

**AIM Tech Round 4 (Div. 2)****A. Diversity**

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

Calculate the minimum number of characters you need to change in the string  $s$ , so that it contains at least  $k$  different letters, or print that it is impossible.

String  $s$  consists only of lowercase Latin letters, and it is allowed to change characters only to lowercase Latin letters too.

**Input**

First line of input contains string  $s$ , consisting only of lowercase Latin letters ( $1 \leq |s| \leq 1000$ ,  $|s|$  denotes the length of  $s$ ).

Second line of input contains integer  $k$  ( $1 \leq k \leq 26$ ).

**Output**

Print single line with a minimum number of necessary changes, or the word «impossible» (without quotes) if it is impossible.

**Examples**

<b>input</b>
yandex 6
<b>output</b>
0
<b>input</b>
yahoo 5
<b>output</b>
1
<b>input</b>
google 7
<b>output</b>
impossible

**Note**

In the first test case string contains 6 different letters, so we don't need to change anything.

In the second test case string contains 4 different letters:  $\{'a', 'h', 'o', 'y'\}$ . To get 5 different letters it is necessary to change one occurrence of 'o' to some letter, which doesn't occur in the string, for example,  $\{'b'\}$ .

In the third test case, it is impossible to make 7 different letters because the length of the string is 6.

## B. Rectangles

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given  $n \times m$  table. Each cell of the table is colored white or black. Find the number of non-empty sets of cells such that:

1. All cells in a set have the same color.
2. Every two cells in a set share row or column.

### Input

The first line of input contains integers  $n$  and  $m$  ( $1 \leq n, m \leq 50$ ) — the number of rows and the number of columns correspondingly.

The next  $n$  lines of input contain descriptions of rows. There are  $m$  integers, separated by spaces, in each line. The number equals 0 if the corresponding cell is colored white and equals 1 if the corresponding cell is colored black.

### Output

Output single integer — the number of non-empty sets from the problem description.

### Examples

input
1 1 0
output
1

input
2 3 1 0 1 0 1 0
output
8

### Note

In the second example, there are six one-element sets. Additionally, there are two two-element sets, the first one consists of the first and the third cells of the first row, the second one consists of the first and the third cells of the second row. To sum up, there are 8 sets.

## C. Sorting by Subsequences

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given a sequence  $a_1, a_2, \dots, a_n$  consisting of **different** integers. It is required to split this sequence into the **maximum** number of subsequences such that after sorting integers in each of them in increasing order, the total sequence also will be sorted in increasing order.

Sorting integers in a subsequence is a process such that the numbers included in a subsequence are ordered in increasing order, and the numbers which are not included in a subsequence don't change their places.

Every element of the sequence must appear in exactly one subsequence.

### Input

The first line of input data contains integer  $n$  ( $1 \leq n \leq 10^5$ ) — the length of the sequence.

The second line of input data contains  $n$  different integers  $a_1, a_2, \dots, a_n$  ( $-10^9 \leq a_i \leq 10^9$ ) — the elements of the sequence. It is guaranteed that all elements of the sequence are distinct.

### Output

In the first line print the maximum number of subsequences  $k$ , which the original sequence can be split into while fulfilling the requirements.

In the next  $k$  lines print the description of subsequences in the following format: the number of elements in subsequence  $c_i$  ( $0 < c_i \leq n$ ), then  $c_i$  integers  $l_1, l_2, \dots, l_{c_i}$  ( $1 \leq l_j \leq n$ ) — indices of these elements in the original sequence.

Indices could be printed in any order. Every index from 1 to  $n$  must appear in output **exactly once**.

If there are several possible answers, print any of them.

### Examples

input
6 3 2 1 6 5 4
output
4 2 1 3 1 2 2 4 6 1 5

input
6 83 -75 -49 11 37 62
output
1 6 1 2 3 4 5 6

### Note

In the first sample output:

After sorting the first subsequence we will get sequence 1 2 3 6 5 4.

Sorting the second subsequence changes nothing.

After sorting the third subsequence we will get sequence 1 2 3 4 5 6.

Sorting the last subsequence changes nothing.

## D. Interactive LowerBound

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

*This is an interactive problem.*

You are given a **sorted** in increasing order singly linked list. You should find the minimum integer in the list which is greater than or equal to  $x$ .

More formally, there is a singly linked list built on an array of  $n$  elements. Element with index  $i$  contains two integers:  $value_i$  is the integer value in this element, and  $next_i$  that is the index of the next element of the singly linked list (or  $-1$ , if the current element is the last). The list is sorted, i.e. if  $next_i \neq -1$ , then  $value_{next_i} > value_i$ .

You are given the number of elements in the list  $n$ , the index of the first element  $start$ , and the integer  $x$ .

You can make up to 2000 queries of the following two types:

- `? i` ( $1 \leq i \leq n$ ) — ask the values  $value_i$  and  $next_i$ ,
- `! ans` — give the answer for the problem: the minimum integer, greater than or equal to  $x$ , or  $-1$ , if there are no such integers. Your program should terminate after this query.

Write a program that solves this problem.

### Input

The first line contains three integers  $n, start, x$  ( $1 \leq n \leq 50000$ ,  $1 \leq start \leq n$ ,  $0 \leq x \leq 10^9$ ) — the number of elements in the list, the index of the first element and the integer  $x$ .

### Output

To print the answer for the problem, print `! ans`, where `ans` is the minimum integer in the list greater than or equal to  $x$ , or  $-1$ , if there is no such integer.

### Interaction

To make a query of the first type, print `? i` ( $1 \leq i \leq n$ ), where `i` is the index of element you want to know information about.

After each query of type `? i` read two integers  $value_i$  and  $next_i$  ( $0 \leq value_i \leq 10^9$ ,  $-1 \leq next_i \leq n$ ,  $next_i \neq 0$ ).

It is guaranteed that if  $next_i \neq -1$ , then  $value_{next_i} > value_i$ , and that the array values give a valid singly linked list with  $start$  being the first element.

Note that you can't ask more than 1999 queries of the type `? i`.

If  $next_i = -1$  and  $value_i = -1$ , then it means that you asked more queries than allowed, or asked an invalid query. Your program should immediately terminate (for example, by calling `exit(0)`). You will receive "Wrong Answer", it means that you asked more queries than allowed, or asked an invalid query. If you ignore this, you can get other verdicts since your program will continue to read from a closed stream.

Your solution will get "Idleness Limit Exceeded", if you don't print anything or forget to `flush` the output, including the final answer.

To `flush` you can use (just after printing a query and line end):

- `fflush(stdout)` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;
- `flush(output)` in Pascal;
- For other languages see documentation.

### Hacks format

For hacks, use the following format:

In the first line print three integers  $n, start, x$  ( $1 \leq n \leq 50000$ ,  $1 \leq start \leq n$ ,  $0 \leq x \leq 10^9$ ).

In the next  $n$  lines print the description of the elements of the list: in the  $i$ -th line print two integers  $value_i$  and  $next_i$  ( $0 \leq value_i \leq 10^9$ ,  $-1 \leq next_i \leq n$ ,  $next_i \neq 0$ ).

The printed structure should be a valid singly linked list. In particular, it should be possible to reach all elements from  $start$  by following links  $next_i$ , and the last element  $end$  should have  $-1$  in the  $next_{end}$ .

### Example

input
5 3 80 97 -1 58 5 16 2 81 1

79 4

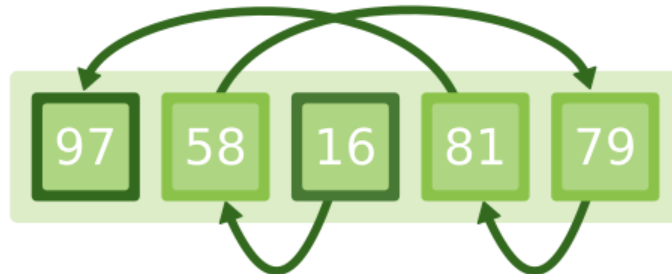
## output

? 1  
? 2  
? 3  
? 4  
? 5  
! 81

### Note

You can read more about singly linked list by the following link: [https://en.wikipedia.org/wiki/Linked\\_list#Singly\\_linked\\_list](https://en.wikipedia.org/wiki/Linked_list#Singly_linked_list)

The illustration for the first sample case. Start and finish elements are marked dark.



## E. Upgrading Tree

time limit per test: 4 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

You are given a tree with  $n$  vertices and you are allowed to perform **no more than**  $2n$  transformations on it. Transformation is defined by three vertices  $x, y, y'$  and consists of deleting edge  $(x, y)$  and adding edge  $(x, y')$ . Transformation  $x, y, y'$  could be performed if all the following conditions are satisfied:

1. There is an edge  $(x, y)$  in the current tree.
2. After the transformation the graph remains a tree.
3. After the deletion of edge  $(x, y)$  the tree would consist of two connected components. Let's denote the set of nodes in the component containing vertex  $x$  by  $V_x$ , and the set of nodes in the component containing vertex  $y$  by  $V_y$ . Then condition  $|V_x| > |V_y|$  should be satisfied, i.e. the size of the component with  $x$  should be strictly larger than the size of the component with  $y$ .

You should **minimize** the sum of squared distances between all pairs of vertices in a tree, which you could get after no more than  $2n$  transformations and output any sequence of transformations leading initial tree to such state.

Note that you don't need to minimize the number of operations. It is necessary to minimize only the sum of the squared distances.

### Input

The first line of input contains integer  $n$  ( $1 \leq n \leq 2 \cdot 10^5$ ) — number of vertices in tree.

The next  $n - 1$  lines of input contains integers  $a$  and  $b$  ( $1 \leq a, b \leq n, a \neq b$ ) — the descriptions of edges. It is guaranteed that the given edges form a tree.

### Output

In the first line output integer  $k$  ( $0 \leq k \leq 2n$ ) — the number of transformations from your example, **minimizing** sum of squared distances between all pairs of vertices.

In each of the next  $k$  lines output three integers  $x, y, y'$  — indices of vertices from the corresponding transformation.

Transformations with  $y = y'$  are allowed (even though they don't change tree) if transformation conditions are satisfied.

If there are several possible answers, print any of them.

### Examples

input
3 3 2 1 3
output
0

input
7 1 2 2 3 3 4 4 5 5 6 6 7
output
2 4 3 2 4 5 6

### Note

This is a picture for the second sample. Added edges are dark, deleted edges are dotted.

