1.Greedy Approach: Implementation of Fractional Knapsack

```
#include <stdio.h>

struct Item {
    int weight;
    int profit;
};

void fractionalKnapsack(struct Item items[], int n, int capacity) {
    double ratio[n];
    for (int i = 0; i < n; i++)
        ratio[i] = (double)items[i].profit / items[i].weight;

    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (ratio[j] < ratio[j + 1]) {
                double temp = ratio[j];
                ratio[j] = ratio[j + 1];
                ratio[j + 1] = temp;

                struct Item t = items[j];
                items[j] = items[j + 1];
                items[j + 1] = t;
            }
        }
    }

    double totalProfit = 0.0;
    int curWeight = 0;

    for (int i = 0; i < n; i++) {
        if (curWeight + items[i].weight <= capacity) {
            curWeight += items[i].weight;
            totalProfit += items[i].profit;
        } else {
            int remain = capacity - curWeight;
            totalProfit += items[i].profit * ((double)remain / items[i].weight);
            break;
        }
    }

    printf("Maximum profit in Knapsack = %.2f\n", totalProfit);
```

```
}

int main() {
    struct Item items[] = {{10, 60}, {20, 100}, {30, 120}};
    int n = sizeof(items) / sizeof(items[0]);
    int capacity = 50;

    fractionalKnapsack(items, n, capacity);
    return 0;
}
```