1.Fractional Knapsack problem

```
#include <stdio.h>
#include <stdlib.h>

struct Item {
    int weight;
    int value;
    float ratio;
};

int compare(const void *a, const void *b) {
    struct Item *item1 = (struct Item *)a;
    struct Item *item2 = (struct Item *)b;
    return (item2->ratio - item1->ratio > 0) ? 1 : -1;
}

float fractional_knapsack(struct Item items[], int n, int capacity) {
    qsort(items, n, sizeof(items[0]), compare);

    int currentWeight = 0;
    float totalValue = 0.0;

    for (int i = 0; i < n; i++) {
        if (currentWeight + items[i].weight <= capacity) {
            currentWeight += items[i].weight;
            totalValue += items[i].value;
        } else {
            int remainingWeight = capacity - currentWeight;
            totalValue += items[i].value * ((float)remainingWeight / items[i].weight);
            break;
        }
```

```
    }

    return totalValue;
}

int main() {
    int n, capacity;

    printf("Enter the number of items: ");
    scanf("%d", &n);

    struct Item items[n];

    printf("Enter the weights and values of the items:\n");
    for (int i = 0; i < n; i++) {
        printf("Item %d - Weight: ", i + 1);
        scanf("%d", &items[i].weight);
        printf("Item %d - Value: ", i + 1);
        scanf("%d", &items[i].value);

        items[i].ratio = (float)items[i].value / items[i].weight;
    }

    printf("Enter the capacity of the knapsack: ");
    scanf("%d", &capacity);

    float maxValue = fractional_knapsack(items, n, capacity);

    printf("Maximum value in Knapsack = %.2f\n", maxValue);

    return 0;
```

}

Output

```
Enter the number of items: 3
Enter the weights and values of the items:
Item 1 - Weight: 5
Item 1 - Value: 66
Item 2 - Weight: 5
Item 2 - Value: 65
Item 3 - Weight: 3
Item 3 - Value: 89
Enter the capacity of the knapsack: 3
Maximum value in Knapsack = 89.00


=== Code Execution Successful ===
```