Program:

```
#include <bits/stdc++.h>
using namespace std;
class Sack
{
public:
   int price;
public:
   int weight;
public:
   float ratio;
};

int max(int a, int b) { return (a > b) ? a : b; }
int knapSack(Sack item[], int capacity, int n) // capacity, Sack item[], int n
{
   if (n == 0 || capacity == 0)
      return 0;

   if (item[n - 1].weight > capacity)
      return knapSack(item,capacity, n - 1);
   else
      return max(
         item[n - 1].price + knapSack(item, capacity- item[n - 1].weight, n - 1),
         knapSack(item, capacity, n - 1));
}

void mergeTwoSortedArr(Sack arr[], int begin, int mid, int end)
{
   int n1 = mid - begin + 1;
   int n2 = end - mid;
   int i = 0, j = 0, k = begin;
   Sack a[n1], b[n2];
   while (i < n1)
   {
      a[i] = arr[k];
      i++;
      k++;
   }
   while (j < n2)
   {
      b[j] = arr[k];
      j++;
      k++;
   }
```

```
    i = 0, j = 0, k = begin;
    while (i < n1 && j < n2)
    {
        if (a[i].ratio < b[j].ratio)
        {
            arr[k] = a[i];
            i++;
            k++;
        }
        else
        {
            arr[k] = b[j];
            j++;
            k++;
        }
    }

    while (i < n1)
    {
        arr[k] = a[i];
        i++;
        k++;
    }

    while (j < n2)
    {
        arr[k] = b[j];
        j++;
        k++;
    }
}

void mergeSort(Sack arr[], int begin, int end)
{
    if (begin < end)
    {
        int mid = (begin + end) / 2;
        mergeSort(arr, begin, mid);
        mergeSort(arr, mid + 1, end);
        mergeTwoSortedArr(arr, begin, mid, end);
    }
}

void printSackItems(Sack item[], int n)
{
    cout << "======================================================" << endl;
    cout << "Price:\t";
    for (int i = 0; i < n; i++)
    cout << item[i].price << "\t";
    cout << endl;
    cout << "Weight:\t";
    for (int i = 0; i < n; i++)
        cout << item[i].weight << "\t";
    cout << endl;
    cout << "Ratio:\t";
    for (int i = 0; i < n; i++)
        cout << item[i].ratio << "\t";
    cout << endl;
    cout << "===================================================" << endl;
}

float fractionalKnapsack(Sack item[], int capacity, int n)
{
    float maxProfit = 0.0;
    float fullyUsedElements[n];
    for (int i = 0; i < n; i++)
    {
        if (capacity - item[i].weight >= 0)
        {
            capacity = capacity - item[i].weight;
            maxProfit = maxProfit + item[i].price;
            fullyUsedElements[i] = 1;
        }
        else
        {
            maxProfit = maxProfit + (capacity / item[i].weight) * item[i].price;
            fullyUsedElements[i] = capacity / item[i].weight;
            break;
        }
    }
    return maxProfit;
}

int main()
{
    int n, capacity;
    cout << "Enter the Capacity of the Sack: ";
    cin >> capacity;
    cout << "Enter the N Item of Array: ";
    cin >> n;
    // int arr[n];
    Sack item[n];
    for (int i = 0; i < n; i++)
    {
        cout << "===========================================" << endl;
        cout << "Enter Price of Item no. " << i + 1 << ": ";
        cin >> item[i].price;
        cout << "Enter Weight of Item no." << i + 1 << ": ";
        cin >> item[i].weight;
        item[i].ratio = item[i].price / item[i].weight;
    }
    cout << "Enter your Choice:\n1. Fractional Knapsack\n2. 0/1 Knapsack\n";
    int choice;
    cin >> choice;

    if(choice == 1)
    {
        cout << "\nBefore Sorting Sack Items: " << endl;
        printSackItems(item, n);
        mergeSort(item, 0, n - 1);
        cout << endl
             << "After Sorting Sack Items: " << endl;
        printSackItems(item, n);
        float maxProfit = fractionalKnapsack(item, capacity, n);
        cout << endl
             << "Maximum Profit is: " << maxProfit;
    }
    else if(choice == 2){
        int knapSackValue = knapSack(item, capacity, n);
        cout << "Maximum Profit: " << knapSackValue;
    }
    return 0;
}
```

Output:

Fractional Knapsack Output:

```
PS H:\STUDY\College\DAA Lab> cd "h:\STUDY\College\DAA Lab\" ; if ($?) { g++
fractionalKnapsack.cpp
-o fractionalKnapsack } ; if ($?) { .\fractionalKnapsack }
Enter the Capacity of the Sack: 60
Enter the N Item of Array: 5
=====================================================
Enter Price of Item no. 1: 15
Enter Weight of Item no.1: 10
=====================================================
Enter Price of Item no. 2: 20
Enter Weight of Item no.2: 20
=====================================================
Enter Price of Item no. 3: 37
Enter Weight of Item no.3: 36
=====================================================
Enter Price of Item no. 4: 44
Enter Weight of Item no.4: 50
=====================================================
Enter Price of Item no. 5: 23
Enter Weight of Item no.5: 60

Enter your Choice:
1. Fractional Knapsack
2. 0/1 Knapsack
1

Before Sorting Sack Items:
=====================================================
Price:   15    20    37    44    23
Weight: 10    20    36    50    60
Ratio:   1     1     1     0     0
=====================================================

After Sorting Sack Items:
=====================================================
Price:   23    44    37    20    15
Weight: 60    50    36    20    10
Ratio:   0     0     1     1     1
=====================================================

Maximum Profit is: 23
PS H:\STUDY\College\DAA Lab>
```

0/1 Knapsack Output:

```
PS H:\STUDY\College\DAA Lab> cd "h:\STUDY\College\DAA Lab\" ; if ($?) { g++
knapsack.cpp -o knapsack } ; if ($?) { .\knapsack }
Enter the Capacity of the Sack: 5
Enter the N Item of Array: 4
Enter Price of Item no. 1: 3
Enter Weight of Item no.1: 2
============================================
Enter Price of Item no. 2: 4
Enter Weight of Item no.2: 3
============================================
Enter Price of Item no. 3: 5
Enter Weight of Item no.3: 4
============================================
Enter Price of Item no. 4: 6
Enter Weight of Item no.4: 5
Enter your Choice:
1. Fractional Knapsack
2. 0/1 Knapsack
2
Maximum Profit: 7
```