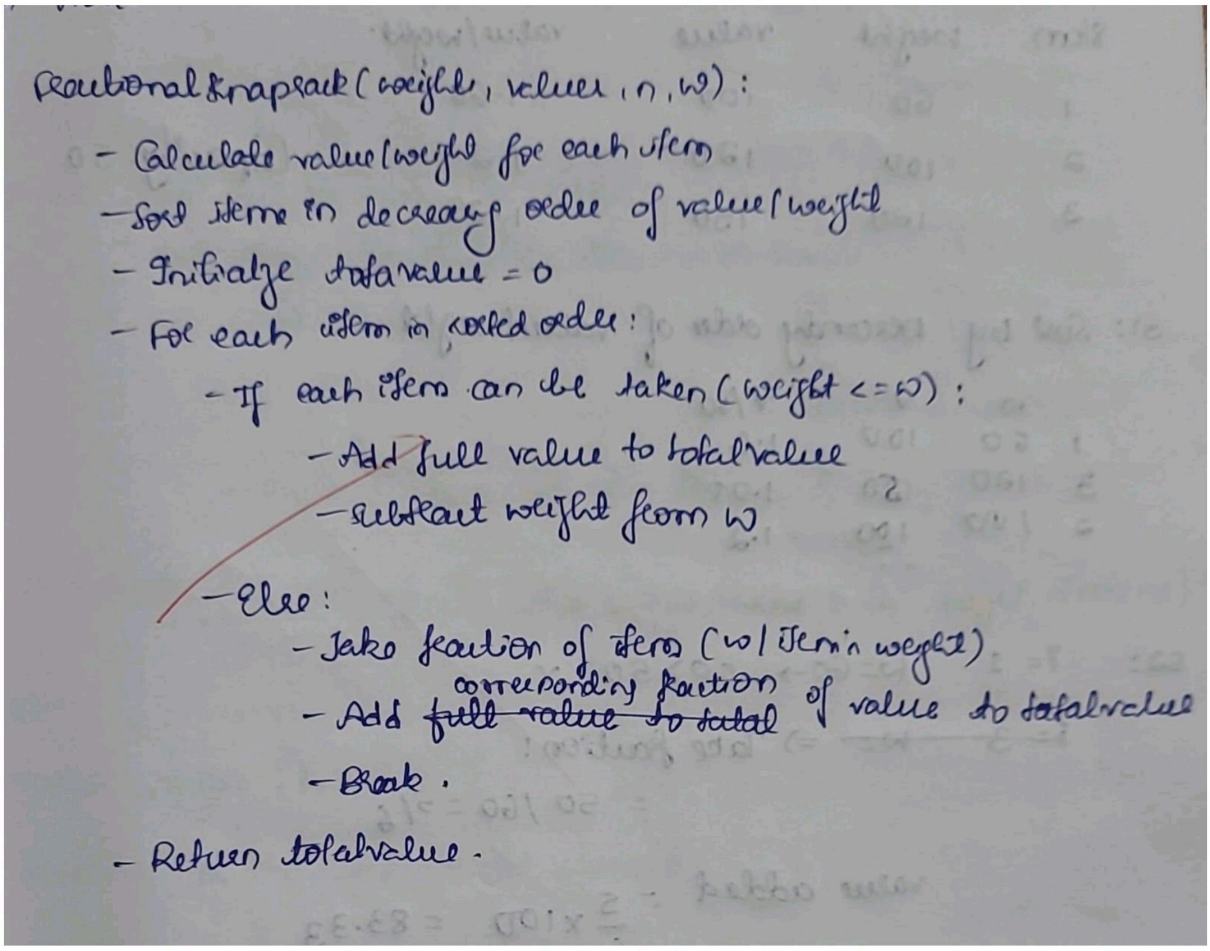
PROGRAM 7

Implement Fractional Knapsack using Greedy technique.

ALGORITHM



CODE

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

// Structure for an item
struct Item {
    int value, weight;
};

// Function to compare items by value/weight ratio
int compare(const void *a, const void *b) {
```

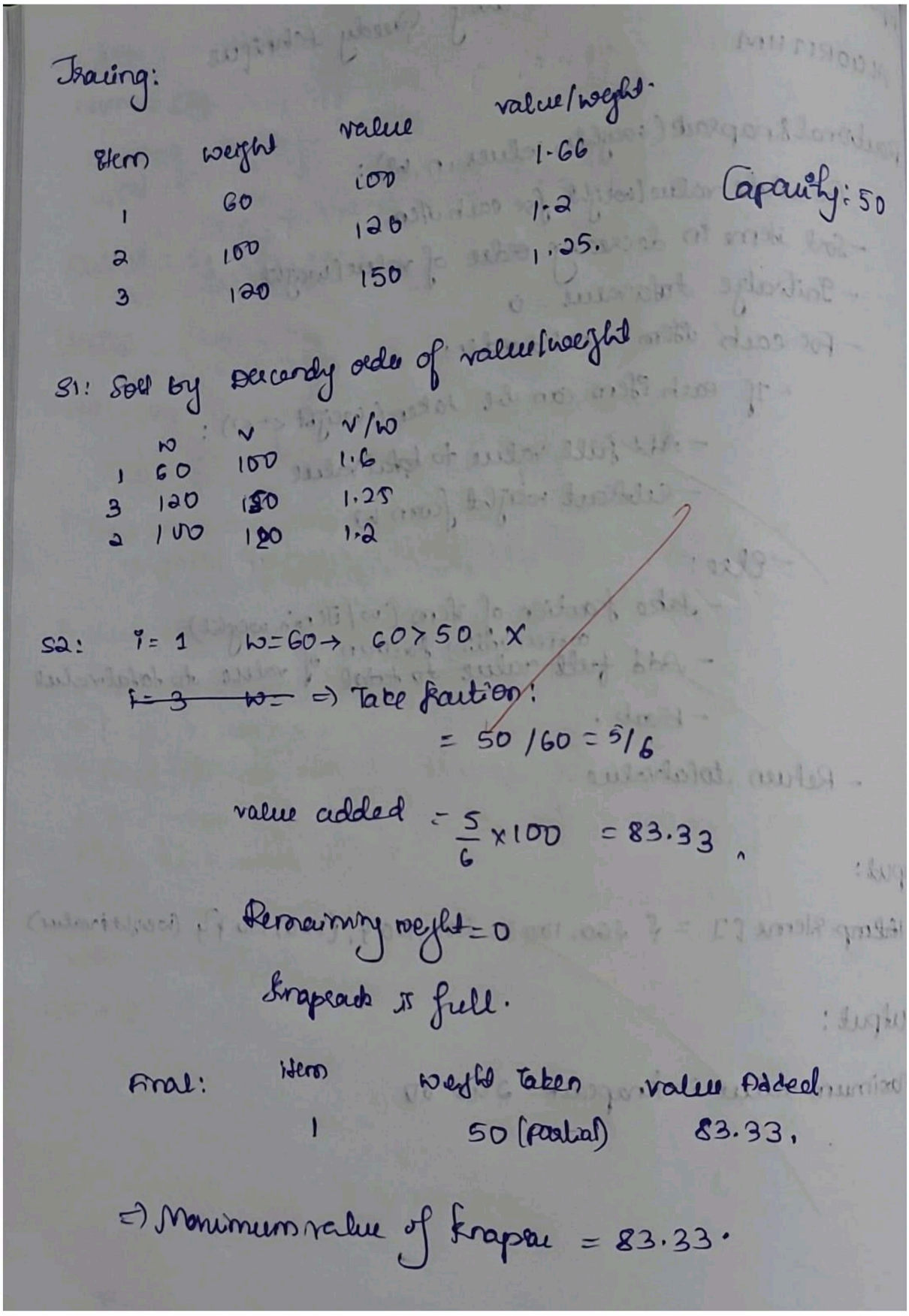
```
    double r1 = (double)((struct Item *)a)->value / ((struct Item *)a)->weight;
    double r2 = (double)((struct Item *)b)->value / ((struct Item *)b)->weight;
    return (r1 < r2) ? 1 : -1; // Descending order
}

// Function to return maximum value that can be put in knapsack
double fractionalKnapsack(int W, struct Item arr[], int n) {
    // Sort items by value/weight ratio
    qsort(arr, n, sizeof(arr[0]), compare);

    double totalValue = 0.0;

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

        if (arr[i].weight <= W) {
            // Take full item
            W -= arr[i].weight;
            totalValue += arr[i].value;
        } else {
            // Take fractional part
            totalValue += arr[i].value * ((double)W / arr[i].weight);
            break;
        }
    }

    return totalValue;
}

// Main function
```

```
int main() {
    int n, W;

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

    struct Item arr[n];
    printf("Enter value and weight of each item:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &arr[i].value, &arr[i].weight);
    }

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

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

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

    return 0;
}
```

OUTPUT:

```
Enter number of items: 3
Enter value and weight of each item:
100 60
120 100
150 120
Enter capacity of knapsack: 50
Maximum value in knapsack = 83.33
```

TRACING



Tracing:

Item	weight	value	value/weight
1	60	100	1.66
2	100	120	1.2
3	120	150	1.25

Capacity: 50

S1: Sort by decreasing order of value/weight

	w	v	v/w
1	60	100	1.6
3	120	150	1.25
2	100	120	1.2

S2: i = 1 w = 60 → 60 > 50 : X

~~i = 3 w =~~ ⇒ Take fraction:

= 50 / 60 = 5/6

value added = $\frac{5}{6} \times 100 = 83.33$.

Remaining weight = 0

Knapsack is full.

Final:

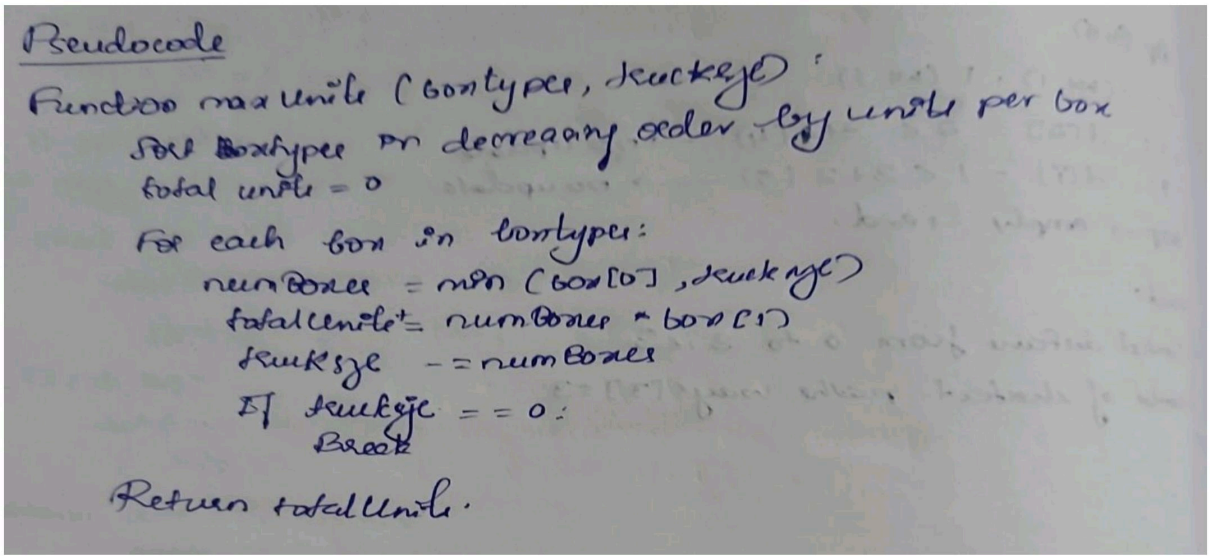
item	weight taken	value Added
1	50 (Partial)	83.33.

⇒ Maximum value of knapsack = 83.33.

LEETCODE 6

MAXIMUM UNITS ON A TRUCK

ALGORITHM

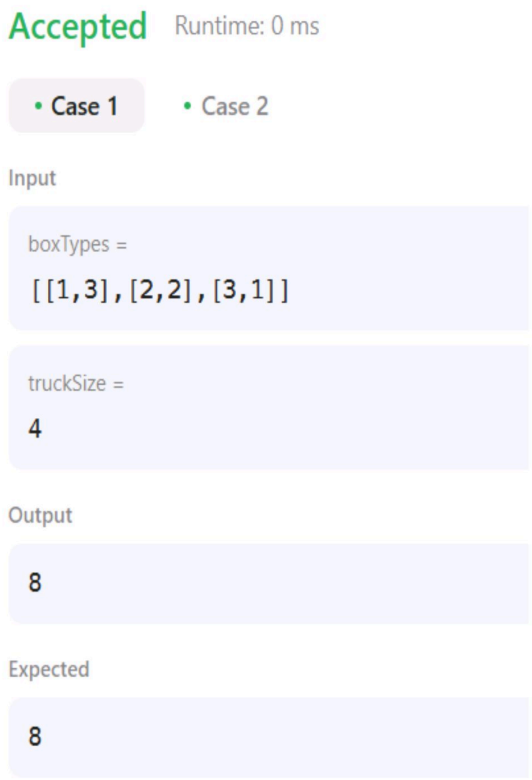


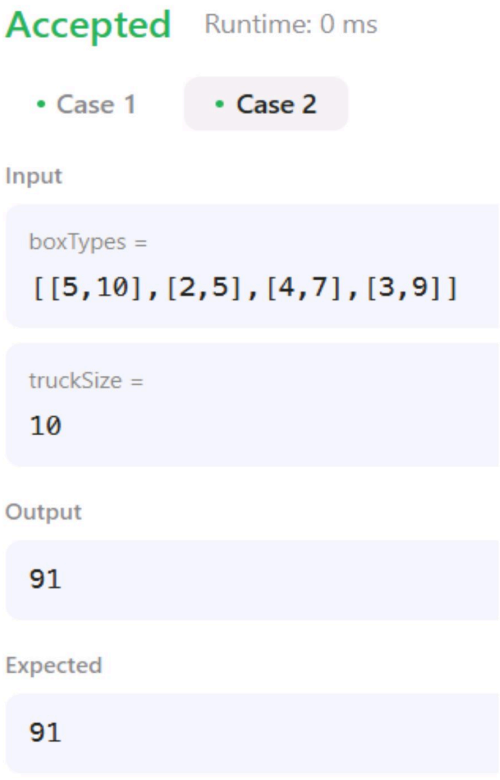
CODE

```
int cmp(const void* a, const void* b) {
    // Sort by units per box in descending order
    int* boxA = *(int**)a;
    int* boxB = *(int**)b;
    return boxB[1] - boxA[1];
}

int maximumUnits(int** boxTypes, int boxTypesSize, int* boxTypesColSize, int truckSize) {
    qsort(boxTypes, boxTypesSize, sizeof(int*), cmp);

    int totalUnits = 0;

    for (int i = 0; i < boxTypesSize; i++) {
        int boxCount = boxTypes[i][0];
        int unitsPerBox = boxTypes[i][1];
```

```
        int numBoxes = boxCount < truckSize ? boxCount : truckSize;
        totalUnits += numBoxes * unitsPerBox;
        truckSize -= numBoxes;

        if (truckSize == 0)
            break;
    }

    return totalUnits;
}
```

OUTPUT





TRACING

Example Tracing

Input: boxTypes = [[5,10], [2,5], [4,7], [3,9]]
truckSize = 10.

S1: Sort by units per box (descending):
sorted boxTypes: [[5,10], [3,9], [4,7], [2,5]]

S2: Initialize totalUnits = 0, truckSize = 10

S3: Loop through sorted boxTypes

Step	Picked Boxes	Units/Box	Used	Remaining truckSize	TotalUnits
1	[5,10]	10	5	10-5=5	0+50=50
2	[3,9]	9	3	5-3=2	50+27=77
3	[4,7]	7	2	2-2=0	77+14=91

→ truckSize = 0 : stop.

Output:
91.