Codeforces Round #669 (Div. 2)

A. Ahahahahahahahaha

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

Alexandra has an even-length array a, consisting of 0s and 1s. The elements of the array are enumerated from 1 to n. She wants to remove **at most** $\frac{n}{2}$ elements (where n — length of array) in the way that alternating sum of the array will be equal 0 (i.e. $a_1 - a_2 + a_3 - a_4 + \dots = 0$). In other words, Alexandra wants sum of all elements at the odd positions and sum of all elements at the even positions to become equal. The elements that you remove don't have to be consecutive.

For example, if she has $a = [1, 0, 1, 0, 0, 0]$ and she removes 2nd and 4th elements, a will become equal $[1, 1, 0, 0]$ and its alternating sum is $1 - 1 + 0 - 0 = 0$.

Help her!

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). Description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 10^3$, n **is even**) — length of the array.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 1$) — elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed 10^3.

Output

For each test case, firstly, print k ($\frac{n}{2} \le k \le n$) — number of elements that will remain after removing **in the order they appear in** a. Then, print this k numbers. Note that you should print the numbers themselves, **not their indices**.

We can show that an answer always exists. If there are several answers, you can output any of them.

Example

input

```
4
2
1 0
2
0 0
4
0 1 1 1
4
1 1 0 0
```

output

```
1
0
1
0
2
1 1
4
1 1 0 0
```

Note

In the first and second cases, alternating sum of the array, obviously, equals 0.

In the third case, alternating sum of the array equals $1 - 1 = 0$.

In the fourth case, alternating sum already equals $1 - 1 + 0 - 0 = 0$, so we don't have to remove anything.

B. Big Vova

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

Alexander is a well-known programmer. Today he decided to finally go out and play football, but with the first hit he left a dent on the new Rolls-Royce of the wealthy businessman Big Vova. Vladimir has recently opened a store on the popular online marketplace

"Zmey-Gorynych", and offers Alex a job: if he shows his programming skills by solving a task, he'll work as a cybersecurity specialist. Otherwise, he'll be delivering some doubtful products for the next two years.

You're given n positive integers $a_1, a_2, \dots, a_n$. Using each of them **exactly at once**, you're to make such sequence $b_1, b_2, \dots, b_n$ that sequence $c_1, c_2, \dots, c_n$ is *lexicographically maximal*, where $c_i = GCD(b_1, \dots, b_i)$ - the greatest common divisor of the first i elements of b.

Alexander is really afraid of the conditions of this simple task, so he asks you to solve it.

A sequence a is lexicographically smaller than a sequence b if and only if one of the following holds:

- a is a prefix of b, but $a \neq b$;
- in the first position where a and b differ, the sequence a has a smaller element than the corresponding element in b.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). Description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 10^3$) — the length of the sequence a.

The second line of each test case contains n integers $a_1, \dots, a_n$ ($1 \le a_i \le 10^3$) — the sequence a.

It is guaranteed that the sum of n over all test cases does not exceed 10^3.

Output

For each test case output the answer in a single line — the desired sequence b. If there are multiple answers, print any.

Example

input

```
7
2
2 5
4
1 8 2 3
3
3 8 9
5
64 25 75 100 50
1
42
6
96 128 88 80 52 7
5
2 4 8 16 17
```

output

```
5 2
8 2 1 3
9 3 8
100 50 25 75 64
42
128 96 80 88 52 7
17 2 4 8 16
```

Note

In the first test case of the example, there are only two possible permutations b — $[2, 5]$ and $[5, 2]$: for the first one $c = [2, 1]$, for the second one $c = [5, 1]$.

In the third test case of the example, number 9 should be the first in b, and $GCD(9, 3) = 3$, $GCD(9, 8) = 1$, so the second number of b should be 3.

In the seventh test case of the example, first four numbers pairwise have a common divisor (a power of two), but none of them can be the first in the optimal permutation b.

C. Chocolate Bunny

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

This is an interactive problem.

We hid from you a permutation p of length n, consisting of the elements from 1 to n. You want to guess it. To do that, you can give us 2 different indices i and j, and we will reply with $p_i \bmod p_j$ (remainder of division p_i by p_j).

We have enough patience to answer at most $2 \cdot n$ queries, so you should fit in this constraint. Can you do it?

As a reminder, a permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation (

$n = 3$ but there is 4 in the array).

Input

The only line of the input contains a single integer n ($1 \le n \le 10^4$) — length of the permutation.

Interaction

The interaction starts with reading n.

Then you are allowed to make at most $2 \cdot n$ queries in the following way:

- "`? x y`" ($1 \le x, y \le n, x \ne y$).

After each one, you should read an integer k, that equals $p_x \bmod p_y$.

When you have guessed the permutation, print a single line "`! `" (without quotes), followed by array p and quit.

After printing a query 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 documentation for other languages.

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.

Hack format

In the first line output n ($1 \le n \le 10^4$). In the second line print the permutation of n integers $p_1, p_2, \ldots, p_n$.

Example

input

```
3

1

2

1

0
```

output

```
? 1 2

? 3 2

? 1 3

? 2 1

! 1 3 2
```

D. Discrete Centrifugal Jumps

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

There are n beautiful skyscrapers in New York, the height of the i-th one is h_i. Today some villains have set on fire first $n-1$ of them, and now the only safety building is n-th skyscraper.

Let's call a jump from i-th skyscraper to j-th ($i < j$) **discrete**, if all skyscrapers between are strictly lower or higher than both of them. Formally, jump is discrete, if $i < j$ and one of the following conditions satisfied:

- $i + 1 = j$
- $\max(h_{i+1}, \ldots, h_{j-1}) < \min(h_i, h_j)$
- $\max(h_i, h_j) < \min(h_{i+1}, \ldots, h_{j-1})$.

At the moment, Vasya is staying on the first skyscraper and wants to live a little longer, so his goal is to reach n-th skyscraper with minimal count of discrete jumps. Help him with calcualting this number.

Input

The first line contains a single integer n ($2 \le n \le 3 \cdot 10^5$) — total amount of skyscrapers.

The second line contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 10^9$) — heights of skyscrapers.

Output

Print single number k — minimal amount of discrete jumps. We can show that an answer always exists.

Examples

input

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

output

```
3
```

input

```
4
4 2 2 4
```

output

```
1
```

input

```
2
1 1
```

output

```
1
```

input

```
5
100 1 100 1 100
```

output

```
2
```

Note

In the first testcase, Vasya can jump in the following way: $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$.

In the second and third testcases, we can reach last skyscraper in one jump.

Sequence of jumps in the fourth testcase: $1 \rightarrow 3 \rightarrow 5$.

E. Egor in the Republic of Dagestan

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

Egor is a famous Russian singer, rapper, actor and blogger, and finally he decided to give a concert in the sunny Republic of Dagestan.

There are n cities in the republic, some of them are connected by m directed roads without any additional conditions. In other words, road system of Dagestan represents *an arbitrary directed graph*. Egor will arrive to the city 1, travel to the city n by roads along some path, give a concert and fly away.

As any famous artist, Egor has lots of haters and too annoying fans, so he can travel only by safe roads. There are two types of the roads in Dagestan, black and white: black roads are safe at night only, and white roads — in the morning. Before the trip Egor's manager's going to make a schedule: for each city he'll specify it's color, black or white, and then if during the trip they visit some city, the only time they can leave it is determined by the city's color: night, if it's black, and morning, if it's white. After creating the schedule Egor chooses an available path from 1 to n, and for security reasons it has to be the shortest possible.

Egor's manager likes Dagestan very much and wants to stay here as long as possible, so he asks you to make such schedule that there would be no path from 1 to n or the shortest path's length would be greatest possible.

A *path* is one city or a sequence of roads such that for every road (excluding the first one) the city this road goes from is equal to the city previous road goes into. Egor can move only along paths consisting of safe roads only.

The path length is equal to the number of roads in it. The shortest path in a graph is a path with smallest length.

Input

The first line contains two integers n, m ($1 \leq n \leq 500000$, $0 \leq m \leq 500000$) — the number of cities and the number of roads.

The i-th of next m lines contains three integers — u_i, v_i and t_i ($1 \leq u_i, v_i \leq n$, $t_i \in \{0, 1\}$) — numbers of cities connected by road and its type, respectively (0 — night road, 1 — morning road).

Output

In the first line output the length of the desired path (or -1, if it's possible to choose such schedule that there's no path from 1 to n).

In the second line output the desired schedule — a string of n digits, where i-th digit is 0, if the i-th city is a night one, and 1 if it's a morning one.

If there are multiple answers, print any.

Examples

input

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

output

```
2
011
```

input

```
4 8
1 1 0
1 3 0
1 3 1
3 2 0
2 1 0
3 4 1
2 4 0
2 4 1
```

output

```
3
1101
```

input

```
5 10
1 2 0
1 3 1
1 4 0
2 3 0
2 3 1
2 5 0
3 4 0
3 4 1
4 2 1
4 5 0
```

output

```
-1
11111
```

Note

For the first sample, if we paint city 1 white, the shortest path is $1 \to 3$. Otherwise, it's $1 \to 2 \to 3$ regardless of other cities' colors.

For the second sample, we should paint city 3 black, and there are both black and white roads going from 2 to 4. Note that there can be a road connecting a city with itself.