S.No: 9	Exp. Name: ***Fractional Knapsack Problem***	Date:2025-07-18

ID: 24070521125

Aim:

The below program has a method void knapsack(). Which takes four parameters **number of objects**, the **weight of each object**, the **profit** corresponding to each one and the **capacity of the knapsack**. Write a program using a fractional knapsack algorithm to get the maximum profit.

Print the output as follows:

```
Sample Input and Output:
Enter the no. of objects: 6
Enter the weights and profits of each object:
1 2
4 5
8 9
4 6
5 2
3 5
Enter the capacity of knapsack:10
Maximum profit is:- 15.500000
```

Source Code:

knapsack.c

```
# include<stdio.h>
void knapsack(int n, float weight[], float profit[], float capacity) {
  // write your code here
  float totalProfit = 0.0, totalWeight = 0.0;

    for (int i = 0; i < n; i++) {
        if (totalWeight + weight[i] <= capacity) {
            // If the entire item fits, take it completely
            totalWeight += weight[i];
            totalProfit += profit[i];
        } else {
            // Take the fraction of the item that fits
            float remain = capacity - totalWeight;
            totalProfit += profit[i] * (remain / weight[i]);
            totalWeight = capacity; // Knapsack is full
            break;
        }
    }

    // Output the result
    printf("Maximum profit is:- %.6f\n", totalProfit);
}

int main() {
   float weight[20], profit[20], capacity;
   int num, i, j;
   float ratio[20], temp;
   printf("Enter the no. of objects: ");
   scanf("%d", &num);
```

```
    printf("Enter the weights and profits of each object:\n");
    for (i = 0; i < num; i++) {
       scanf("%f %f", &weight[i], &profit[i]);
    }
    printf("Enter the capacity of knapsack:");
    scanf("%f", &capacity);
    for (i = 0; i < num; i++) {
       ratio[i] = profit[i] / weight[i];
    }

    for (i = 0; i < num; i++) {
       for (j = i + 1; j < num; j++) {
          if (ratio[i] < ratio[j]) {
             temp = ratio[j];
             ratio[j] = ratio[i];
             ratio[i] = temp;
             temp = weight[j];
             weight[j] = weight[i];
             weight[i] = temp;
             temp = profit[j];
             profit[j] = profit[i];
             profit[i] = temp;
          }
       }
    }
    knapsack(num, weight, profit, capacity);
    return(0);
}
```

Execution Results - All test cases have succeeded!

Test Case - 1
User Output
Enter the no. of objects: 6
Enter the weights and profits of each object: 1 2
4 5
8 9
4 6
5 2
3 5
Enter the capacity of knapsack: 10
Maximum profit is:- 15.500000

Test Case - 2
User Output
Enter the no. of objects: 5
Enter the weights and profits of each object: 4 6
1 3
7 5
5 3
3 4

ID: 24070521125

```
Enter the capacity of knapsack: 10
Maximum profit is:- 14.428572
```