```
#include <iostream>

#include<algorithm>

using namespace std;



struct item {

int profit, weight;

};



bool cmp(struct item a, struct item b) { //compare item a and item b based on the ration of value and weight

double aRatio = (double)a.profit / a.weight;

double bRatio = (double)b.profit / b.weight;

return aRatio > bRatio;

}



double FracKnap(int weight, item itemList[], int n) {

sort(itemList, itemList + n, cmp); //sort item list using compare function reference
https://en.cppreference.com/w/cpp/algorithm/sort

int currWeight = 0; // Current weight in knapsack

double knapsackVal = 0.0;



for (int i = 0; i < n; i++) { //check through all items

if (currWeight + itemList[i].weight <= weight) { //when the space is enough for selected item, add it
```

```
currWeight += itemList[i].weight;

knapsackVal += itemList[i].profit;



}else{ //when no place for whole item, break it into smaller parts

int remaining = weight - currWeight;

knapsackVal += itemList[i].profit * ((double) remaining / itemList[i].weight);

break;

}

}

return knapsackVal;

}



int main() {

int weight = 15; // Weight of knapsack

item itemList[] = {{10,2}, {5, 3}, {15, 5},{7,7},{6,1},{18,4},{3,1} }; /*{profit,weight}*/

int n = 7;

cout << "Maximum value: " << FracKnap(weight, itemList, n);

}
```