Assignment No: 4

Name: Kedar Kale

Roll No: SCETTYB129

Batch: B12

Title: Write a program for Fractional Knapsack using greedy method.

Algorithm:

```
for i = 1 to n
  do x[i] = 0
weight = 0
for i = 1 to n
  if weight + w[i] ≤ W then
    x[i] = 1
    weight = weight + w[i]
  else
    x[i] = (W - weight) / w[i]
    weight = W
    break
return x
```

Time Complexity:

If the provided items are already sorted into a decreasing order of piwipiwi, then the whileloop takes a time in **O(n)**; Therefore, the total time including the sort is in **O(n logn)**.

Analysis:

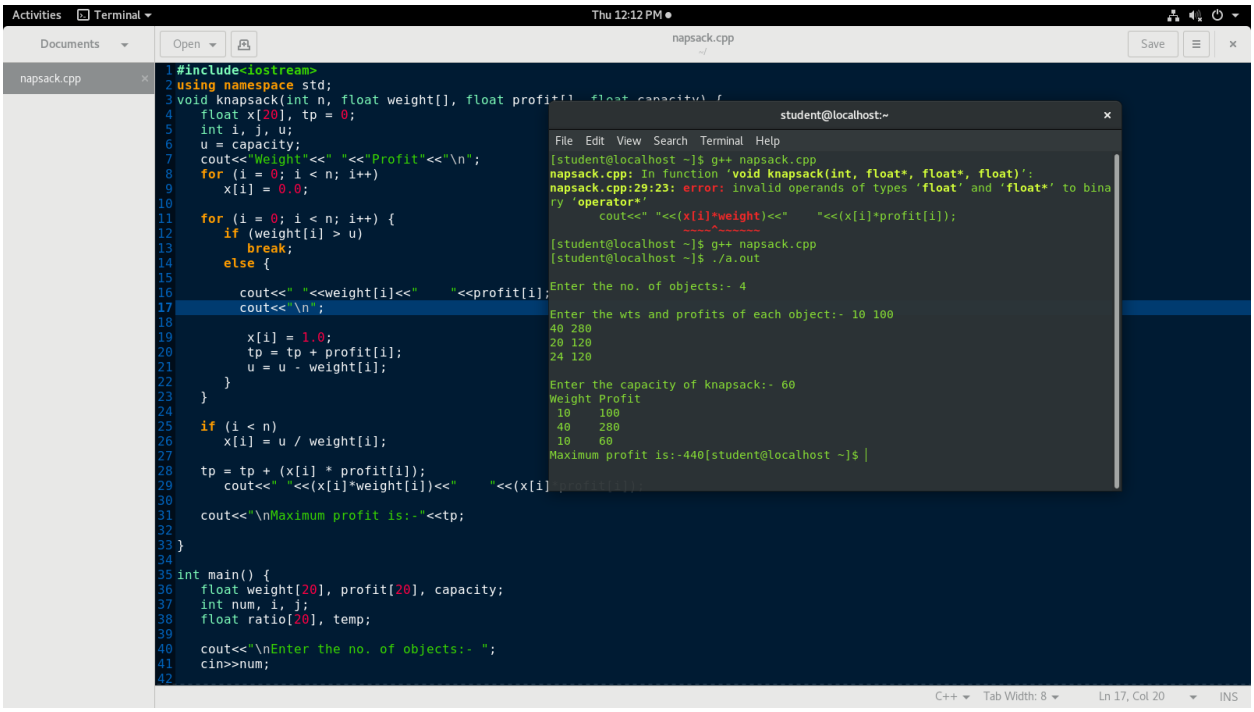
```
algorithm FractionalKnapsack(S,W):
  Input:  Set S of items i with weight wi and benefit bi all positive.
        Knapsack capacity W>0.
  Output: Amount xi of i that maximizes the total benefit without
        exceeding the capacity.

  for each i in S do
    xi ← 0        { for items not chosen in next phase }
    vi ← bi/wi    { the value of item i "per pound" }
  w ← W           { remaining capacity in knapsack }

  while w > 0 do
    remove from S an item of maximal value   { greedy choice }
    xi ← min(wi,w)  { can't carry more than w more }
    w ← w-xi
```

Program:

```
#include<iostream>
using namespace std;
void knapsack(int n, float weight[], float profit[], float capacity) {
   float x[20], tp = 0;
   int i, j, u;
   u = capacity;
   cout<<"Weight"<<" "<<"Profit"<<"\n";
   for (i = 0; i < n; i++)
      x[i] = 0.0;

   for (i = 0; i < n; i++) {
      if (weight[i] > u)
         break;
      else {

       cout<<" "<<weight[i]<<"    "<<profit[i];
       cout<<"\n";

         x[i] = 1.0;
         tp = tp + profit[i];
         u = u - weight[i];
      }
   }

   if (i < n)
      x[i] = u / weight[i];

   tp = tp + (x[i] * profit[i]);
      cout<<" "<<(x[i]*weight[i])<<"    "<<(x[i]*profit[i]);

   cout<<"\nMaximum profit is:-"<<tp;

}

int main() {
   float weight[20], profit[20], capacity;
   int num, i, j;
   float ratio[20], temp;

   cout<<"\nEnter the no. of objects:- ";
   cin>>num;

   cout<<"\nEnter the wts and profits of each object:- ";
   for (i = 0; i < num; i++) {
      cin>>weight[i]>>profit[i];
   }

   cout<<"\nEnter the capacity of knapsack:- ";
   cin>>capacity;

   for (i = 0; i < num; i++) {
      ratio[i] = profit[i] / weight[i];
```

```
    }

    for (i = 0; i < num; i++) {
        for (j = i + 1; j < num; j++) {
            if (ratio[i] < ratio[j]) {
                temp = ratio[j];
                ratio[j] = ratio[i];
                ratio[i] = temp;

                temp = weight[j];
                weight[j] = weight[i];
                weight[i] = temp;

                temp = profit[j];
                profit[j] = profit[i];
                profit[i] = temp;
            }
        }
    }

    knapsack(num, weight, profit, capacity);
    return(0);
}
```

Input/Output:



Conclusion: We have successfully executed the Fractional Knapsack Problem using Greedy Approach.