****

Assignment No: 5(DAA)

Title Name: Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy.

Name: **Aniket Rajani**

Class :**B.E** **Div**:A **Batch**:**A**

Roll No:405A008

****

Program

```
#include<iostream>
using namespace std;
int v[20][20];
int max1(int a,int b)
{
return(a>b)?a:b;
}
int main() {
int i,j,p[20],w[20],n,max;
cout<<"\n enter the number of items\n ";
cin >> n;
for(i=1;i<=n;i++)
{
cout << "\n enter the weight and profit of the item "<<i <<":";
cin>> w[i] >> p[i];
}
cout<< "\n enter the capacity of the knapsack: ";
cin>> max;
```

```
for(i=0;i<=n;i++)
v[i][0]=0;
for(j=0;j<=max;j++)
v[0][j]=0;
for(i=1;i<=n;i++)
for(j=1;j<=max;j++)
{
if(w[i]>j)
v[i][j]=v[i-1][j];
else
v[i][j]=max1(v[i-1][j],v[i-1][j-w[i]]+p[i]);
}
cout<<"\n\nThe table is\n";
for(i=0;i<=n;i++)
{
for(j=0;j<=max;j++)
cout<<"\t"<< v[i][j];
cout<<"\n";
}
cout<<"\nThe maximum profit is = "<< v[n][max];
cout<<"\nThe most valuable subset is:{";
j=max;
for(i=n;i>=1;i--)
if(v[i][j]!=v[i-1][j])
{
```

```
cout<<"\t item: "<< i;j=j-w[i];
}
cout<<" }";
}
```

OUTPUT

```
enter the number of items 4
enter the weight and profit of the item 1:2 12
enter the weight and profit of the item 2:1 10
enter the weight and profit of the item 3:3 20
enter the weight and profit of the item 4:2 15
enter the capacity of the knapsack5
The table is
0    0    0    0    0
0    0   10  10  10
0    12  12  12  15
0    12   22  22  25
0    12   22 30  30
0    12   22  32  37
The maximum profit is= 37
The most valuable subset is :{
item 4:
item 2:
item 1 :}
```