Write a c program to perform Knapsack Problem using GreedySolution.

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

#define MAX_ITEMS 100

// Function to compare items based on their value-to-weight ratio for sorting
int compare(const void *a, const void *b) {
    double ratioA = ((int *)a)[0] / (double)((int *)a)[1];
    double ratioB = ((int *)b)[0] / (double)((int *)b)[1];

    return (ratioB > ratioA) - (ratioB < ratioA);
}

// Function to solve the Knapsack problem using a Greedy approach
double knapsackGreedy(int items[][2], int n, int capacity) {
    // Sort items based on value-to-weight ratio in non-increasing order
    qsort(items, n, sizeof(items[0]), compare);

    double totalValue = 0.0;
    int currentWeight = 0;

    // Iterate through sorted items and add them to the knapsack
    for (int i = 0; i < n; i++) {
        if (currentWeight + items[i][1] <= capacity) {
            // Add the whole item
            totalValue += items[i][0];
            currentWeight += items[i][1];
        } else {
            // Add a fraction of the item to fill the knapsack
            double remainingWeight = capacity - currentWeight;
            totalValue += (remainingWeight / items[i][1]) * items[i][0];
            break;  // Knapsack is full
        }
    }

    return totalValue;
}

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

```
    // Input the number of items and the knapsack capacity
    printf("Enter the number of items: ");
    scanf("%d", &n);
    printf("Enter the knapsack capacity: ");
    scanf("%d", &capacity);

    int items[MAX_ITEMS][2];

    // Input values and weights for each item
    printf("Enter the values and weights for each item:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &items[i][0], &items[i][1]);
    }

    // Solve the Knapsack problem using Greedy approach
    double result = knapsackGreedy(items, n, capacity);

    // Display the result
    printf("Maximum value in the knapsack using Greedy approach: %.2f\n", result);

    return 0;
}
```

```
Enter the number of items: 4
Enter the knapsack capacity: 5
Enter the values and weights for each item:
3 2
4 3
5 4
6 5
Maximum value in the knapsack using Greedy approach: 8.33
```