CHITTAGONG UNIVERSITY OF ENGINEERING AND TECHNOLOGY

Department of Computer Science & Engineering

Course No: CSE- 244

Course Title:
Algorithm Design & Analysis (Sessional)

Experiment No: 05

Name Of the Experiment: Dynamic Programming.

Identity Details

Name of the Student: Md. Safayat Bin Nasir

Student Id Number: 2004099

Level: 2 Term: 2 Section: B

Group: B1

Date of performance: 08-10-2023

Date of Submission: 05-11-2023

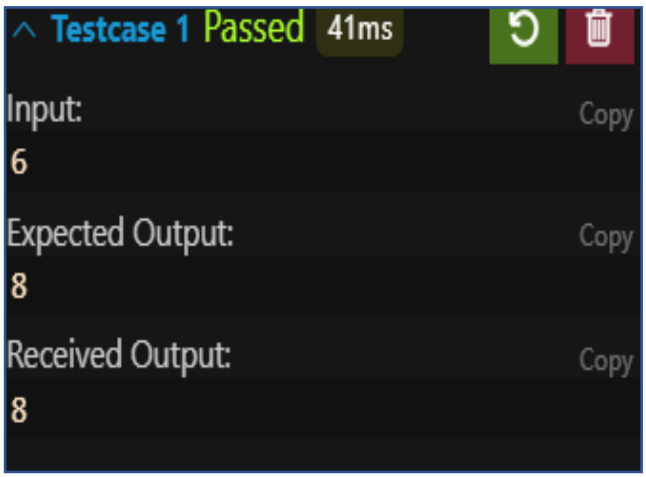
Remarks

Program 1: Calculation of Fibonacci series:

F(0)=0 , F(1)=1 , F(n)=F(n-1)+F(n-2).

```cpp
#include <bits/stdc++.h>
using namespace std;
const int mx = 1e6+5;
long long dp[mx];
long long fibo(long long n ){
    if(n == 0) return 0;
    if(n == 1) return 1;
    if(dp[n] != -1) return dp[n];
    dp[n] = fibo(n-1) + fibo(n-2);
    return dp[n];
}
int main()
{
    long long n ;
    cin >>  n;
    memset(dp , -1 , sizeof(dp));
    cout << fibo(n) << endl;
}
```

Output:



Program 2: Calculation of nCr.

```
#include <bits/stdc++.h>
using namespace std;
long long t = 1;
const int mx = 1e3+5;
long long dp[mx][mx];
long long cal_nCr(long long n , long long r){
    if( r == 0 ) return 1;
    if( r == n ) return 1;
    if(dp[n][r] != 0) return dp[n][r];
    dp[n][r] = cal_nCr(n-1 ,r) + cal_nCr(n-1 , r-1);
    return dp[n][r];
}
int main()
{
    long long n , r;
    cin >> n >> r;
    memset(dp , 0 , sizeof(dp));
    cout << cal_nCr(n , r) << endl;
}
```

Output:

Testcase 1 Passed 26ms

Input: Copy
6 3

Expected Output: Copy
20

Received Output: Copy
20

Testcase 2 Passed 24ms

Input: Copy
8 3

Expected Output: Copy
56

Received Output: Copy
56

Program 3: Implementation of 0/1 knapsack problem. Given the weight and profit of some objects. Pick objects in a way that maximizes the profit and does not exceed a given weight limit.

```
#include <bits/stdc++.h>
using namespace std;
int dp[1000][1000];

int knapsack(int index, vector<pair<int, int>>& v, int weight) {
    if (index < 0) return 0;
    if (dp[index][weight] != -1) return dp[index][weight];
    if (v[index].first > weight) {
        dp[index][weight] = knapsack(index - 1, v, weight);
        return dp[index][weight];
    }
    return max(v[index].second+knapsack(index-1,v,weight-v[index].first),knapsack(index-1,v,weight));
}
int main() {
    int n;
    cin >> n;
    vector<pair<int, int>> v(n);
    for (int i = 0; i < n; i++) {
        cin >> v[i].first >> v[i].second;
    }
    memset(dp, -1, sizeof(dp));
    int weight;
    cin >> weight;
    cout << knapsack(n - 1, v, weight) << endl;
}
```

Output:

Testcase 1 Passed 32ms

Input: Copy

```
5
1 120
4 280
4 200
7 400
3 150
10
```

Expected Output: Copy

```
600
```

Received Output: Copy

```
600
```