3 Mad Trucker

3.1 General information

Your submission for all the parts of the exercise should be in Java. All the challenge exercises must be done in pairs.

3.2 Assignment description

One of your friends gets himself stranded in his truck a lot lately. The problem 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 can fuel the truck. He takes estimates on how far he has to drive to get to his destination, and makes sure that he always has exactly enough cans, of suitable mileage, to reach that destination. To make things worse, however, there are often some places on the road where he cannot 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 without stopping at any of the places where stopping is not allowed.

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

3.3 Exercise

Write a program that takes as input:

1. The number of fuel cans in the back of the truck;
2. The mileage of each individual can (in a space-separated list);
3. The locations of the places where he cannot stop (in a space-separated list). These locations are given relative to the starting location, which is at 0 km.

Your program then has to calculate a sequence of all the cans in the back of the truck, that, when used, does not lead to the truck stopping at any of the places where stopping is not possible.

3.4 Input

The input is a positive integer n (the number of cans), followed after a space by n positive integers. Starting to count these numbers at 0, the i-th number (with i ranging from 0 to $n-1$) is the mileage of can i, indicating how many kilometers it will move the truck. Finally, after another space, $n-1$ positive integers are

given indicating the places where he cannot stop, given in kilometers measured from the start.

3.5 Output

The output consists of n positive integers, identifying an order in which the cans must be used to fuel the truck. Again, the first can is numbered 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 cannot stop.

3.6 Examples

Input	**Output**
3 2 4 6 4 8	0 1 2 or 2 1 0 (two possible outputs)
10 19 12 20 11 10 8 7 1 13 18 3 4 14 16 2 6 5 15 17	0 1 2 3 4 5 6 7 8 9 or 2 7 6 4 5 3 1 8 9 0 (two of many possibilities)
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	19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 (one of many possibilities)
40 42 6 66 96 86 59 8 83 12 18 90 26 98 9 48 55 84 24 49 70 77 27 85 41 63 57 21 38 22 14 11 94 73 47 7 82 93 64 89 54 2062 1994 1967 1963 1897 1872 1840 1821 1806 1800 1754 1598 1576 1399 1391 1263 1252 1229 1166 1098 1001 997 973 770 744 738 687 667 565 545 388 356 320 274 235 177 173 110 78	1 34 6 13 30 8 29 9 26 28 17 11 21 27 23 0 33 14 18 39 15 25 5 24 37 2 19 32 20 35 7 16 22 4 38 10 31 36 3 12
40 98 97 93 89 83 81 78 74 73 71 69 67 66 65 62 61 58 55 54 49 43 42 41 40 39 38 37 35 32 29 26 24 20 19 17 16 13 9 3 104 21 42 63 139 141 227 229 299 339 374 426 456 537 549 558 577 605 609 651 661 770 785 874 891 943 950 1050 1150 1155 1169 1289 1437 1537 1545 1694 1832 1840 1865 10000	38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 39

3.7 Specifications

1. Each input is a (single-space separated) sequence of integers between 1 and 10000. You don't need to check this.

2. Please avoid in the output any (except an empty line in the end) extra strings or any extra characters, including leading or trailing spaces.

3. Your submission is to be a single Java file.

3.8 Restrictions and recommendations

In terms of performance, try to achieve calculating an ordering for input with $n = 100$ in no more than a few seconds. It is **mandatory to solve this assignment with recursion**. Moreover, **randomly testing sequences**, i.e., a program in which sequences are randomly selected and tested for correctness, **is not allowed**, even if it involves recursion.

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 in Momotor but for all inputs you can generate given the restriction. Your program should essentially be a proof of this fact.

Note that for some inputs **there can be multiple solutions** (see the examples). We recommend you to write a small program (method?) which, using your input and output, checks that, given the former, the latter is correct.