A. Quasi-palindrome

1 second, 256 megabytes

Let *quasi-palindromic* number be such number that adding some leading zeros (possible none) to it produces a palindromic string.

String t is called a palindrome, if it reads the same from left to right and from right to left.

For example, numbers 131 and 2010200 are *quasi-palindromic*, they can be transformed to strings "131" and "002010200", respectively, which are palindromes.

You are given some integer number x. Check if it's a *quasi-palindromic* number.

Input

The first line contains one integer number x ($1 \le x \le 10^9$). This number is given without any leading zeroes.

Output

Print "YES" if number x is *quasi-palindromic*. Otherwise, print "NO" (without quotes).

input
131

output
YES

input
320

output
NO

input
2010200

output
YES

B. Kayaking

2 seconds, 256 megabytes

Vadim is really keen on travelling. Recently he heard about kayaking activity near his town and became very excited about it, so he joined a party of kayakers.

Now the party is ready to start its journey, but firstly they have to choose kayaks. There are $2 \cdot n$ people in the group (including Vadim), and they have exactly $n - 1$ tandem kayaks (each of which, obviously, can carry two people) and 2 single kayaks. i-th person's weight is w_i, and weight is an important matter in kayaking — if the difference between the weights of two people that sit in the same tandem kayak is too large, then it can crash. And, of course, people want to distribute their seats in kayaks in order to minimize the chances that kayaks will crash.

Formally, the instability of a single kayak is always 0, and the instability of a tandem kayak is the absolute difference between weights of the people that are in this kayak. Instability of the whole journey is the total instability of all kayaks.

Help the party to determine minimum possible total instability!

Input

The first line contains one number n ($2 \le n \le 50$).

The second line contains $2 \cdot n$ integer numbers $w_1, w_2, \ldots, w_{2n}$, where w_i is weight of person i ($1 \le w_i \le 1000$).

Output

Print minimum possible total instability.

input
2 1 2 3 4

output

```
1
```

input

```
4
1 3 4 6 3 4 100 200
```

output

```
5
```

C. 1-2-3

1 second, 256 megabytes

Ilya is working for the company that constructs robots. Ilya writes programs for entertainment robots, and his current project is "Bob", a new-generation game robot. Ilya's boss wants to know his progress so far. Especially he is interested if Bob is better at playing different games than the previous model, "Alice".

So now Ilya wants to compare his robots' performance in a simple game called "1-2-3". This game is similar to the "Rock-Paper-Scissors" game: both robots secretly choose a number from the set $\{1, 2, 3\}$ and say it at the same moment. If both robots choose the same number, then it's a draw and noone gets any points. But if chosen numbers are different, then one of the robots gets a point: 3 beats 2, 2 beats 1 and 1 beats 3.

Both robots' programs make them choose their numbers in such a way that their choice in $(i + 1)$-th game depends only on the numbers chosen by them in i-th game.

Ilya knows that the robots will play k games, Alice will choose number a in the first game, and Bob will choose b in the first game. He also knows both robots' programs and can tell what each robot will choose depending on their choices in previous game. Ilya doesn't want to wait until robots play all k games, so he asks you to predict the number of points they will have after the final game.

Input

The first line contains three numbers k, a, b ($1 \le k \le 10^{18}$, $1 \le a, b \le 3$).

Then 3 lines follow, i-th of them containing 3 numbers $A_{i,1}, A_{i,2}, A_{i,3}$, where $A_{i,j}$ represents Alice's choice in the game if Alice chose i in previous game and Bob chose j ($1 \le A_{i,j} \le 3$).

Then 3 lines follow, i-th of them containing 3 numbers $B_{i,1}, B_{i,2}, B_{i,3}$, where $B_{i,j}$ represents Bob's choice in the game if Alice chose i in previous game and Bob chose j ($1 \le B_{i,j} \le 3$).

Output

Print two numbers. First of them has to be equal to the number of points Alice will have, and second of them must be Bob's score after k games.

input

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

output

```
1 9
```

input

```
8 1 1
2 2 1
3 3 1
3 1 3
1 1 1
2 1 1
1 2 3
```

output

```
5 2
```

input

```
5 1 1
1 2 2
2 2 2
2 2 2
1 2 2
2 2 2
2 2 2
```

output

```
0 0
```

In the second example game goes like this:

$$(1,1) \to (2,1) \to (3,2) \to (1,2) \to (2,1) \to (3,2) \to (1,2) \to (2,1)$$

The fourth and the seventh game are won by Bob, the first game is draw and the rest are won by Alice.

D. Yet Another Array Queries Problem

2 seconds, 256 megabytes

You are given an array a of size n, and q queries to it. There are queries of two types:

- $1\ l_i\ r_i$ — perform a cyclic shift of the segment $[l_i, r_i]$ to the right. That is, for every x such that $l_i \le x < r_i$ new value of a_{x+1} becomes equal to old value of a_x, and new value of a_{l_i} becomes equal to old value of a_{r_i};
- $2\ l_i\ r_i$ — reverse the segment $[l_i, r_i]$.

There are m important indices in the array $b_1, b_2, ..., b_m$. For each i such that $1 \le i \le m$ you have to output the number that will have index b_i in the array after all queries are performed.

Input

The first line contains three integer numbers n, q and m ($1 \le n, q \le 2 \cdot 10^5$, $1 \le m \le 100$).

The second line contains n integer numbers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$).

Then q lines follow. i-th of them contains three integer numbers t_i, l_i, r_i, where t_i is the type of i-th query, and $[l_i, r_i]$ is the segment where this query is performed ($1 \le t_i \le 2$, $1 \le l_i \le r_i \le n$).

The last line contains m integer numbers $b_1, b_2, ..., b_m$ ($1 \le b_i \le n$) — important indices of the array.

Output

Print m numbers, i-th of which is equal to the number at index b_i after all queries are done.

input

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

output

```
3 3 1 5 2
```

E. Turn Off The TV

2 seconds, 256 megabytes

Luba needs your help again! Luba has n TV sets. She knows that i-th TV set will be working from moment of time l_i till moment r_i, inclusive.

Luba wants to switch off one of TV sets in order to free the socket. Let's call some TV set *redundant* if after switching it off the number of **integer** moments of time when at least one of TV sets is working won't decrease. Luba will be very upset if she has to switch off a non-*redundant* TV set.

Help Luba by telling her the index of some *redundant* TV set. If there is no any, print `-1`.

Input

The first line contains one integer number n ($1 \le n \le 2 \cdot 10^5$) — the number of TV sets.

Then n lines follow, each of them containing two integer numbers l_i, r_i ($0 \le l_i \le r_i \le 10^9$) denoting the working time of i-th TV set.

Output

If there is no any redundant TV set, print `-1`. Otherwise print the index of any redundant TV set (TV sets are indexed from 1 to n).

If there are multiple answers, print any of them.

input

```
3
1 3
4 6
1 7
```

output

```
1
```

input

```
2
0 10
0 10
```

output

```
1
```

input

```
3
1 2
3 4
6 8
```

output

```
-1
```

input

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

output

```
2
```

Consider the first sample. Initially all integer moments of time such that at least one TV set is working are from the segment $[1;7]$. It's easy to see that this segment won't change if we switch off the first TV set (or the second one).

Note that in the fourth sample you can switch off the second TV set, since even without it all integer moments such that any of the TV sets is working denote the segment $[1;4]$.

F. Almost Permutation

3 seconds, 512 megabytes

Recently Ivan noticed an array a while debugging his code. Now Ivan can't remember this array, but the bug he was trying to fix didn't go away, so Ivan thinks that the data from this array might help him to reproduce the bug.

Ivan clearly remembers that there were n elements in the array, and each element was not less than 1 and not greater than n. Also he remembers q facts about the array. There are two types of facts that Ivan remembers:

- $1\ l_i\ r_i\ v_i$ — for each x such that $l_i \le x \le r_i\ a_x \ge v_i$;
- $2\ l_i\ r_i\ v_i$ — for each x such that $l_i \le x \le r_i\ a_x \le v_i$.

Also Ivan thinks that this array was a permutation, but he is not so sure about it. He wants to restore some array that corresponds to the q facts that he remembers and is very similar to permutation. Formally, Ivan has denoted the *cost* of array as follows:

$cost = \sum_{i=1}^{n} (cnt(i))^2$, where $cnt(i)$ is the number of occurences of i in the array.

Help Ivan to determine minimum possible *cost* of the array that corresponds to the facts!

Input

The first line contains two integer numbers n and q ($1 \le n \le 50$, $0 \le q \le 100$).

Then q lines follow, each representing a fact about the array. i-th line contains the numbers t_i, l_i, r_i and v_i for i-th fact ($1 \le t_i \le 2$, $1 \le l_i \le r_i \le n$, $1 \le v_i \le n$, t_i denotes the type of the fact).

Output

If the facts are controversial and there is no array that corresponds to them, print `-1`. Otherwise, print minimum possible *cost* of the array.

input

```
3 0
```

output

```
3
```

input

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

output

```
5
```

input

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

output

```
9
```

input

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

output

```
-1
```

G. Graphic Settings

2 seconds, 256 megabytes

Recently Ivan bought a new computer. Excited, he unpacked it and installed his favourite game. With his old computer Ivan had to choose the worst possible graphic settings (because otherwise the framerate would be really low), but now he wants to check, maybe his new computer can perform well even with the best possible graphics?

There are m graphics parameters in the game. i-th parameter can be set to any positive integer from 1 to a_i, and initially is set to b_i ($b_i \le a_i$). So there are $p = \prod_{i=1}^{m} a_i$ different combinations of parameters. Ivan can increase or decrease any of these parameters by 1; after that the game will be restarted with new parameters (and Ivan will have the opportunity to check chosen combination of parameters).

Ivan wants to try all p possible combinations. Also he wants to return to the initial settings after trying all combinations, because he thinks that initial settings can be somehow best suited for his hardware. But Ivan doesn't really want to make a lot of restarts.

So he wants you to tell the following:

- If there exists a way to make exactly p changes (each change either decreases or increases some parameter by 1) to try all possible combinations and return to initial combination, then Ivan wants to know this way.
- Otherwise, if there exists a way to make exactly p - 1 changes to try all possible combinations (including the initial one), then Ivan wants to know this way.

Help Ivan by showing him the way to change parameters!

Input

The first line of input contains one integer number m ($1 \le m \le 6$).

The second line contains m integer numbers $a_1, a_2, ..., a_m$ ($2 \le a_i \le 1000$). It is guaranteed that $p = \prod_{i=1}^{m} a_i \le 10^5$.

The third line contains m integer numbers $b_1, b_2, ..., b_m$ ($1 \le b_i \le a_i$).

Output

If there is a way to make exactly p changes (each change either decreases or increases some parameter by 1) to try all possible combinations and return to initial combination, then output `Cycle` in the first line. Then p lines must follow, each desribing a change. The line must be either `inc x` (increase parameter x by 1) or `dec x` (decrease it).

Otherwise, if there is a way to make exactly p - 1 changes to try all possible combinations (including the initial one), then output `Path` in the first line. Then p - 1 lines must follow, each describing the change the same way as mentioned above.

Otherwise, output `No`.

input

```
1
3
1
```

output

```
Path
inc 1
inc 1
```

input

```
1
3
2
```

output

```
No
```

input

```
2
3 2
1 1
```

output

```
Cycle
inc 1
inc 1
inc 2
dec 1
dec 1
dec 2
```