A. Black-and-White Cube

time limit per test: 0.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a cube of size $k \times k \times k$, which consists of unit cubes. Two unit cubes are considered neighbouring, if they have common face.

Your task is to paint each of k^3 unit cubes one of two colours (black or white), so that the following conditions must be satisfied:

- each white cube has exactly 2 neighbouring cubes of white color;
- each black cube has exactly 2 neighbouring cubes of black color.

Input

The first line contains integer k $(1 \le k \le 100)$, which is size of the cube.

Output

Print -1 if there is no solution. Otherwise, print the required painting of the cube consequently by layers. Print a $k \times k$ matrix in the first k lines, showing how the first layer of the cube should be painted. In the following k lines print a $k \times k$ matrix — the way the second layer should be painted. And so on to the last k-th layer. Note that orientation of the cube in the space does not matter.

Mark a white unit cube with symbol "w" and a black one with "b". Use the format of output data, given in the test samples. You may print extra empty lines, they will be ignored.

Sample test(s)

input
```
1
```
output
```
-1
```

input
```
2
```
output
```
bb
ww

bb
ww
```

B. Tournament-graph

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

In this problem you have to build tournament graph, consisting of n vertices, such, that for any oriented pair of vertices (v, u) $(v \neq u)$ there exists a path from vertex v to vertex u consisting of no more then two edges.

A directed graph without self-loops is a *tournament*, if there is exactly one edge between any two distinct vertices (in one out of two possible directions).

Input

The first line contains an integer n $(3 \leq n \leq 1000)$, the number of the graph's vertices.

Output

Print `-1` if there is no graph, satisfying the described conditions.

Otherwise, print n lines with n integers in each. The numbers should be separated with spaces. That is adjacency matrix a of the found tournament. Consider the graph vertices to be numbered with integers from 1 to n. Then $a_{v,u} = 0$, if there is no edge from v to u, and $a_{v,u} = 1$ if there is one.

As the output graph has to be a tournament, following equalities must be satisfied:

- $a_{v,u} + a_{u,v} = 1$ for each v, u $(1 \leq v, u \leq n;\ v \neq u)$;
- $a_{v,v} = 0$ for each v $(1 \leq v \leq n)$.

Sample test(s)

input

```
3
```

output

```
0 1 0
0 0 1
1 0 0
```

input

```
4
```

output

```
-1
```

C. Two permutations

time limit per test: 6 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given two permutations p and q, consisting of n elements, and m queries of the form: l_1, r_1, l_2, r_2 $(l_1 \le r_1;\ l_2 \le r_2)$. The response for the query is the number of such integers from 1 to n, that their position in the first permutation is in segment $[l_1, r_1]$ (borders included), and position in the second permutation is in segment $[l_2, r_2]$ (borders included too).

A *permutation* of n elements is the sequence of n distinct integers, each not less than 1 and not greater than n.

Position of number v $(1 \le v \le n)$ in permutation $g_1, g_2, ..., g_n$ is such number i, that $g_i = v$.

Input

The first line contains one integer n $(1 \le n \le 10^6)$, the number of elements in both permutations. The following line contains n integers, separated with spaces: $p_1, p_2, ..., p_n$ $(1 \le p_i \le n)$. These are elements of the first permutation. The next line contains the second permutation $q_1, q_2, ..., q_n$ in same format.

The following line contains an integer m $(1 \le m \le 2 \cdot 10^5)$, that is the number of queries.

The following m lines contain descriptions of queries one in a line. The description of the i-th query consists of four integers: a, b, c, d $(1 \le a, b, c, d \le n)$. Query parameters l_1, r_1, l_2, r_2 are obtained from the numbers a, b, c, d using the following algorithm:

1. Introduce variable x. If it is the first query, then the variable equals 0, else it equals the response for the previous query plus one.
2. Introduce function $f(z) = ((z - 1 + x)\ mod\ n) + 1$.
3. Suppose $l_1 = min(f(a), f(b)), r_1 = max(f(a), f(b)), l_2 = min(f(c), f(d)), r_2 = max(f(c), f(d))$.

Output

Print a response for each query in a separate line.

Sample test(s)

input

```
3
3 1 2
3 2 1
1
1 2 3 3
```

output

```
1
```

input

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

output

```
1
1
2
```