Nikit Gokhe
Comp D1
Roll no. 324022
GR no. 21810522

ASSIGNMENT

0/1 KNAPSACK PROBLEM (BRANCH AND BOUND ALGORITHM)

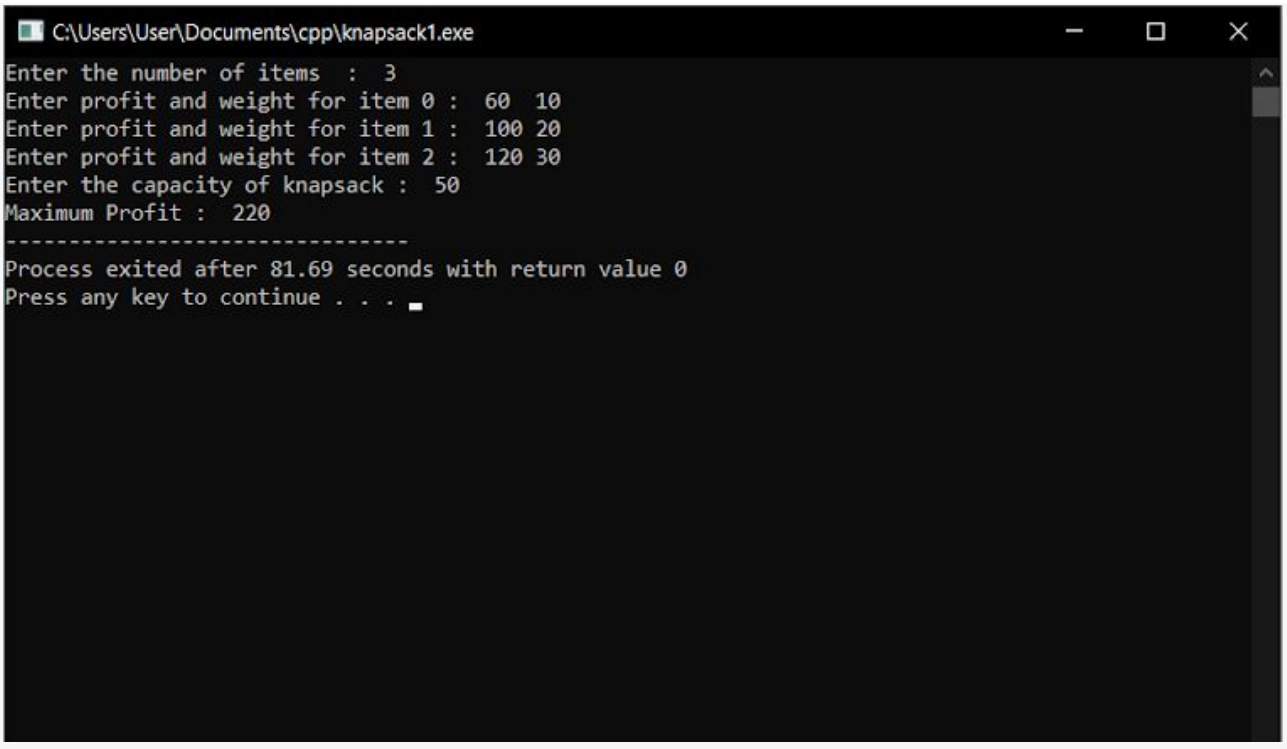
SOURCE CODE:

```
#include<stdio.h>
#include<conio.h>
#include<iostream>

using namespace std;

// A utility function that returns maximum of two integers
int max(int a, int b)
{
    return (a > b) ? a : b;
}

// Returns the maximum value that can be put in a knapsack of capacity W
int knapSack(int W, int wt[], int val[], int n)
{

    if (n == 0 || W == 0)
        return 0;


    if (wt[n - 1] > W)
        return knapSack(W, wt, val, n - 1);


    else
        return max(val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1), knapSack(W, wt,
val, n - 1));
}
```

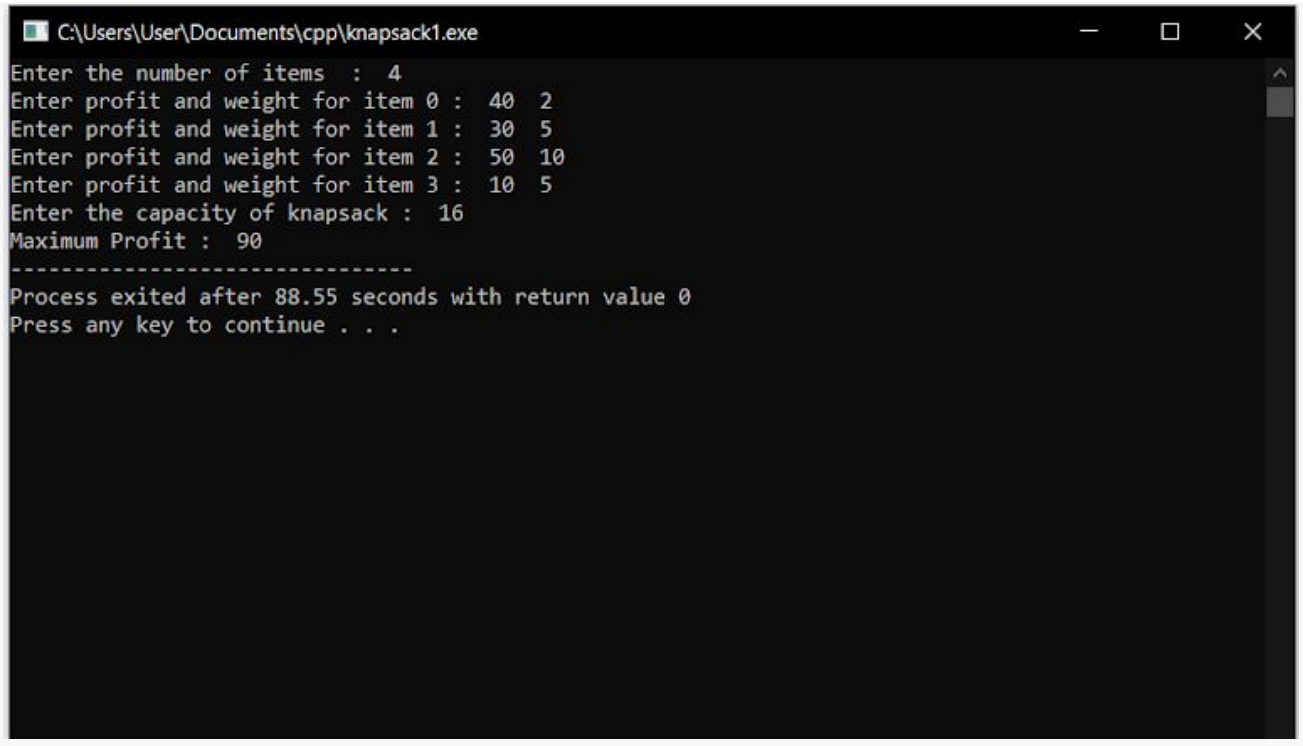
```
// Driver program to test above function
int main()
{
    cout << "Enter the number of items  :  ";
    int n, W;
    cin >> n;
    int val[n], wt[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Enter profit and weight for item " << i << " :  ";
        cin >> val[i];
        cin >> wt[i];
    }

    cout << "Enter the capacity of knapsack :  ";
    cin >> W;
    cout <<"Maximum Profit :  "<< knapSack(W, wt, val, n);

    return 0;
}
```

OUTPUT :

1.


C:\Users\User\Documents\cpp\knapsack1.exe
Enter the number of items : 3
Enter profit and weight for item 0 : 60 10
Enter profit and weight for item 1 : 100 20
Enter profit and weight for item 2 : 120 30
Enter the capacity of knapsack : 50
Maximum Profit : 220
Process exited after 81.69 seconds with return value 0
Press any key to continue . . .

2.


C:\Users\User\Documents\cpp\knapsack1.exe
Enter the number of items : 4
Enter profit and weight for item 0 : 40 2
Enter profit and weight for item 1 : 30 5
Enter profit and weight for item 2 : 50 10
Enter profit and weight for item 3 : 10 5
Enter the capacity of knapsack : 16
Maximum Profit : 90
Process exited after 88.55 seconds with return value 0
Press any key to continue . . .