	Marwari University **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing 0/1 Knapsack Problem using Dynamic Programming Approach	
Experiment No: 09	**Date:**	**Enrollment No: 92200133030**

Aim: Implementing 0/1 Knapsack Problem using Dynamic Programming Approach

IDE: Visual Studio Code

0/1 Knapsack Problem Using Dynamic Programming

Theory: -

- The 0/1 Knapsack Problem is a classic optimization problem in computer science and operations research. It involves selecting items with given weights and values to maximize the total value while staying within a weight limit.

1. Problem Definition

- You are given:
  1. A set of n items, where each item has:
     - A weight w[i]
     - A value v[i]
  2. A knapsack with a maximum weight capacity W.

- The task is to determine the maximum value you can achieve by selecting a subset of items such that the total weight does not exceed W. Each item can either be included once (1) or not included at all (0), which is why it is called the 0/1 Knapsack Problem.

2. Dynamic Programming Approach :-

- The problem has overlapping subproblems and optimal substructure properties, making it suitable for a **dynamic programming (DP)** solution.

1. **Define the DP Table**: Let dp[i][w] represent the maximum value that can be obtained using the first iii items with a weight limit of www.
2. **Recurrence Relation**:
   For each item i, we have two choices:
   a. **Exclude the item**:
      The maximum value remains the same as for the previous item with the same weight limit.
      dp[i][w]=dp[i−1][w]
   b. **Include the item (if the weight allows)**:
      The value is the item's value plus the maximum value achievable with the remaining weight.
      dp[i][w]=v[i−1]+dp[i−1][w−w[i−1]]

The final value is the maximum of these two choices:
dp[i][w]=max(dp[i−1][w],v[i−1]+dp[i−1][w−w[i−1]])

3. **Initialization**:
   dp[0][w]= 0 for all w, since no items mean no value.
   dp[i][0]= 0 for all iii, since a knapsack with 0 capacity can hold no value.
4. **Final Result**:
   The value at dp[n][W] gives the maximum value that can be achieved for the given weight limit W.

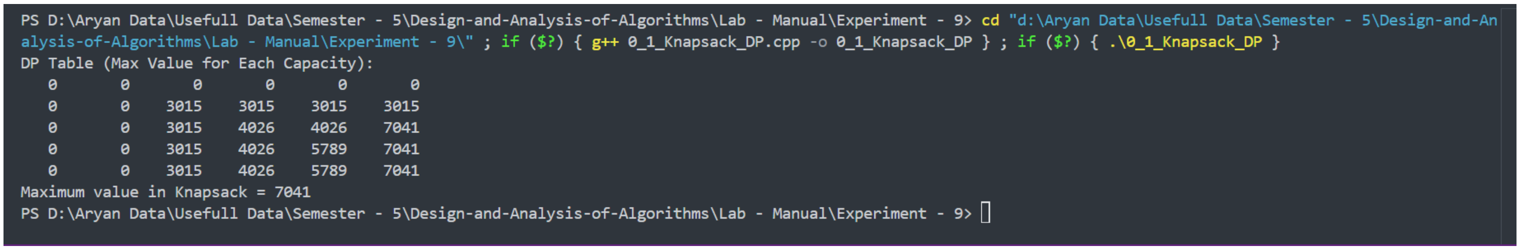
Algorithm: -

Programming Language: - C++

Code :-

```
#include<bits/stdc++.h>
using namespace std;

int knapsack(int weights[], int profits[], int n, int capacity) {
    vector<vector<int>> dp(n + 1, vector<int>(capacity + 1, 0));
    for (int i = 1; i <= n; i++) {
        for (int w = 1; w <= capacity; w++) {
            if (weights[i - 1] <= w) {
                dp[i][w] = max(dp[i - 1][w], profits[i - 1] + dp[i - 1][w - weights[i - 1]]);
            }
            else {
                dp[i][w] = dp[i - 1][w];
            }
        }
    }

    cout << "DP Table (Max Value for Each Capacity):\n";
    for (int i = 0; i <= n; i++) {
        for (int w = 0; w <= capacity; w++) {
            cout << setw(4) << dp[i][w] << "\t";
        }

        cout << endl;
    }

    return dp[n][capacity];
}

int main() {
    int weights[] = { 2, 3, 4, 5 };
    int profits[] = { 3015, 4026, 5789, 6147 };
    int capacity = 5;
    int n = sizeof(weights) / sizeof(weights[0]);

    int max_profit = knapsack(weights, profits, n, capacity);

    cout << "Maximum value in Knapsack = " << max_profit << endl;

    return 0;
 }
```

Output :-



```
PS D:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 9> cd "d:\Aryan Data\Usefull Data\Semester - 5\Design-and-An
alysis-of-Algorithms\Lab - Manual\Experiment - 9\" ; if ($?) { g++ 0_1_Knapsack_DP.cpp -o 0_1_Knapsack_DP } ; if ($?) { .\0_1_Knapsack_DP }
DP Table (Max Value for Each Capacity):
   0       0       0       0       0       0
   0       0    3015    3015    3015    3015
   0       0    3015    4026    4026    7041
   0       0    3015    4026    5789    7041
   0       0    3015    4026    5789    7041
Maximum value in Knapsack = 7041
PS D:\Aryan Data\Usefull Data\Semester - 5\Design-and-Analysis-of-Algorithms\Lab - Manual\Experiment - 9>
```

	Marwari University **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: Design and Analysis of Algorithms (01CT0512)	**Aim:** Implementing 0/1 Knapsack Problem using Dynamic Programming Approach	
Experiment No: 09	**Date:**	**Enrollment No: 92200133030**

Space Complexity:- __________
Justification: -

Time Complexity:

Best Case Time Complexity: __________
Justification: -

Worst Case Time Complexity:- ____________
Justification: -

Conclusion:-