Mad Trucker

General Introduction

The challenge exercises are intended for students who are done with the regular weekly homework exercises and seek an extra challenge. Only those who passed the challenge program entry test will be considered participants of the challenge program.

Your submission for all the parts of the exercise should be in Java. All the challenge exercises must be done individually. Feel free to send an e-mail to course2ip90+CP@gmail.com if you have any questions.

Assignment description

One of your friends gets himself stranded in his truck a lot lately. The problem for this is that his brakes don't work and his gas pedal is stuck. Therefore as soon as he puts fuel into the truck it starts driving and only stops when it runs out of fuel. To help him go places he has in the back of his truck a number of fuel cans, all of different sizes, with labels on them indicating for how many kilometers they bring him. He takes estimates on how far he has to drive to get to his destination. To make things worse, there are often some places on the road, where he can't stand still to refuel.

Because he is starting to really find it annoying he asked you to help him out with this problem. He wants you to write a program that calculates in what order he has to refuel his truck to get to a destination. He wants the program to be able to take as an input the number and the sizes of the fuel cans he has in the back of his trucks, and the distances to the places where he can't stand still. Your program then has to calculate a sequence of cans with no stops in places where he can't stand still.

Again, *all unstoppable places are unique* (e.g. there are no two at the same location) *and all cans are unique* (there are no two cans with the same mileage).

Exercise

The input is a positive integer **N** (number of cans), followed by **N** positive integers. Each of these are the numbers the corresponding cans will take him in kilometers. Then there are **N-1** positive integers which are places where he can't stop in kilometers from the start.
The output is **N** integers, the zero-based indices of the cans to fill up (The first can on the list is indexed 0). The order of the output is the order in which the fuel cans have to be poured into the truck. This order is such that the truck will never stop on a spot where it can't stop.
The truck starts at 0 km, and *the target destination is the place you get after using all cans*. *You can always stop at the this destination location* (this can't be an unstoppable location). All of the unstoppable locations are between 0 km and the destination.

Constraints

You have to do better than simply trying all possible sequences of cans (brute-forcing.) This would take way too long. You have to at least be able to calculate an ordering for of **N** = 100 in no more than a few seconds. It is **mandatory solving this assignment with recursion**.

Hints

It is always possible to get the truck to some destination with all fuel used, no matter the sizes of the cans or the locations of the unstoppable stops. Not just in the test cases below but for all inputs you can generate given the restriction. Your program should essentially be a proof of this fact. This is a really important given property if you want to solve the program effectively.

Examples

Some example inputs are below. If the last case takes long you're on the wrong track with your algorithm. **Do note that for some inputs there can have multiple solutions but Momotor only checks for one**, so don't panic. If Momotor says you got it wrong, have a look at the in- and output. And as always, do your own tests to make sure it works, Momotor is just there to help you test.

To help you test your program use Java program MadTruckerChecker.java to check if your output is correct. The program takes some correct input and the output you think is correct and checks if it's true.

input (split to three lines for clarity)

```
3
2 4 6
4 8
```

output

```
0 1 2
```

a different output

```
2 1 0
```

input (split to three lines for clarity)

```
10
19 12 0 11 10 8 7 1 13 18
3 4 14 16 2 6 5 15 17
```

output

```
0 1 2 3 4 5 6 7 8 9
```

input (split to three lines for clarity)

```
20
20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 209 208 207 206 205
204 203 202 201 200 199 198 197 196 195 194 193 192 191
```

output

```
19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

Handing in the assignment

Upload into Canvas a single file named MadTrucker.java.