A. Game with Cards

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice and Bob play a game. Alice has n cards, the i-th of them has the integer a_i written on it. Bob has m cards, the j-th of them has the integer b_j written on it.

On the first turn of the game, **the first player** chooses one of his/her cards and puts it on the table (plays it). On the second turn, **the second player** chooses one of his/her cards **such that the integer on it is greater than the integer on the card played on the first turn**, and plays it. On the third turn, **the first player** chooses one of his/her cards **such that the integer on it is greater than the integer on the card played on the second turn**, and plays it, and so on — the players take turns, and each player has to choose one of his/her cards with greater integer than the card played by the other player on the last turn.

If some player cannot make a turn, he/she loses.

For example, if Alice has 4 cards with numbers $[10, 5, 3, 8]$, and Bob has 3 cards with numbers $[6, 11, 6]$, the game may go as follows:

- Alice can choose any of her cards. She chooses the card with integer 5 and plays it.
- Bob can choose any of his cards with number greater than 5. He chooses a card with integer 6 and plays it.
- Alice can choose any of her cards with number greater than 6. She chooses the card with integer 10 and plays it.
- Bob can choose any of his cards with number greater than 10. He chooses a card with integer 11 and plays it.
- Alice can choose any of her cards with number greater than 11, but she has no such cards, so she loses.

Both Alice and Bob play **optimally (if a player is able to win the game no matter how the other player plays, the former player will definitely win the game)**.

You have to answer two questions:

- who wins if Alice is the first player?
- who wins if Bob is the first player?

Input

The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases. Each test case consists of four lines.

The first line of a test case contains one integer n ($1 \le n \le 50$) — the number of cards Alice has.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 50$) — the numbers written on the cards that Alice has.

The third line contains one integer m ($1 \le m \le 50$) — the number of Bob's cards.

The fourth line contains m integers $b_1, b_2, \dots, b_m$ ($1 \le b_i \le 50$) — the numbers on Bob's cards.

Output

For each test case, print two lines. The first line should be `Alice` if Alice wins when she is the first player; otherwise, the first line should be Bob. The second line should contain the name of the winner if Bob is the first player, in the same format.

Example

input

```
4
1
6
2
6 8
4
1 3 3 7
2
4 2
1
50
2
25 50
10
1 2 3 4 5 6 7 8 9 10
2
5 15
```

output

```
Bob
```

```
Bob
Alice
Alice
Alice
Bob
Bob
Bob
```

Note

Let's consider the first test case of the example.

Alice has one card with integer 6, Bob has two cards with numbers $[6, 8]$.

If Alice is the first player, she has to play the card with number 6. Bob then has to play the card with number 8. Alice has no cards left, so she loses.

If Bob is the first player, then no matter which of his cards he chooses on the first turn, Alice won't be able to play her card on the second turn, so she will lose.

B. Card Trick

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Monocarp has just learned a new card trick, and can't wait to present it to you. He shows you the entire deck of n cards. You see that the values of cards from the topmost to the bottommost are integers $a_1, a_2, \dots, a_n$, and all values are different.

Then he asks you to shuffle the deck m times. With the j-th shuffle, you should take b_j topmost cards and move them under the remaining $(n - b_j)$ cards without changing the order.

And then, using some magic, Monocarp tells you the topmost card of the deck. However, you are not really buying that magic. You tell him that you know the topmost card yourself. Can you surprise Monocarp and tell him the topmost card before he shows it?

Input

The first line contains a single integer t ($1 \le t \le 10^4$) — the number of testcases.

The first line of each testcase contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of cards in the deck.

The second line contains n pairwise distinct integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$) — the values of the cards.

The third line contains a single integer m ($1 \le m \le 2 \cdot 10^5$) — the number of shuffles.

The fourth line contains m integers $b_1, b_2, \dots, b_m$ ($1 \le b_j \le n - 1$) — the amount of cards that are moved on the j-th shuffle.

The sum of n over all testcases doesn't exceed $2 \cdot 10^5$. The sum of m over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase, print a single integer — the value of the card on the top of the deck after the deck is shuffled m times.

Example

input

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

output

```
2
3
3
```

Note

In the first testcase, each shuffle effectively swaps two cards. After three swaps, the deck will be $[2, 1]$.

In the second testcase, the second shuffle cancels what the first shuffle did. First, three topmost cards went underneath the last card, then that card went back below the remaining three cards. So the deck remained unchanged from the initial one — the topmost card has value 3.

C. Double Sort

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two arrays a and b, both consisting of n integers.

In one move, you can choose two indices i and j ($1 \le i, j \le n$; $i \neq j$) and swap a_i with a_j and b_i with b_j. You have to perform the swap in both arrays.

You are allowed to perform at most 10^4 moves (possibly, zero). Can you make both arrays sorted in a non-decreasing order at the end? If you can, print any sequence of moves that makes both arrays sorted.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of testcases.

The first line of each testcase contains a single integer n ($2 \le n \le 100$) — the number of elements in both arrays.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$) — the first array.

The third line contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le n$) — the second array.

Output

For each testcase, print the answer. If it's impossible to make both arrays sorted in a non-decreasing order in at most 10^4 moves, print -1. Otherwise, first, print the number of moves k $(0 \le k \le 10^4)$. Then print i and j for each move $(1 \le i, j \le n; i \neq j)$.

If there are multiple answers, then print any of them. You don't have to minimize the number of moves.

Example

input

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

output

```
0
-1
3
3 1
3 2
4 3
```

D. Required Length

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given two integer numbers, n and x. You may perform several operations with the integer x.

Each operation you perform is the following one: choose any digit y that occurs in the decimal representation of x at least once, and replace x by $x \cdot y$.

You want to make the length of decimal representation of x (without leading zeroes) equal to n. What is the minimum number of operations required to do that?

Input

The only line of the input contains two integers n and x ($2 \le n \le 19$; $1 \le x < 10^{n-1}$).

Output

Print one integer — the minimum number of operations required to make the length of decimal representation of x (without leading zeroes) equal to n, or -1 if it is impossible.

Examples

input

```
2 1
```

output

```
-1
```

input

```
3 2
```

output

```
4
```

input

```
13 42
```

output

```
12
```

Note

In the second example, the following sequence of operations achieves the goal:

1. multiply x by 2, so $x = 2 \cdot 2 = 4$;
2. multiply x by 4, so $x = 4 \cdot 4 = 16$;
3. multiply x by 6, so $x = 16 \cdot 6 = 96$;
4. multiply x by 9, so $x = 96 \cdot 9 = 864$.

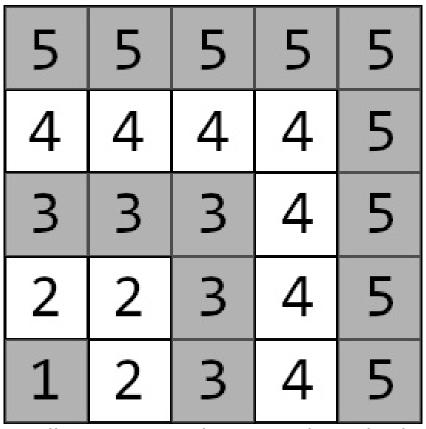
E. Labyrinth Adventures

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

You found a map of a weirdly shaped labyrinth. The map is a grid, consisting of n rows and n columns. The rows of the grid are numbered from 1 to n from bottom to top. The columns of the grid are numbered from 1 to n from left to right.

The labyrinth has n layers. The first layer is the bottom left corner (cell $(1, 1)$). The second layer consists of all cells that are in the grid and adjacent to the first layer by a side or a corner. The third layer consists of all cells that are in the grid and adjacent to the second layer by a side or a corner. And so on.

The labyrinth with 5 layers, for example, is shaped as follows:



The layers are separated from one another with walls. However, there are doors in these walls.

Each layer (except for layer n) has exactly two doors to the next layer. One door is placed on the top wall of the layer and another door is placed on the right wall of the layer. For each layer from 1 to $n - 1$ you are given positions of these two doors. The doors can be passed in both directions: either from layer i to layer $i + 1$ or from layer $i + 1$ to layer i.

If you are standing in some cell, you can move to an adjacent by a side cell if a wall doesn't block your move (e.g. you can't move to a cell in another layer if there is no door between the cells).

Now you have m queries of sort: what's the minimum number of moves one has to make to go from cell (x_1, y_1) to cell (x_2, y_2).

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the number of layers in the labyrinth.

The i-th of the next $n - 1$ lines contains four integers $d_{1,x}, d_{1,y}, d_{2,x}$ and $d_{2,y}$ ($1 \le d_{1,x}, d_{1,y}, d_{2,x}, d_{2,y} \le n$) — the coordinates of the doors. Both cells are on the i-th layer. The first cell is adjacent to the top wall of the i-th layer by a side — that side is where the door is. The second cell is adjacent to the right wall of the i-th layer by a side — that side is where the door is.

The next line contains a single integer m ($1 \le m \le 2 \cdot 10^5$) — the number of queries.

The j-th of the next m lines contains four integers x_1, y_1, x_2 and y_2 ($1 \le x_1, y_1, x_2, y_2 \le n$) — the coordinates of the cells in the j-th query.

Output

For each query, print a single integer — the minimum number of moves one has to make to go from cell (x_1, y_1) to cell (x_2, y_2).

Examples

input

```
2
1 1 1 1
10
1 1 1 1
1 1 1 2
1 1 2 1
1 1 2 2
1 2 1 2
1 2 2 1
1 2 2 2
2 1 2 1
2 1 2 2
2 2 2 2
```

output

```
0
1
1
2
0
2
1
0
1
0
```

input

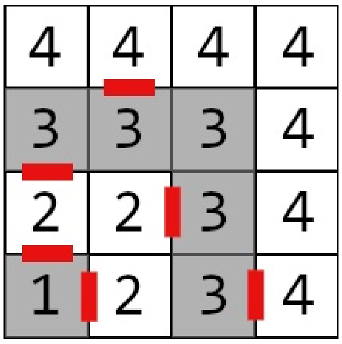
```
4
1 1 1 1
2 1 2 2
3 2 1 3
5
2 4 4 3
4 4 3 3
1 2 3 3
2 2 4 4
1 4 2 3
```

output

```
3
4
3
6
2
```

Note

Here is the map of the labyrinth from the second example. The doors are marked red.



F. Unique Occurrences

time limit per test: 6 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

You are given a tree, consisting of n vertices. Each edge has an integer value written on it.

Let $f(v, u)$ be the number of values that appear **exactly once** on the edges of a simple path between vertices v and u.

Calculate the sum of $f(v, u)$ over all pairs of vertices v and u such that $1 \le v < u \le n$.

Input

The first line contains a single integer n ($2 \le n \le 5 \cdot 10^5$) — the number of vertices in the tree.

Each of the next $n - 1$ lines contains three integers v, u and x ($1 \le v, u, x \le n$) — the description of an edge: the vertices it connects and the value written on it.

The given edges form a tree.

Output

Print a single integer — the sum of $f(v, u)$ over all pairs of vertices v and u such that $v < u$.

Examples

input

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

output

```
4
```

input

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

output

```
2
```

input

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

output

```
14
```

input

```
2
2 1 1
```

output

```
1
```

input

```
10
10 2 3
3 8 8
4 8 9
5 8 5
3 10 7
7 8 2
5 6 6
9 3 4
1 6 3
```

output

```
120
```