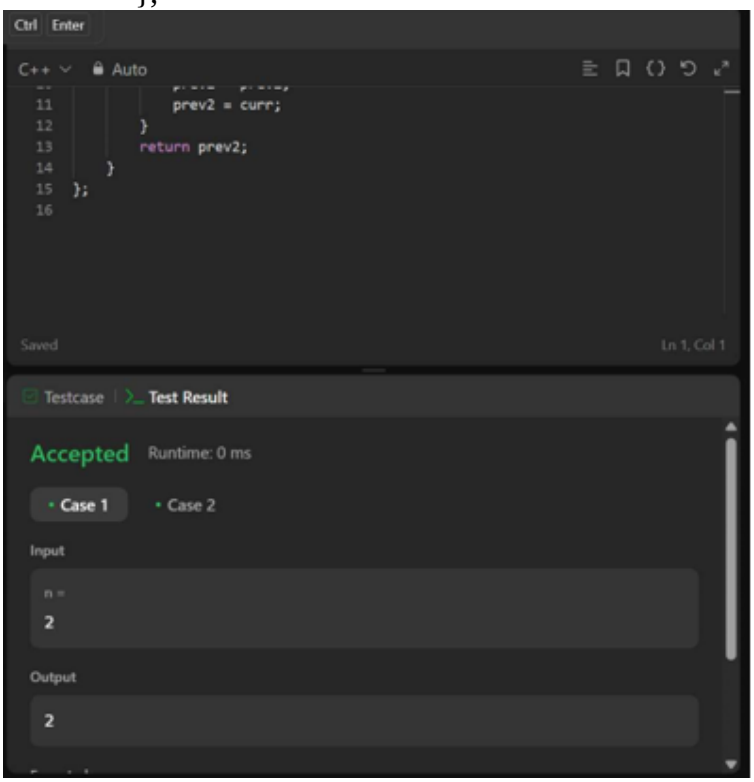
Experiment 07

Student Name: Harshit Shandilya
UID: 22BET10274
Branch: BE-IT
Section/Group: BET_701/A
Semester: 06th
Date of Performance: 21-03-2025
Subject Name: Advanced Programming-II
Subject Code: 22ITP-351

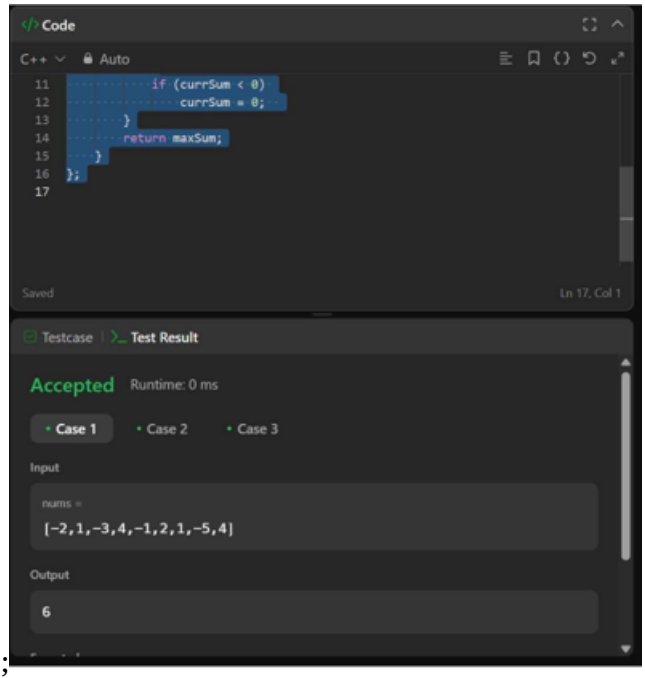
1. **Problem: Climbing Stairs**

```
class Solution {
public:
    int climbStairs(int n) {
        if (n == 1) return 1;
        if (n == 2) return 2;

        int prev1 = 1, prev2 = 2;
        for (int i = 3; i <= n; i++) {
            int curr = prev1 + prev2;
            prev1 = prev2;
            prev2 = curr;
        }
        return prev2;
    }
};
```



2. **Problem: Maximum Subarray**

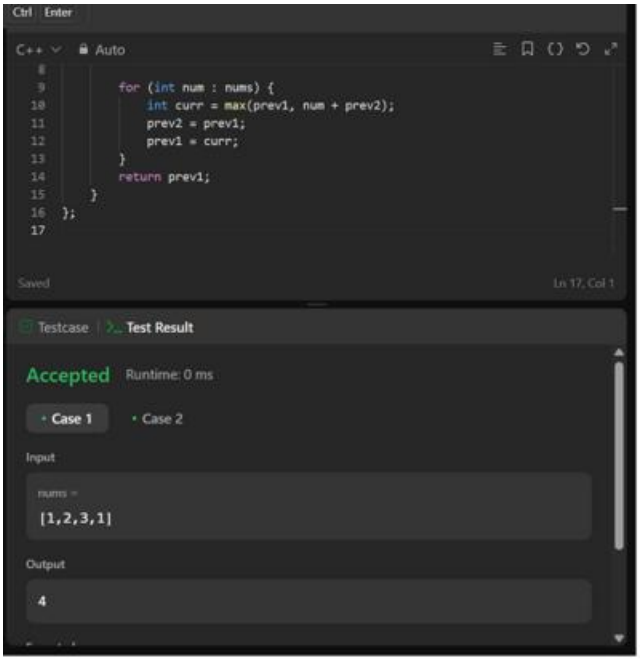
```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int maxSum = nums[0];
        int currSum = 0;
        for (int num : nums) {
            currSum += num;
            maxSum = max(maxSum, currSum);
            if (currSum < 0)
                currSum = 0;
        }
        return maxSum;
    }
```



3. **Problem: House Robber**

```
class Solution {
public:
    int rob(vector<int>& nums) {
        if (nums.empty()) return 0;
```

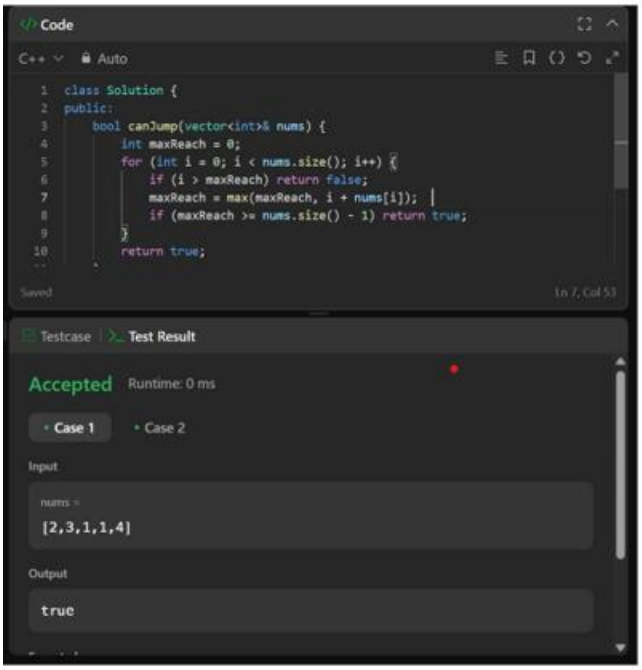
```
        if (nums.size() == 1) return nums[0];
        int prev2 = 0, prev1 = 0; // prev2 -> dp[i-2], prev1 -> dp[i-1]
        for (int num : nums) {
            int curr = max(prev1, num + prev2);
            prev2 = prev1;
            prev1 = curr;
        }
        return prev1;
    }
};
```



4. **Problem: Jump Game**

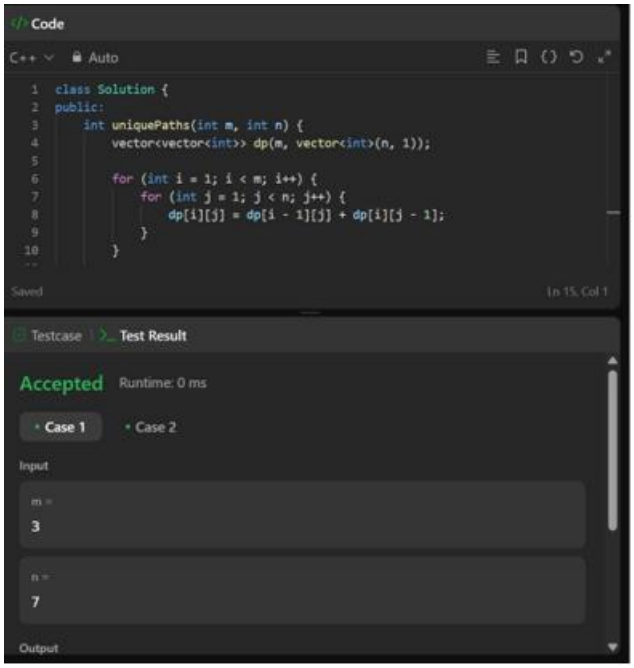
```
class Solution {
public:
    bool canJump(vector<int>& nums) {
        int maxReach = 0;
        for (int i = 0; i < nums.size(); i++) {
            if (i > maxReach) return false;
            maxReach = max(maxReach, i + nums[i]);
            if (maxReach >= nums.size() - 1) return true;
        }
        return true;
```

```
    }
};
```



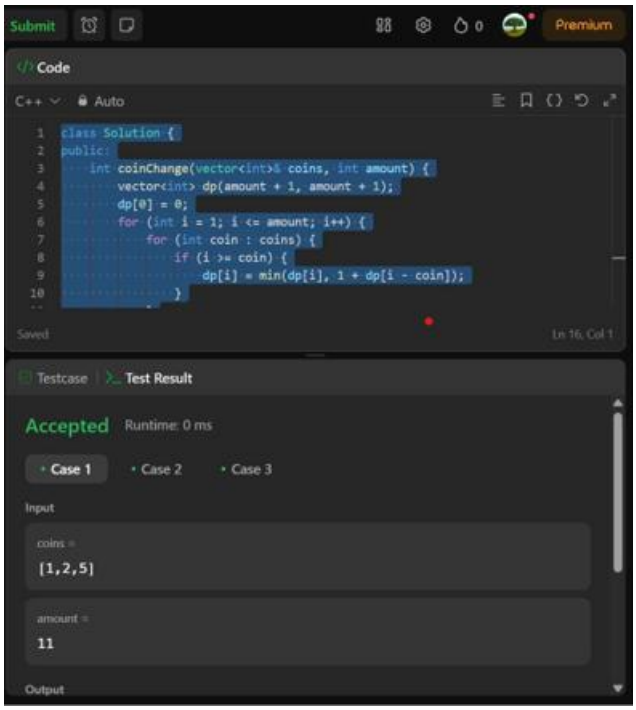
5. **Problem: Unique Paths**

```
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];
    }
};
```



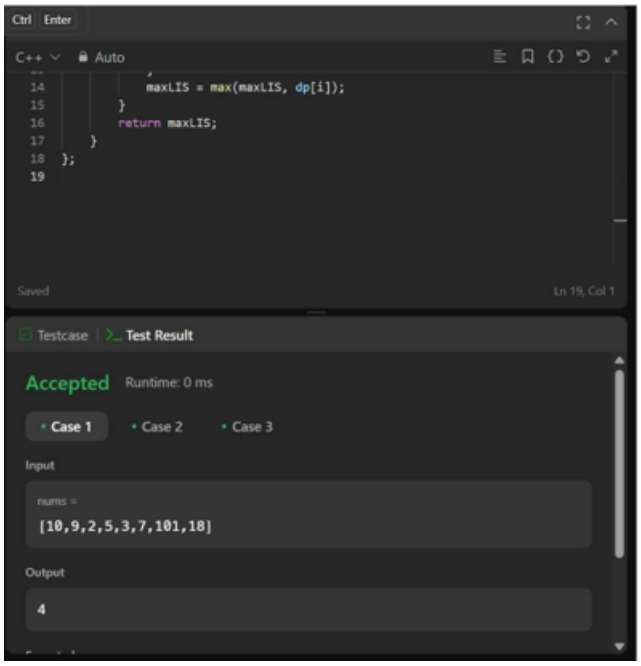
6. **Problem: Coin Change**

```
class Solution {
public:
    int coinChange(vector<int>& coins, int amount) {
        vector<int> dp(amount + 1, amount + 1);
        dp[0] = 0;
        for (int i = 1; i <= amount; i++) {
            for (int coin : coins) {
                if (i >= coin) {
                    dp[i] = min(dp[i], 1 + dp[i - coin]);
                }
            }
        }
        return (dp[amount] == amount + 1) ? -1 : dp[amount];
    }
};
```



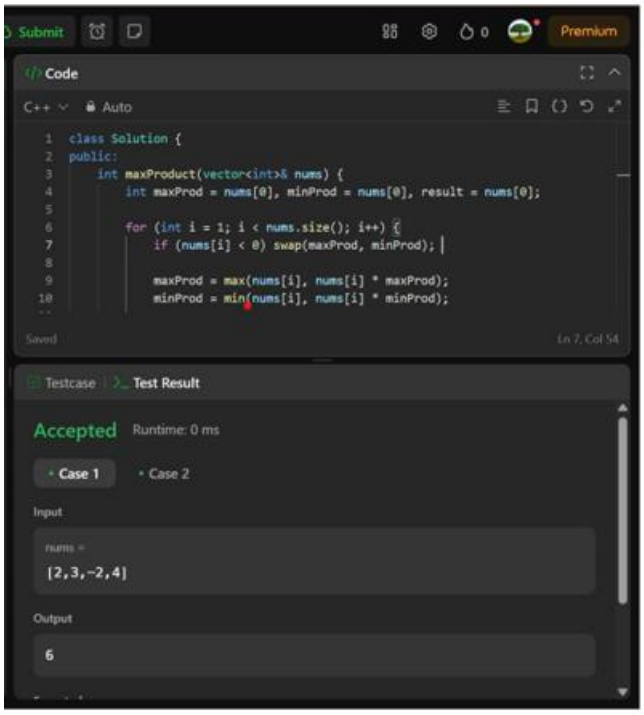
7. **Problem: Longest Increasing Subsequence**

```
class Solution {
public:
    int lengthOfLIS(vector<int>& nums) {
        int n = nums.size();
        vector<int> dp(n, 1);
        int maxLIS = 1;
        for (int i = 1; i < n; i++) {
            for (int j = 0; j < i; j++) {
                if (nums[j] < nums[i]) {
                    dp[i] = max(dp[i], dp[j] + 1);
                }
            }
            maxLIS = max(maxLIS, dp[i]);
        }
        return maxLIS;
    }
};
```



8. **Problem: Maximum Product Subarray**

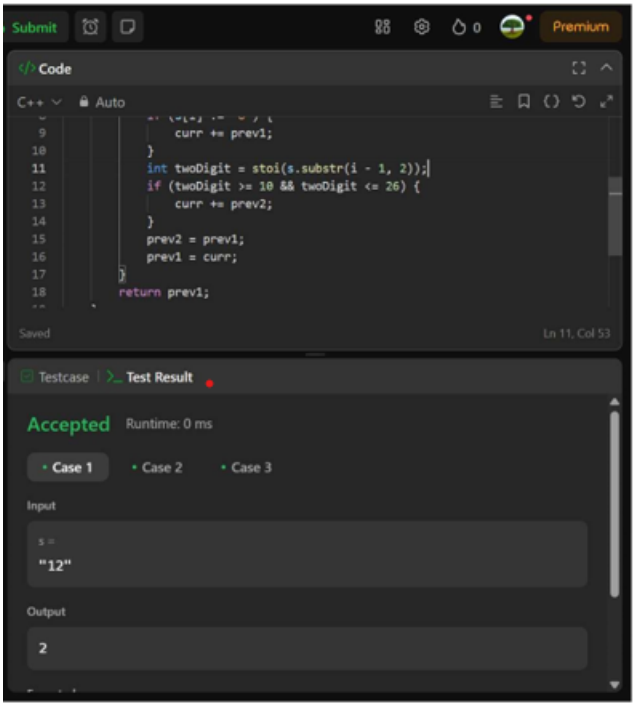
```
class Solution {
public:
    int maxProduct(vector<int>& nums) {
        int maxProd = nums[0], minProd = nums[0], result = nums[0];
        for (int i = 1; i < nums.size(); i++) {
            if (nums[i] < 0) swap(maxProd, minProd);
            maxProd = max(nums[i], nums[i] * maxProd);
            minProd = min(nums[i], nums[i] * minProd);
            result = max(result, maxProd);
        }
        return result;
    }
};
```



9. **Problem: Decode Ways**

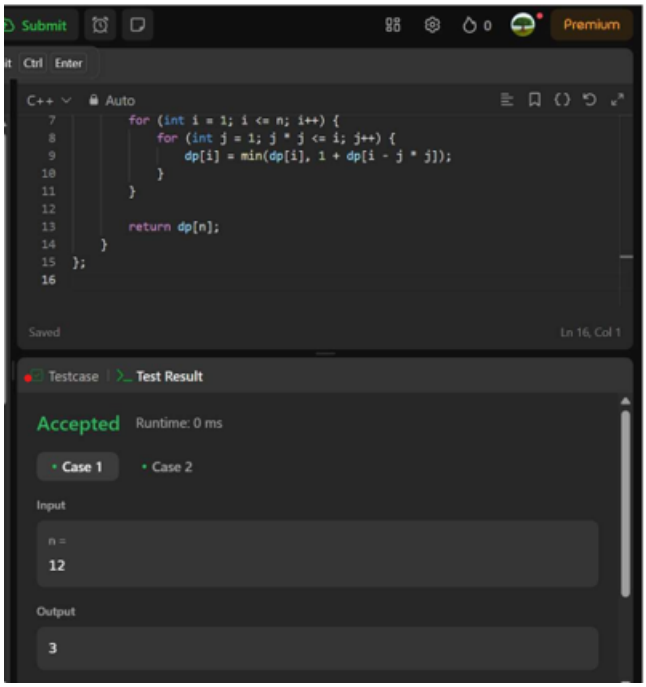
```
class Solution {
public:
    int numDecodings(string s) {
        if (s.empty() || s[0] == '0') return 0;
        int prev1 = 1, prev2 = 1;
        for (int i = 1; i < s.size(); i++) {
            int curr = 0;
            if (s[i] != '0') {
                curr += prev1;
            }
            int twoDigit = stoi(s.substr(i - 1, 2));
            if (twoDigit >= 10 && twoDigit <= 26) {
                curr += prev2;
            }
            prev2 = prev1;
            prev1 = curr;
        }
        return prev1;
    }
```

```
};
```



10. Problem: Perfect squares

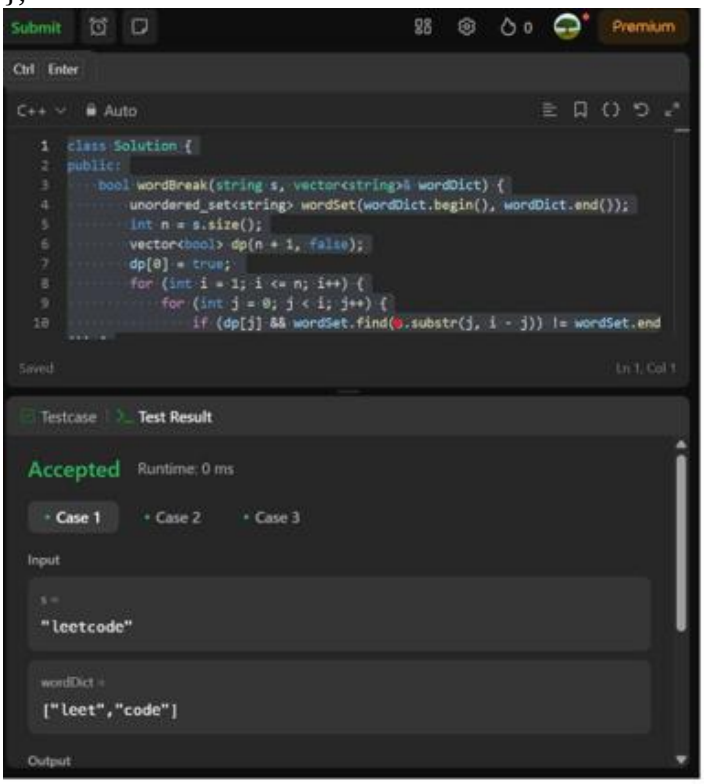
```
class Solution {
public:
    int numSquares(int n) {
        vector<int> dp(n + 1, INT_MAX);
        dp[0] = 0;

        for (int i = 1; i <= n; i++) {
            for (int j = 1; j * j <= i; j++) {
                dp[i] = min(dp[i], 1 + dp[i - j * j]);
            }
        }
        return dp[n];
    }
};
```



11. Problem: Word Break

```
class Solution {
public:
    bool wordBreak(string s, vector<string>& wordDict) {
        unordered_set<string> wordSet(wordDict.begin(), wordDict.end());
        int n = s.size();
        vector<bool> dp(n + 1, false);
        dp[0] = true;
        for (int i = 1; i <= n; i++) {
            for (int j = 0; j < i; j++) {
                if (dp[j] && wordSet.find(s.substr(j, i - j)) != wordSet.end()) {
                    dp[i] = true;
                    break;
                }
            }
        }
        return dp[n];
    }
```

};


Submit
Premium
Ctrl Enter
C++
Auto
class Solution {
public:
bool wordBreak(string s, vector<string>& wordDict) {
unordered_set<string> wordSet(wordDict.begin(), wordDict.end());
int n = s.size();
vector<bool> dp(n + 1, false);
dp[0] = true;
for (int i = 1; i <= n; i++) {
for (int j = 0; j < i; j++) {
if (dp[j] && wordSet.find(s.substr(j, i - j)) != wordSet.end
Saved
Ln 1, Col 1
Testcase
Test Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
s =
"leetcode"
wordDict =
["leet","code"]
Output