"The Knapsack Problem"
(knapsack)

You are expected to demonstrate a working C program according to the description below. Your program should be uploaded (via Blackboard) before the deadline. You must use the file names specified in the problem description below. Late submissions will not be corrected and will receive zero credit, so even if your programs are not running correctly, you should submit your "best attempt".

Program description

One of the classic NP-hard problems is the knapsack problem. In this problem, the objective is to find an "optimal" packing of a backpack which will maximise the total value of the objects in the backpack, without exceeding a weight and cost budget. For example, Table 1 gives a list of items which you might take on a camping trip, together with their weight, cost, and "value" to the user. The goal is to find the selection of items which maximises the total value, while staying within a weight budget of 13 kg and a cost budget of €60.

For example, one packing might be
Pillow (V=1,W=2, C=6), Tent (V=7,W=6,C=30), Water (V=6,W=5,C=4)
which has a total value of 14, weight of 13 kg, and cost of €40.

However, a better packing might be
Knife (V=8,W=1, C=10), Water (V=6,W=5,C=4), Map (V=5, W=1,C=15), Tent (V=7, W=6,C=30)
which has a total value of 26, weight of 13 kg, and cost of €59.

Many NP-hard problems can be "solved" by brute force, if the dimensions of the data set are small. In a "brute force" optimisation, the algorithm evaluates every possible solution, and finds the best solution by checking out the total value, weight and cost of every possible combination.

Table 1: Items that you might take on a camping trip, together with their values, weights, and cost.

Item	Value	Weight	Cost
Knife	8	1 kg	€10
Rope	2	3 kg	€5
Book	1	1 kg	€6
Water	6	5 kg	€4
Tin of Food	4	3 kg	€4
Map	5	1 kg	€15
Tent	7	6 kg	€30
Pillow	1	2 kg	€6
Coat	6	2 kg	€9

Your job is to write a C program which finds good solutions for the knapsack problem. An important point is to evaluate whether the "brute force" approach provides a practical solution to the problem in general.

Example data files (`objectsA.txt` and `objectsB.txt`) are provided, listing the value, weight, and cost of the individual items that may go into the backpack.

Your programs need to request the filename and weight / cost limits from the user.
Suggested limits are:

- for `objectsA.txt`, the weight limit is 13 kg, and the cost limit is €45;
- for `objectsB.txt`, the weight limit is 28 kg, and the cost limit is €70.

You are required to write two programs:

(1) A program called `optpack.c`, which evaluates all the possible packing possibilities, and exhaustively finds the best solution.
(2) A program called `subpack.c` which provides a "good" solution, by whatever suboptimal technique you can devise.

Both programs should print out the packing solution obtained.

Deliverables

File name	Description
`optpack.c` `subpack.c`	C source code as described above

YOUR PROGRAMS WILL NOT BE MARKED UNLESS THEY HAVE THE CORRECT NAMES

STUDENTS MUST WORK INDEPENDENTLY. PROGRAMS WILL BE EXAMINED FOR EVIDENCE OF COPYING. COPYING OR ALLOWING YOUR WORK TO BE COPIED WILL RESULT IN A GRADE OF "NG" BEING ALLOCATED FOR THE ASSIGNMENT.