Assignment 5

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

// Structure to represent an item with profit and weight
struct Item {
    int profit;
    int weight;
};

// Comparison function to sort items by profit-to-weight ratio
bool compare(Item a, Item b) {
    double r1 = (double)a.profit / a.weight;
    double r2 = (double)b.profit / b.weight;
    return r1 > r2;
}

// Function to calculate the maximum profit in the knapsack using Greedy approach
double fractionalKnapsack(int W, Item arr[], int n) {
    // Sort items by profit-to-weight ratio
    sort(arr, arr + n, compare);

    double totalProfit = 0.0;   // Total profit in the knapsack
    int currentWeight = 0;      // Current weight in the knapsack

    cout << "Item details taken in knapsack:\n";
    cout << "Profit\tWeight\tFraction taken\n";

    for (int i = 0; i < n; i++) {
        // If adding the whole item doesn't exceed capacity
```

```
        if (currentWeight + arr[i].weight <= W) {
            currentWeight += arr[i].weight;
            totalProfit += arr[i].profit;
            cout << arr[i].profit << "\t" << arr[i].weight << "\t" << "1 (Full)" << endl;
        }
        // Otherwise take the fraction of the item that fits
        else {
            int remainingWeight = W - currentWeight;
            double fraction = (double)remainingWeight / arr[i].weight;
            totalProfit += arr[i].profit * fraction;
            cout << arr[i].profit << "\t" << arr[i].weight << "\t" << fraction << endl;
            break;  // Knapsack is full
        }
    }

    return totalProfit;
}

int main() {
    int n, W;

    // Input number of items and knapsack capacity
    cout << "Enter the number of items: ";
    cin >> n;

    Item arr[n];

    // Input profit and weight of each item
    cout << "Enter profit and weight of each item:\n";
    for (int i = 0; i < n; i++) {
        cout << "Item " << i + 1 << " - Profit: ";
```

```
        cin >> arr[i].profit;
        cout << "Item " << i + 1 << " - Weight: ";
        cin >> arr[i].weight;
    }

    // Input knapsack capacity
    cout << "Enter the capacity of the knapsack: ";
    cin >> W;

    // Calculate maximum profit for the knapsack
    double maxProfit = fractionalKnapsack(W, arr, n);

    cout << "Maximum profit in Knapsack: " << maxProfit << endl;

    return 0;
}
```

Output:

```
Enter the number of items: 3
Enter profit and weight of each item:
Item 1 - Profit: 60
Item 1 - Weight: 10
Item 2 - Profit: 100
Item 2 - Weight: 20
Item 3 - Profit: 120
Item 3 - Weight: 30
Enter the capacity of the knapsack: 50
Item details taken in knapsack:
Profit  Weight  Fraction taken
60      10      1 (Full)
100     20      1 (Full)
120     30      0.666667
Maximum profit in Knapsack: 240

--------------------------------
Process exited after 38.85 seconds with return value 0
Press any key to continue . . .
```