****

**NAME: -** Darji Akshatkumar Hiteshbhai

**RollNo: -** 23MCD001

**Branch: -** M.tech-CSE**(Data Science)**

**Subject: -** Complexity Theory & Algorithms

**Practical-10**

**Aim:** Implement 0-1 knapsack problem using dynamic programming.

**Code for 0-1 Knapsack –**

#include<bits/stdc++.h>

using namespace std;

int knapsackProblem(int W, int n, vector<int>& weight, vector<int>& profit) {

    vector<vector<int>> B(n + 1, vector<int>(W + 1, 0));

    for (int i = 1; i <= n; i++) {

        for (int w = 0; w <= W; w++) {

            if (weight[i - 1] <= w) {

                if (profit[i - 1] + B[i - 1][w - weight[i - 1]] > B[i - 1][w]) {

                    B[i][w] = profit[i - 1] + B[i - 1][w - weight[i - 1]];

                } else {

                    B[i][w] = B[i - 1][w];

                }

            } else {

                B[i][w] = B[i - 1][w];

            }

        }

    }

    cout << "DP Table is as follows:\n";

    for (int i = 0; i <= n; i++) {

        for (int w = 0; w <= W; ++w) {

            cout << B[i][w] << " ";

        }

        cout << "\n";

    }

    int i = n, w = W;

    cout << "\nSelected items:\n";

    while (i > 0 && w > 0) {

        if (B[i][w] != B[i - 1][w]) {

            cout << "Item " << i << " (Weight: " << weight[i - 1] << ", Value: " << profit[i - 1] << ")\n";

            w -= weight[i - 1];

        }

        i--;

    }

    return B[n][W];

}

int main() {

    int W;

    int n;

    cout << "Enter the Knapsack Capacity W: ";

    cin >> W;

    cout << endl << "Enter the No. of items n: ";

    cin >> n;

    vector<int> weight(n);

    vector<int> profit(n);

    cout << endl << "Enter [Weight,Profit]";

    for(int i=0;i<n;i++){

        cin >> weight[i];

        cin >> profit[i];

    }

    int maxProfit = knapsackProblem(W, n, weight, profit);

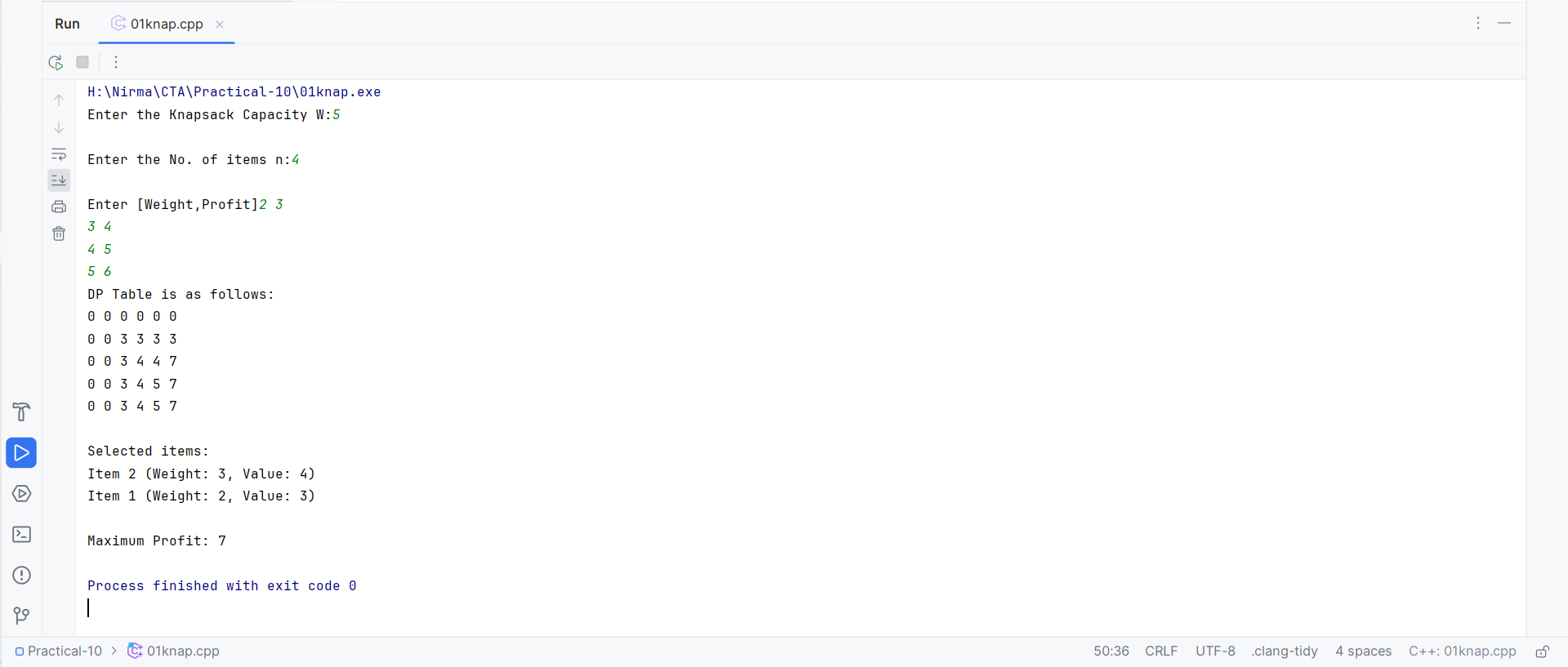
    cout << "\nMaximum Profit: " << maxProfit << endl;

    return 0;

}

**Output –**

**Test Case – 1**

****