COCI '10 Contest 1 #6 Žabe

Time limit: 1.0s **Memory limit:** 64M

The Frog Regent has arranged his N frog servants in a circle, with each frog facing the back of the next one. Each frog is assigned a unique integer identifier (ID) from the set of 1 to N. The frog arrangement is specified as a sequence of IDs. The sequence **always starts** with the frog with ID 1. It is followed by the ID of the frog in front of it, then the ID of the next one, and so on until the ID of the last frog - the one behind the frog with ID 1.

A frog is considered to have made a single leap if it has jumped over the frog in front of it, swapping places with it in the process. For example, if the frogs are sequenced as $1\ 5\ 4\ 3\ 2\ 6$ and the frog with ID 2 makes two leaps, the resulting sequence will be $1\ 2\ 5\ 4\ 3\ 6$ (the frog has shifted two places forward). When the Frog Regent proclaims the number B, the frog with ID B makes B leaps.

The Frog Regent wishes, using some number of proclamations, to rearrange the frogs from the starting sequence to his favourite resulting sequence. Given the starting and resulting frog sequences, write a program that will compute a sequence of proclamations needed for the Regent to rearrange the frogs into the resulting sequence. Naturally, the starting and resulting sequences will not be equal.

Input Specification

The first line of input contains a positive integer N, the number of frogs $(3 \leq N \leq 100)$.

The second line of input contains a permutation of the first N positive integers, the starting frog sequence.

The third line of input contains another permutation of the first N positive integers, the resulting frog sequence.

Test cases worth 40% of total points have N not greater than 8.

Output Specification

Output any sequence of integers (one integer per line) that the Frog Regent can proclaim in order to rearrange the frogs into the resulting sequence.

The number of proclamations must not exceed $100\,000$.

Note: The test data will ensure that a solution will always exist.

Sample Input 1

```
6
1 5 4 3 2 6
1 2 5 4 3 6
```

Sample Output 1

```
2
```

Sample Input 2

```
5
1 5 3 2 4
1 5 4 2 3
```

Sample Output 2

```
5
3
5
2
```