Codeforces Round #649 (Div. 2)

A. XXXXX

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

Ehab loves number theory, but for some reason he hates the number x. Given an array a, find the length of its longest subarray such that the sum of its elements **isn't** divisible by x, or determine that such subarray doesn't exist.

An array a is a subarray of an array b if a can be obtained from b by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Input

The first line contains an integer t $(1 \le t \le 5)$ — the number of test cases you need to solve. The description of the test cases follows.

The first line of each test case contains 2 integers n and x $(1 \le n \le 10^5, 1 \le x \le 10^4)$ — the number of elements in the array a and the number that Ehab hates.

The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 10^4)$ — the elements of the array a.

Output

For each testcase, print the length of the longest subarray whose sum isn't divisible by x. If there's no such subarray, print -1.

Example

input
3 3 3 1 2 3 3 4 1 2 3 2 2 0 6
output
2 3 -1

Note

In the first test case, the subarray $[2, 3]$ has sum of elements 5, which isn't divisible by 3.

In the second test case, the sum of elements of the whole array is 6, which isn't divisible by 4.

In the third test case, all subarrays have an even sum, so the answer is -1.

B. Most socially-distanced subsequence

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

Given a permutation p of length n, find its subsequence $s_1, s_2, \ldots, s_k$ of length at least 2 such that:

- $|s_1 - s_2| + |s_2 - s_3| + \ldots + |s_{k-1} - s_k|$ is as big as possible over all subsequences of p with length at least 2.
- Among all such subsequences, choose the one whose length, k, is as small as possible.

If multiple subsequences satisfy these conditions, you are allowed to find any of them.

A sequence a is a subsequence of an array b if a can be obtained from b by deleting some (possibly, zero or all) elements.

A permutation of length n is an array of length n in which every element from 1 to n occurs exactly once.

Input

The first line contains an integer t $(1 \le t \le 2 \cdot 10^4)$ — the number of test cases. The description of the test cases follows.

The first line of each test case contains an integer n $(2 \le n \le 10^5)$ — the length of the permutation p.

The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$, p_i are distinct) — the elements of the permutation p.

The sum of n across the test cases doesn't exceed 10^5.

Output
For each test case, the first line should contain the length of the found subsequence, k. The second line should contain $s_1, s_2, \ldots, s_k$ — its elements.

If multiple subsequences satisfy these conditions, you are allowed to find any of them.

Example

input
2 3 3 2 1 4 1 3 4 2
output
2 3 1 3 1 4 2

Note
In the first test case, there are 4 subsequences of length at least 2:

- $[3, 2]$ which gives us $|3 - 2| = 1$.
- $[3, 1]$ which gives us $|3 - 1| = 2$.
- $[2, 1]$ which gives us $|2 - 1| = 1$.
- $[3, 2, 1]$ which gives us $|3 - 2| + |2 - 1| = 2$.

So the answer is either $[3, 1]$ or $[3, 2, 1]$. Since we want the subsequence to be as short as possible, the answer is $[3, 1]$.

C. Ehab and Prefix MEXs

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

Given an array a of length n, find another array, b, of length n such that:

- for each i $(1 \le i \le n)$ $MEX(\{b_1, b_2, \ldots, b_i\}) = a_i$.

The MEX of a set of integers is the smallest non-negative integer that doesn't belong to this set.

If such array doesn't exist, determine this.

Input
The first line contains an integer n ($1 \le n \le 10^5$) — the length of the array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le i$) — the elements of the array a. **It's guaranteed that $a_i \le a_{i+1}$ for $1 \le i < n$.**

Output
If there's no such array, print a single line containing -1.

Otherwise, print a single line containing n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^6$)

If there are multiple answers, print any.

Examples

input
3 1 2 3
output
0 1 2

input
4 0 0 0 2
output
1 3 4 0

input
3 1 1 3
output
0 2 1

Note
In the second test case, other answers like $[1, 1, 1, 0]$, for example, are valid.

D. Ehab's Last Corollary

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

Given a connected undirected graph with n vertices and an integer k, you have to either:

- either find an independent set that has **exactly** $\lceil \frac{k}{2} \rceil$ vertices.
- or find a **simple** cycle of length **at most** k.

An independent set is a set of vertices such that no two of them are connected by an edge. A simple cycle is a cycle that doesn't contain any vertex twice.

I have a proof that for any input you can always solve at least one of these problems, but it's left as an exercise for the reader.

Input
The first line contains three integers n, m, and k ($3 \leq k \leq n \leq 10^5$, $n - 1 \leq m \leq 2 \cdot 10^5$) — the number of vertices and edges in the graph, and the parameter k from the statement.

Each of the next m lines contains two integers u and v ($1 \leq u, v \leq n$) that mean there's an edge between vertices u and v. It's guaranteed that the graph is connected and doesn't contain any self-loops or multiple edges.

Output
If you choose to solve the first problem, then on the first line print 1, followed by a line containing $\lceil \frac{k}{2} \rceil$ distinct integers not exceeding n, the vertices in the desired independent set.

If you, however, choose to solve the second problem, then on the first line print 2, followed by a line containing one integer, c, representing the length of the found cycle, followed by a line containing c distinct integers not exceeding n, the vertices in the desired cycle, **in the order they appear in the cycle**.

Examples

input
4 4 3 1 2 2 3 3 4 4 1
output
1 1 3

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

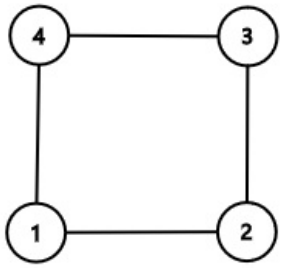
input
4 6 3 1 2 2 3 3 4 4 1 1 3 2 4
output
2

```
3
1 2 3
```

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

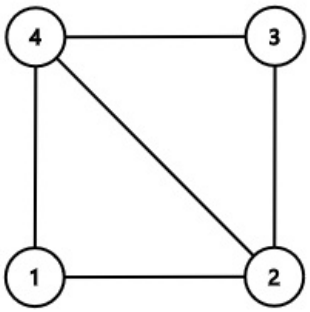
Note

In the first sample:



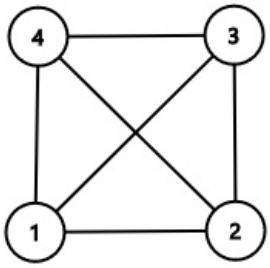
Notice that printing the independent set $\{2, 4\}$ is also OK, but printing the cycle $1 - 2 - 3 - 4$ isn't, because its length must be at most 3.

In the second sample:

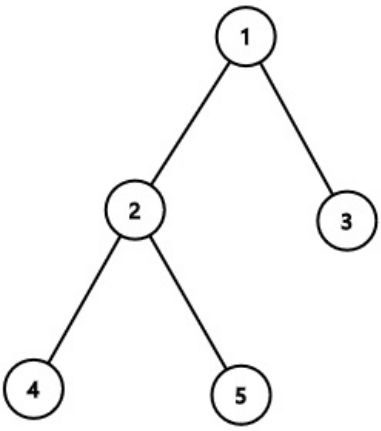


Notice that printing the independent set $\{1, 3\}$ or printing the cycle $2 - 1 - 4$ is also OK.

In the third sample:



In the fourth sample:



E. X-OR

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

This is an interactive problem!

Ehab has a hidden permutation p of length n consisting of the elements from 0 to $n-1$. You, for some reason, want to figure out the permutation. To do that, you can give Ehab 2 **different** indices i and j, and he'll reply with $(p_i|p_j)$ where $|$ is the bitwise-or operation.

Ehab has just enough free time to answer 4269 questions, and while he's OK with answering that many questions, he's too lazy to play your silly games, so he'll fix the permutation beforehand and will **not** change it depending on your queries. Can you guess the permutation?

Input

The only line contains the integer n $(3 \le n \le 2048)$ — the length of the permutation.

Interaction

To ask a question, print "? i j" (without quotes, $i \neq j$) Then, you should read the answer, which will be $(p_i|p_j)$.

If we answer with -1 instead of a valid answer, that means you exceeded the number of queries or made an invalid query.

Exit immediately after receiving -1 and you will see `wrong answer` verdict. Otherwise, you can get an arbitrary verdict because your solution will continue to read from a closed stream.

To print the answer, print "! p_1 p_2 $\ldots$ p_n" (without quotes). **Note that answering doesn't count as one of the 4269 queries**.

After printing a query or printing the answer, do not forget to output end of line and flush the output. Otherwise, you will get `idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- See the documentation for other languages.

Hacks:

The first line should contain the integer n ($3 \leq n \leq 2^{11}$) — the length of the permutation p.

The second line should contain n space-separated integers p_1, p_2, ..., p_n ($0 \leq p_i < n$) — the elements of the permutation p.

Example

input
3 1 3 2
output
? 1 2 ? 1 3 ? 2 3 ! 1 0 2

Note

In the first sample, the permutation is $[1, 0, 2]$. You start by asking about $p_1 | p_2$ and Ehab replies with 1. You then ask about $p_1 | p_3$ and Ehab replies with 3. Finally, you ask about $p_2 | p_3$ and Ehab replies with 2. You then guess the permutation.