****

**NAME: -** Darji Akshatkumar Hiteshbhai

**RollNo: -** 23MCD001

**Branch: -** M.tech-CSE**(Data Science)**

**Subject: -** Complexity Theory & Algorithms

**Practical-6**

**Aim:** Implement Fraction Knapsack problem using Greedy approach.

**Code for Fractional Knapsack –**

#include<bits/stdc++.h>

using namespace std;

void knapsackProblem(map<float, pair<int, int>, greater<float>> &final, int object){

    int M;

    cout << endl;

    cout << "Enter the Knapsack capacity : ";

    cin >> M;

    float p = 0;

    for(auto it : final){

        if(M > 0 && (it.second.second <= M)){

            M -= it.second.second;

            p += it.second.first;

        }

        else{

            if(M > 0){

                p += it.second.first \* (static\_cast<float>(M) / it.second.second);

            }

            break;

        }

    }

    cout << "------------------------------------------";

    cout << endl << "Maximum Profit : " << p;

}

int main(){

    int object;

    cin >> object;

    vector<int> profit(object);

    vector<int> weight(object);

    vector<pair<int, int>> pw;

    vector<float> pw\_ratio;

    map<float, pair<int, int>, greater<float>> final;

    for(int i=0;i<object;i++){

        cin >> profit[i];

        cin >> weight[i];

    }

    for(int i=0;i<object;i++){

        for(int j=i;j<i+1;j++){

            pw.push\_back({profit[i], weight[i]});

        }

    }

    cout << endl;

//    for(auto it : pw){

//        cout << "[profit: " << it.first << ", weight: " << it.second << "]";

//    }

    for(int i=0;i<object;i++){

        float result = static\_cast<float>(pw[i].first) / static\_cast<float>(pw[i].second);

        pw\_ratio.push\_back(result);

    }

//    for(auto it : pw\_ratio){

//        cout << it << ", ";

//    }

    for(int i=0;i<object;i++){

        final[pw\_ratio[i]] = {pw[i].first, pw[i].second};

    }

    cout << "Knapsack Table is as follows : ";

    cout << endl;

    for(auto it : final){

        cout << "[p/w: "<< it.first << ", (Profit: " << it.second.first << ", Weight: " << it.second.second << ")]";

    }

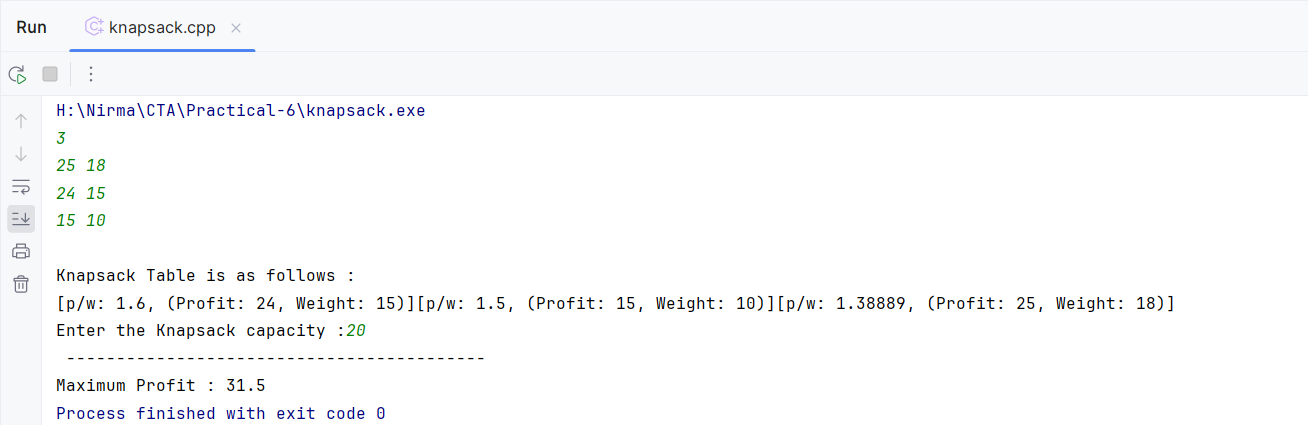
    knapsackProblem(final, object);

    return 0;

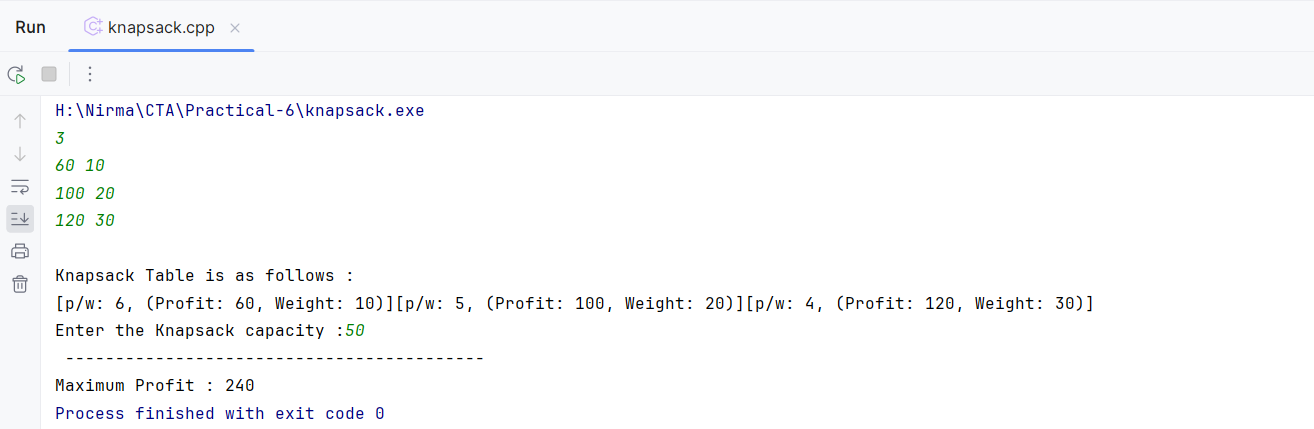
}

**Output –**

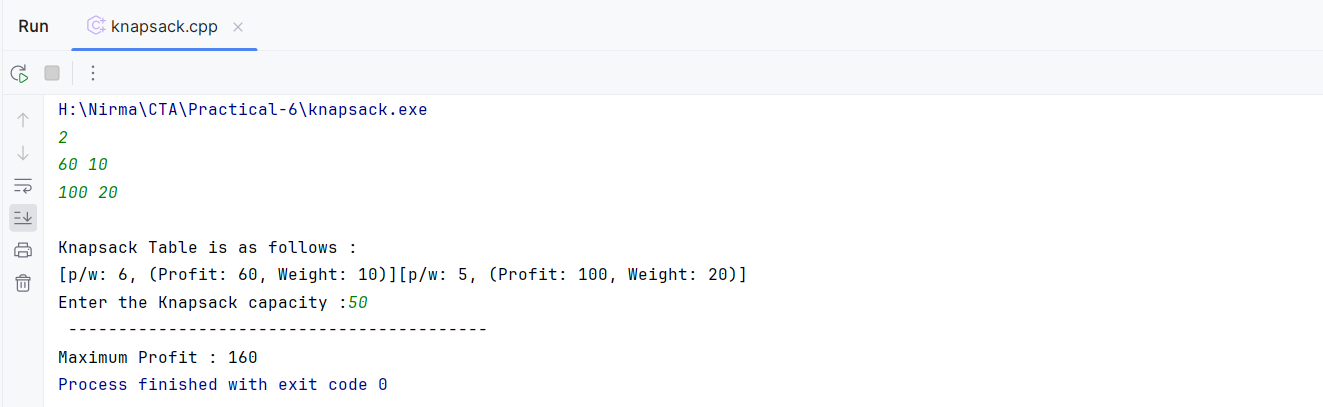
**Test Case – 1**

****

**Test Case – 2**

****

**Test Case – 3**

****

**Test Case – 4**

