#include <stdio.h>

#include <stdlib.h>

// Structure to represent an item

struct Item {

int value;

int weight;

double valuePerWeight; // Value-to-weight ratio

};

// Function to compare two items based on value-to-weight ratio

int compare(const void \*a, const void \*b) {

struct Item \*item1 = (struct Item \*)a;

struct Item \*item2 = (struct Item \*)b;

return (item2->valuePerWeight - item1->valuePerWeight);

}

// Function to perform fractional knapsack

double fractional\_knapsack(int capacity, struct Item items[], int n) {

// Sort items based on value-to-weight ratio

qsort(items, n, sizeof(items[0]), compare);

int currentWeight = 0; // Current weight in knapsack

double totalValue = 0.0; // Total value in knapsack

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

// If adding the entire item doesn't exceed capacity

if (currentWeight + items[i].weight <= capacity) {

currentWeight += items[i].weight;

totalValue += items[i].value;

} else {

// Otherwise, add fraction of the item to fill the knapsack

int remainingWeight = capacity - currentWeight;

totalValue += items[i].valuePerWeight \* remainingWeight;

break;

}

}

return totalValue;

}

int main() {

int capacity = 50;

struct Item items[] = {{60, 10, 0}, {100, 20, 0}, {120, 30, 0}};

int n = sizeof(items) / sizeof(items[0]);

// Calculate value-to-weight ratios for all items

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

items[i].valuePerWeight = (double)items[i].value / items[i].weight;

}

double maxValue = fractional\_knapsack(capacity, items, n);

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

return 0;

}