

**CHANDIGARH UNIVERSITY  
UNIVERSITY INSTITUTE OF ENGINEERING  
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



<b>Submitted By:</b> Vivek Kumar(21BCS8129)		<b>Submitted To:</b> Neha Dutta(E12830)	
<b>Subject Name</b>	Design and Analysis of Algorithm Lab		
<b>Subject Code</b>	20CSP-312		
<b>Branch</b>	Computer Science and Engineering		
<b>Semester</b>	5 <sup>th</sup>		

## Experiment - 7

**Student Name: Vivek Kumar**

**Branch: BE-CSE(LEET)**

**Semester: 5<sup>th</sup>**

**Subject Name: DAA Lab**

**UID: 21BCS8129**

**Section/Group: 20BCS-WM-616/A**

**Date of Performance: 31/10/2022**

**Subject Code: 20CSP-312**

### 1. Aim/Overview of the practical:

Code to implement 0-1 Knapsack using Dynamic Programming

### 2. Task to be done/ Which logistics used:

Write a program to implement 0-1 Knapsack using the dynamic programming.

### 3. Requirements (For programming-based labs):

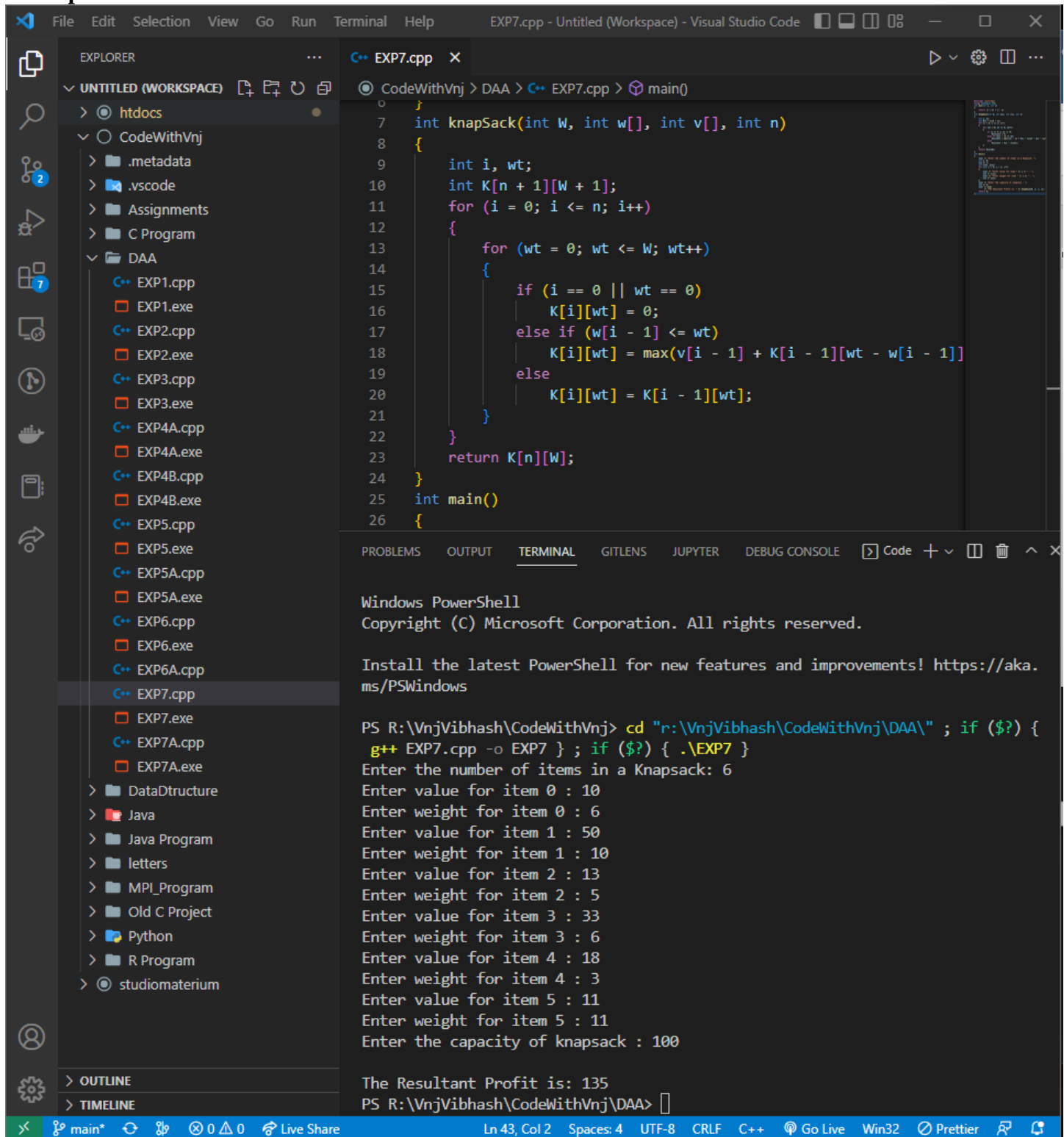
- Laptop or PC.
- Operation system (Mac, Windows, Linux, or any)
- Vs-Code with MinGw or any C++ Compiler

### 4. Steps for experiment/practical/Code:

```
#include <iostream>
#include <iostream>
using namespace std;
int max(int x, int y)
{
    return (x > y) ? x : y;
}
int knapSack(int W, int w[], int v[], int n)
{
    int i, wt;
    int K[n + 1][W + 1];
    for (i = 0; i <= n; i++)
    {
        for (wt = 0; wt <= W; wt++)
        {
            if (i == 0 || wt == 0)
                K[i][wt] = 0;
            else if (w[i - 1] <= wt)
                K[i][wt] = max(v[i - 1] + K[i - 1][wt - w[i - 1]], K[i - 1][wt]);
            else
                K[i][wt] = K[i - 1][wt];
        }
    }
    return K[n][W];
}
```

```
int main()
{
    cout << "Enter the number of items in a Knapsack: ";
    int n, W;
    cin >> n;
    int v[n], w[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Enter value for item " << i << " : ";
        cin >> v[i];
        cout << "Enter weight for item " << i << " : ";
        cin >> w[i];
    }
    cout << "Enter the capacity of knapsack : ";
    cin >> W;
    cout << endl
        << "The Resultant Profit is: " << knapSack(W, w, v, n);
    return 0;
}
```

## 5. Output:



The screenshot displays the Visual Studio Code interface with a C++ file named `EXP7.cpp` open. The code implements a dynamic programming solution for the 0-1 Knapsack problem. The `main` function prompts the user to enter the number of items, their weights, and values, and finally the knapsack capacity. The output in the terminal shows the execution of the program with the following inputs: 6 items, weights [10, 50, 10, 33, 18, 11], values [6, 10, 13, 6, 3, 11], and a capacity of 100. The resultant profit is calculated as 135.

```

7  int knapSack(int W, int w[], int v[], int n)
8  {
9      int i, wt;
10     int K[n + 1][W + 1];
11     for (i = 0; i <= n; i++)
12     {
13         for (wt = 0; wt <= W; wt++)
14         {
15             if (i == 0 || wt == 0)
16                 K[i][wt] = 0;
17             else if (w[i - 1] <= wt)
18                 K[i][wt] = max(v[i - 1] + K[i - 1][wt - w[i - 1]]
19                               , K[i - 1][wt]);
20             else
21                 K[i][wt] = K[i - 1][wt];
22         }
23     }
24     return K[n][W];
25 }
26 int main()
27 {
28     int n, W;
29     cout << "Enter the number of items in a Knapsack: ";
30     cin >> n;
31     int w[n], v[n];
32     for (int i = 0; i < n; i++)
33     {
34         cout << "Enter value for item " << i << " : ";
35         cin >> v[i];
36         cout << "Enter weight for item " << i << " : ";
37         cin >> w[i];
38     }
39     cout << "Enter the capacity of knapsack : ";
40     cin >> W;
41     int ans = knapSack(W, w, v, n);
42     cout << "The Resultant Profit is: " << ans << endl;
43 }
```

Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! <https://aka.ms/PSWindows>

PS R:\VnjVibhash\CodeWithVnj> cd "r:\VnjVibhash\CodeWithVnj\DAA" ; if (\$?) { g++ EXP7.cpp -o EXP7 } ; if (\$?) { .\EXP7 }

Enter the number of items in a Knapsack: 6  
Enter value for item 0 : 10  
Enter weight for item 0 : 6  
Enter value for item 1 : 50  
Enter weight for item 1 : 10  
Enter value for item 2 : 13  
Enter weight for item 2 : 5  
Enter value for item 3 : 33  
Enter weight for item 3 : 6  
Enter value for item 4 : 18  
Enter weight for item 4 : 3  
Enter value for item 5 : 11  
Enter weight for item 5 : 11  
Enter the capacity of knapsack : 100

The Resultant Profit is: 135  
PS R:\VnjVibhash\CodeWithVnj\DAA>

## Learning outcomes (What I have learnt):

1. How to solve the 0-1 Knapsack using dynamic programming.

**Evaluation Grid (To be created per the faculty's SOP and Assessment guidelines):**

Sr. No.	Parameters	Marks Obtained	Maximum Marks
1.	Worksheet completion including writing learning objectives/Outcomes. (To be submitted at the end of the day).		
2.	Post-Lab Quiz Result.		
3.	Student Engagement in Simulation/Demonstration/Performance and Controls/Pre-Lab Questions.		
	Signature of Faculty (with Date):	Total Marks Obtained:	