Problem F. Foreach

Time limit: 2 seconds
Memory limit: 512 megabytes

Do you like really good programming languages? Let's, for example, discuss PHP.

PHP, like many other languages, supports arrays. For purposes of this problem, we only consider an array with consecutive integer indexes starting from 0. It would be denoted as $[a_0, a_1, \ldots, a_{n-1}]$.

PHP, like many other languages, supports variables. All of them must start with `$` symbol. We will have two in this problem: array `$a`, and its element `$x`.

Also, as many other languages, PHP supports loops over all elements of array. We consider only two of four possible forms of `foreach` loop:

non-reference form: `foreach ($a as  $x) // code`
reference form: `foreach ($a as &$x) // code`

Formally, non-reference form goes through elements of the array one by one, and sets the value of `$x` to the value of the corresponding element and then executes specified code. Reference form goes though elements of the array one by one and makes `$x` a reference to the corresponding element and then executes code. So, in the first case, if one changes the value of `$x`, the value of the array element wouldn't be changed, while in the second one it would.

PHP supports a lot of other control-flow constructions. Two of them we are interested in are `if` and `break`. As in other languages, `break` stops the loop immediately, and `if` allows to execute code only if some condition holds.

Unfortunately, unlike many other languages, PHP doesn't have variable scopes. So, even after a reference form `foreach` loop, variable `$x` still would be a reference to an element of the array it was set to last time. So, changing its value after the loop would still change the element of the array. On the other side, making `$x` a reference to another element doesn't change the value of the previous element. Let's consider an example.

One has an array $[1, 2, 3]$ and executes the following code

```
foreach ($a as &$x) if ($x == 2) break;
foreach ($a as  $x) if ($x == 2) break;
```

After the first loop `$x` is a reference to the middle element of the array. On the first step of the second loop, the array becomes $[1, 1, 3]$, because the value of `$x` is changed to 1, and it is the reference to the middle element. On the second step, the value of `$x` would be changed to a new value of the next element of the array, which turned out to be 1 again. It's still not equal to 2, so on the next step, the value of `$x` changes to 3, and the resulting state of the array is $[1, 3, 3]$.

As you see, code can change arrays even having no explicit sets of variables.

Can you create a program that changes one array to another using only lines of the form shown in the example above?

Input

The first line of the input contains one integer n — the length of the array `$a` ($1 \le n \le 50$). The second line contains n integers s_i — the initial state of the array ($1 \le s_i \le 100$). The third line contains n integers t_i — the target state of the array ($1 \le t_i \le 100$).

Output

If it is not possible to change the state of array `$a` from s to t, output a single line containing a single integer -1.

Otherwise, the first line of the output must contain an integer k — the number of lines in the program ($0 \le k \le 10\,000$). You don't have to minimize this value.

Each of the following k lines should be either in the form

```
foreach ($a as &$x) if ($x == <some integer value>) break;
```

or

```
foreach ($a as  $x) if ($x == <some integer value>) break;
```

All integers should be positive and should not exceed 100. No other variables or language constructions are allowed. You should follow the format as close as possible, including whitespaces (e. g. there should be two spaces between `as` and `$x` in the non-reference form). Note that since there is no "`Presentation Error`" outcome in the contest rules, if you fail to follow these requirements, you will get "`Wrong answer`" outcome.

Your code will be executed by formal rules described in the statement. It must convert array `$a` from the initial state to the target state.

Examples

standard input	standard output

3
1 2 3
1 3 3
	2
	foreach ($a as &$x) if ($x == 2) break;
	foreach ($a as $x) if ($x == 2) break;

2 1 2 1 3	-1