins[0] = 0;

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

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

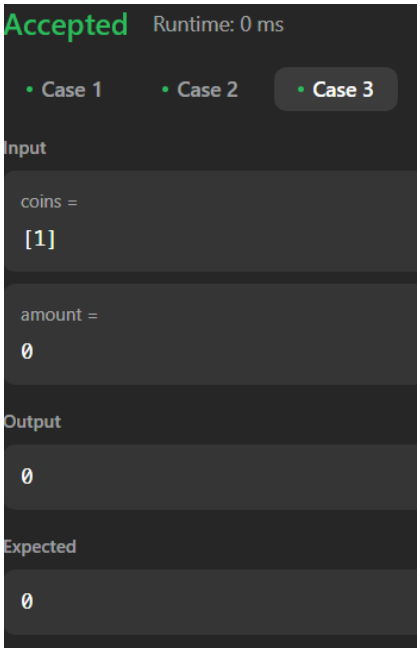
Output:



Case 1



Case 2



Case 3