Name: LAKHAN KUMAWAT

Roll No: 1906055

Course Code: CS4403

Assignment no: 02

Date: 03/02/2021

1) Solve the following knapsack problem using Greedy method and find the maximum profit. Assume the Capacity of knapsack is 15.

Item { I1, I2, I3, I4 }   profit {10, 10, 12, 18}

weight {2, 4, 6, 9}

Solution: Given Capacity of knapsack = 15

Step 01: To get the solution arrange objects in decending order of profit/weights ratio as shown below:-

$P_1/W_1 = 10/2 = 5$

$P_2/W_2 = 10/4 = 2.5$

$P_3/W_3 = 12/6 = 2$

$P_4/W_4 = 18/9 = 2$

Arranging in decreasing order of pi/wi we get:

Item	weight	profit	Pi/wi
I1	2	10	5
I2	4	10	2.5
I3	6	12	2
I4	9	18	2

Step:02 The fraction of the objects selected and the profit we get Can be Computed as shown below:

Remaining Capacity Table is given below:-

Remaining Cap.	object selected	weight of the object	fraction of the object selected
15	I1	2	1 full unit
15-2 = 13	I2	4	1 full unit
13-4 = 9	I3	6	1 full unit
9-6 = ·	I4	9	1/3 fraction

Step:03

So, the solution vector will be $(1,1,1,1/3)$

$\text{profits} = 1 \times 10 + 1 \times 10 + 1 \times 12 + \frac{1}{3} \times 18$

$\text{profits} = 20 + 18$

$[\text{profit} = 38]$

So, the maximum profit of knapsack problem, which we get by fractional greedy method is 38.

Name : Lakhan Kumawat

Roll No : 1906055

Subject : CSL4403

Lab 04(Que 1)

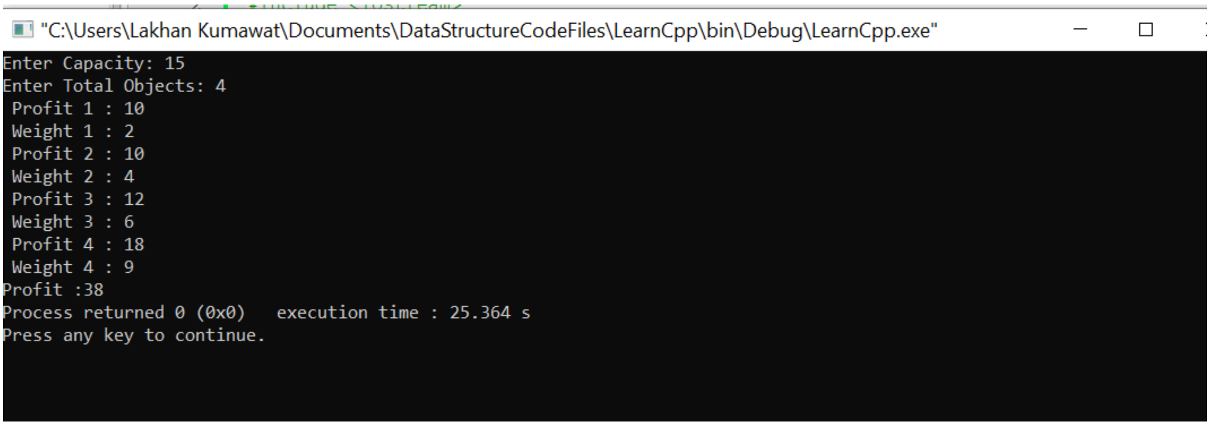
Write a program to implement Fractional Knapsack .

Program Code : C++

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

//1.Take profit/weight ratio and sort in decreasing order.
//2.Take the sum weight till sum is not exceed capacity.
//3.Output the maximum profit or all the included weights.

struct Object{
float weight;
float profit;
float PWratio;
};
bool cmp(struct Object a, struct Object b)
{
   return a.PWratio > b.PWratio;
}
```

```
void KnapSackFractional(Object a[],int n,int capacity){
sort(a,a + n,cmp);
float p=0;
for(int i=0;i<n;i++){
    if(a[i].weight<=capacity){
        p+=a[i].profit;
        capacity-=a[i].weight;
    }
    else if(capacity!=0){
        a[i].PWratio= capacity/a[i].weight;
        p+=a[i].PWratio*a[i].profit;
         capacity-=a[i].PWratio*a[i].weight;
    }
}
cout<<"Profit :"<<p;
}

int main(){
    int no,capacity;
    cout<<"Enter Capacity: ";
   cin>>capacity;
    cout<<"Enter Total Objects: ";
cin>>no;
Object Arr[no];
for(int i=0;i<no;i++){
   cout<<" Profit "<<i+1<<" : "; cin>>Arr[i].profit;
     cout<<" Weight "<<i+1<<" : "; cin>>Arr[i].weight;
   Arr[i].PWratio = Arr[i].profit/Arr[i].weight;
  //cout<<" Ratio "<<i<<" : "; cout<<Arr[i].PWratio<<endl;//Ratio
}

KnapSackFractional(Arr,no,capacity);

return 0;
}
```

Outputs :



"C:\Users\Lakhan Kumawat\Documents\DataStructureCodeFiles\LearnCpp\bin\Debug\LearnCpp.exe"

```
Enter Capacity: 15
Enter Total Objects: 4
 Profit 1 : 10
 Weight 1 : 2
 Profit 2 : 10
 Weight 2 : 4
 Profit 3 : 12
 Weight 3 : 6
 Profit 4 : 18
 Weight 4 : 9
Profit :38
Process returned 0 (0x0)   execution time : 25.364 s
Press any key to continue.
```

Q2) **Problem statement :02**

Solve the following job sequencing with deadline problem for the following jobs using greedy algorith and find the maximum profit.

Job Number {$J_1, J_2, J_3, J_4, J_5, J_6$}

profit = {300, 250, 130, 212, 100, 424}

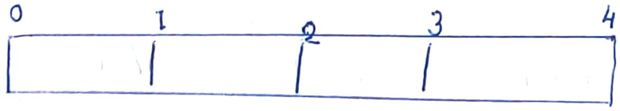
Deadline = {4, 2, 3, 3, 3, 3}

Solution: Step 01: Sort all the given jobs in decreasing order of their profit :-

Jobs	J_6	J_1	J_2	J_4	J_3	J_5
Deadlines	3	4	2	3	3	3
profit	424	300	250	212	130	100

Step 02: Value of maximum deadline = 4

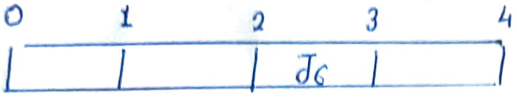
So, draw a Gantt chart with maximum time on Gantt chart = 4 units.



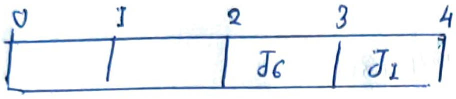
Now we take each job one by one in order they appear in Step 01.

We place the Job on Gantt chart as far as possible from 0.

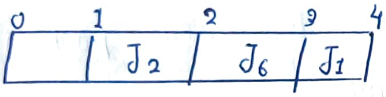
Step 03 : We take Job 6 , Since its deadline is 3 so we place it in the first empty cell before deadline 3 as-



Now we Select J1 , Since deadline is 4 place is just 4 in the empty Cell.

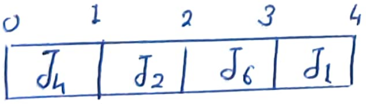


Again , J2 deadline is 2 place in the empty Cell 2 as-



Now we take Joby 4, Since its deadline is 4, we place it in first empty Cell before deadline.

But the Second and third Cell are already filled so we place job J4 , in the first Cell.



Now left over Jobs J3 and J5 Cannot be Completed Cause all the slots for Jobs before deadline 4 are filled Completely . Thus Job3 and Job5 Cannot be Completed.

J_4 , J_2, J_6, J_7 is the required order in Which jobs must be completed so they get maximum profit.

Maximum profit earned

= Sums of profits of all the jobs in optimal sch dule

= 212 + 250 + 424 + 300

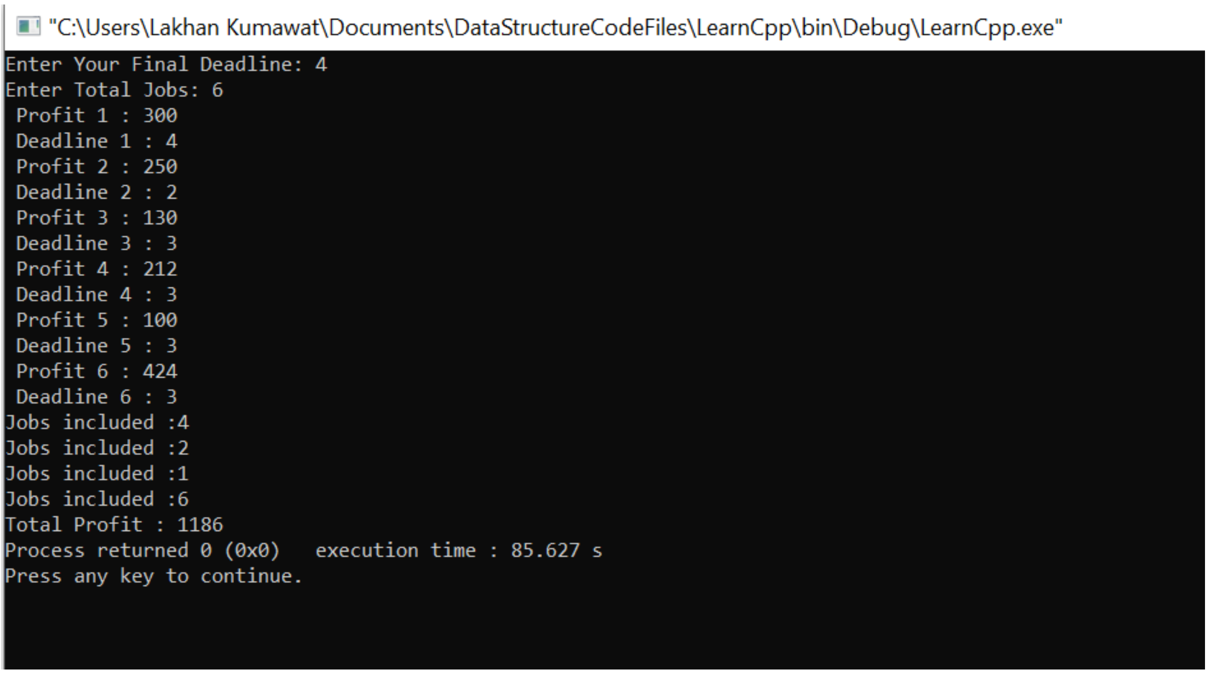
= 1186 Units

So the maximum profits of job Sequencing problem using Greedy method is 1186 Units.

Program Code C++ :

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

//1.Sort the profit in decreasing order
//2.set the job just before of before deadline Time.

struct Job{
   int JobNo;
int deadL;
int profit;
};

bool comparison(Job a, Job b)
{
    return (a.profit > b.profit);
}

void JobSchedulingUsingDeadline(Job a[],int n, int finaldead){
sort(a,a+n,comparison);

int result[n],totalprofit=0;
bool slot[n];

for(int j=0;j<n;j++)
   slot[j]=false;

for(int i=0;i<n;i++){

  for(int j=min(n,a[i].deadL)-1;j>=0;j--){
   if(slot[j]==false){
      totalprofit+=a[i].profit;
       result[j] = i;  // Add this job to result
          slot[j] = true; // Make this slot occupied
          break;
   }
  }
}
cout<<"Jobs Sequencing Order : ";
for(int o=0;o<n;o++)
   if(slot[o]){
   cout<< a[result[o]].JobNo+1<<" ";
   }
   cout<<"Total Profit : "<<totalprofit;
}

int main(){
 int no,finaldead;

    cout<<"Enter Total Jobs: ";
cin>>no;
cout<<"Enter Your Final Deadline: ";
   cin>>finaldead;
Job Arr[no];
cout<<" Profit  Deadline"<<"\n";
for(int i=0;i<no;i++){


  cin>>Arr[i].profit>>Arr[i].deadL;
     Arr[i].JobNo = i;
}

JobSchedulingUsingDeadline(Arr,no,finaldead);
}
```

Outputs : Job sequencing and Profit



"C:\Users\Lakhan Kumawat\Documents\DataStructureCodeFiles\LearnCpp\bin\Debug\LearnCpp.exe"

```
Enter Your Final Deadline: 4
Enter Total Jobs: 6
 Profit 1 : 300
 Deadline 1 : 4
 Profit 2 : 250
 Deadline 2 : 2
 Profit 3 : 130
 Deadline 3 : 3
 Profit 4 : 212
 Deadline 4 : 3
 Profit 5 : 100
 Deadline 5 : 3
 Profit 6 : 424
 Deadline 6 : 3
Jobs included :4
Jobs included :2
Jobs included :1
Jobs included :6
Total Profit : 1186
Process returned 0 (0x0)   execution time : 85.627 s
Press any key to continue.
```