Practical No 03: Program to solve a fractional Knapsack problem using a greedy method.

#Code:

```
#include <iostream>
#include<vector>
#include<algorithm>
using namespace std;

double knapsack(vector<int>&weight,vector<int>&value,int w){
    int n=value.size();
    vector<pair<double,int>>v;
    for(int i=0;i<n;i++){
        double a=double(value[i])/weight[i];
        v.push_back({a,i});

    }

    sort(v.rbegin(),v.rend());
    int i=0;
    double profit=0.0;
    while(w && i<n){
        int idx=v[i].second;
        if(w>=weight[idx]){
            profit+=value[idx];
            w=w-weight[idx];
        }
        else{
            profit+=v[i].first*w;
            w=0;
        }
        i++;
    }

    return profit;

}
int main()
{
    int w=50;
    vector<int>weight={10,20,30};
    vector<int>value={60,100,120};

    cout<<”Maximum Profit:”<<knapsack(weight,value,w);

    return 0;
}
```

#Output:

```
Maximum Profit :240
```