**Practical No : 4**

**Name :** Samrudhi Pawar

**Roll No :** 44

**Title :**Develop a program to solve a fractional Knapsack problem using a greedy method.

**Program :**

#include <iostream>

#include <algorithm>

using namespace std;

// Structure to represent an item

struct Item {

int value; // Value of the item

int weight; // Weight of the item

};

// Function to calculate value/weight ratio and sort items based on this ratio

bool compare(Item a, Item b) {

double r1 = (double)a.value / a.weight;

double r2 = (double)b.value / b.weight;

return r1 > r2;

}

// Function to solve the fractional knapsack problem

double fractionalKnapsack(int W, Item arr[], int n) {

// Sort items based on value/weight ratio

sort(arr, arr + n, compare);

int currentWeight = 0; // Current weight in knapsack

double finalValue = 0.0; // Result (total value in knapsack)

// Loop through all items

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

// If adding the current item doesn't exceed capacity, add it completely

if (currentWeight + arr[i].weight <= W) {

currentWeight += arr[i].weight;

finalValue += arr[i].value;

}

// If we can't add the full item, add fractional part

else {

int remaining = W - currentWeight;

finalValue += arr[i].value \* ((double)remaining / arr[i].weight);

break; // Knapsack is full

}

}

return finalValue;

}

int main() {

int n, W;

// Input the number of items

cout << "Enter the number of items: ";

cin >> n;

// Array to store items

Item\* arr = new Item[n];

// Input each item's value and weight

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

cout << "Enter value and weight of item " << i + 1 << ": ";

cin >> arr[i].value >> arr[i].weight;

}

// Input the capacity of the knapsack

cout << "Enter the capacity of the knapsack: ";

cin >> W;

// Function call to solve the problem

double maxValue = fractionalKnapsack(W, arr, n);

// Output the maximum value achievable

cout << "Maximum value we can obtain = " << maxValue << endl;

// Free dynamically allocated memory

delete[] arr;

return 0;

}

**Output :**

Enter the number of items: 3

Enter value and weight of item 1: 60

10

Enter value and weight of item 2: 100

20

Enter value and weight of item 3: 120

30

Enter the capacity of the knapsack: 50

Maximum value we can obtain = 240