Codeforces Round #618 (Div. 1)

A. Anu Has a Function

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

Anu has created her own function f: $f(x, y) = (x|y) - y$ where $|$ denotes the bitwise OR operation. For example, $f(11, 6) = (11|6) - 6 = 15 - 6 = 9$. It can be proved that for any nonnegative numbers x and y value of $f(x, y)$ is also nonnegative.

She would like to research more about this function and has created multiple problems for herself. But she isn't able to solve all of them and needs your help. Here is one of these problems.

A value of an array $[a_1, a_2, \ldots, a_n]$ is defined as $f(f(\ldots f(f(a_1, a_2), a_3), \ldots a_{n-1}), a_n)$ (see notes). You are given an array with **not necessarily distinct** elements. How should you reorder its elements so that the value of the array is maximal possible?

Input

The first line contains a single integer n ($1 \le n \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$). Elements of the array are **not guaranteed** to be different.

Output

Output n integers, the reordering of the array with maximum value. If there are multiple answers, print any.

Examples

input
4 4 0 11 6
output
11 6 4 0

input
1 13
output
13

Note

In the first testcase, value of the array $[11, 6, 4, 0]$ is $f(f(f(11, 6), 4), 0) = f(f(9, 4), 0) = f(9, 0) = 9$.

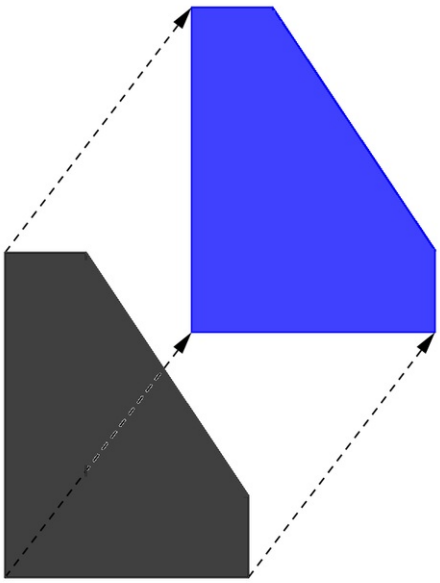
$[11, 4, 0, 6]$ is also a valid answer.

B. Aerodynamic

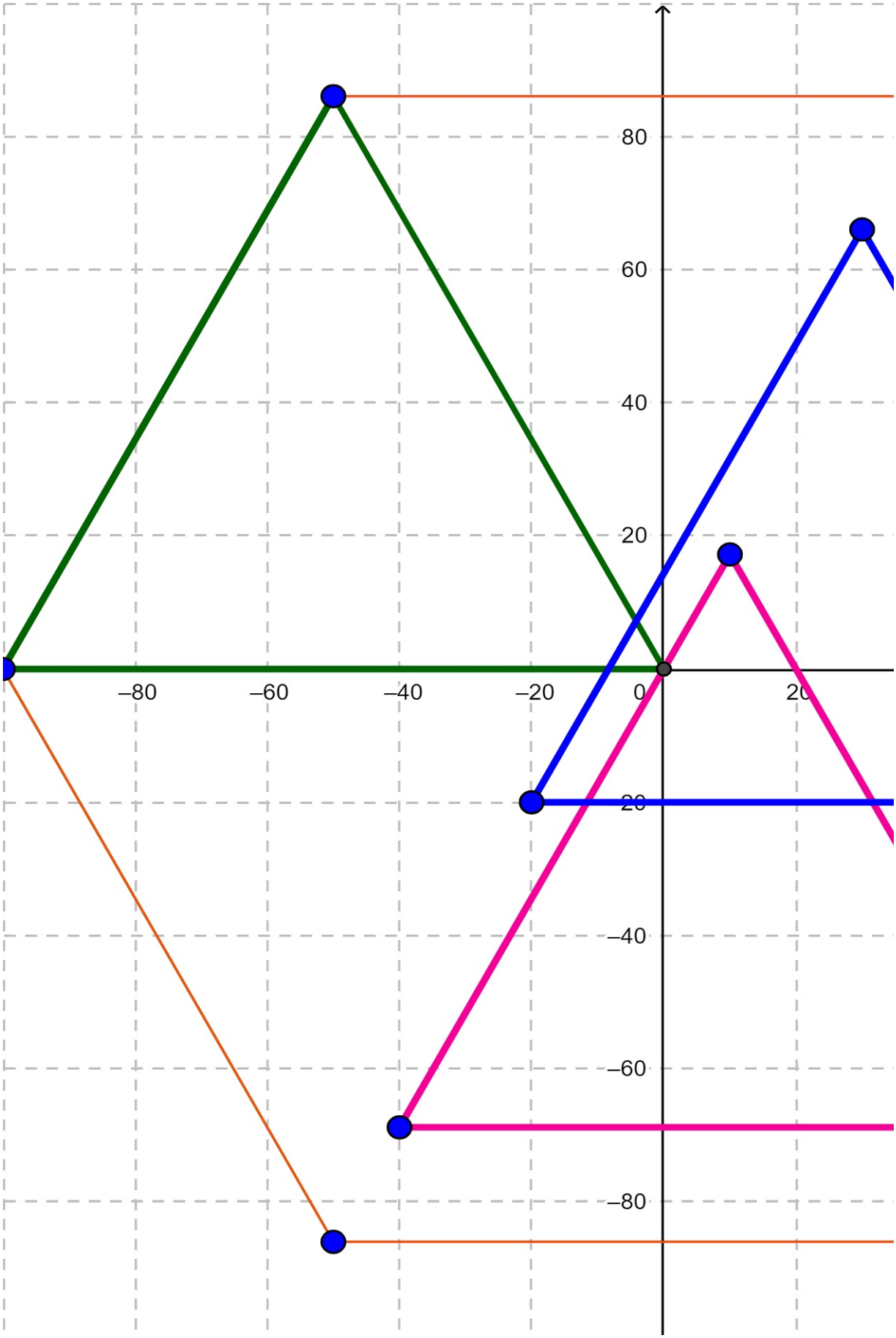
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Guy-Manuel and Thomas are going to build a polygon spaceship.

You're given a **strictly convex** (i. e. no three points are collinear) polygon P which is defined by coordinates of its vertices. Define $P(x, y)$ as a polygon obtained by translating P by vector $\overrightarrow{(x, y)}$. The picture below depicts an example of the translation:



Define T as a set of points which is the union of all $P(x, y)$ such that the origin $(0, 0)$ lies in $P(x, y)$ (both strictly inside and on the boundary). There is also an equivalent definition: a point (x, y) lies in T only if there are two points A, B in P such that $\overrightarrow{AB} = \overrightarrow{(x, y)}$. One can prove T is a polygon too. For example, if P is a regular triangle then T is a regular hexagon. At the picture below P is drawn in black and some $P(x, y)$ which contain the origin are drawn in colored:



The spaceship has the best aerodynamic performance if P and T are similar. Your task is to check whether the polygons P and T are similar.

Input

The first line of input will contain a single integer n ($3 \le n \le 10^5$) — the number of points.

The i-th of the next n lines contains two integers x_i, y_i ($|x_i|, |y_i| \le 10^9$), denoting the coordinates of the i-th vertex.

It is guaranteed that these points are listed in counterclockwise order and these points form a strictly convex polygon.

Output

Output "YES" in a separate line, if P and T are similar. Otherwise, output "NO" in a separate line. You can print each letter in any case (upper or lower).

Examples

input
4 1 0 4 1 3 4

```
0 3
```

output

```
YES
```

input

```
3
100 86
50 0
150 0
```

output

```
nO
```

input

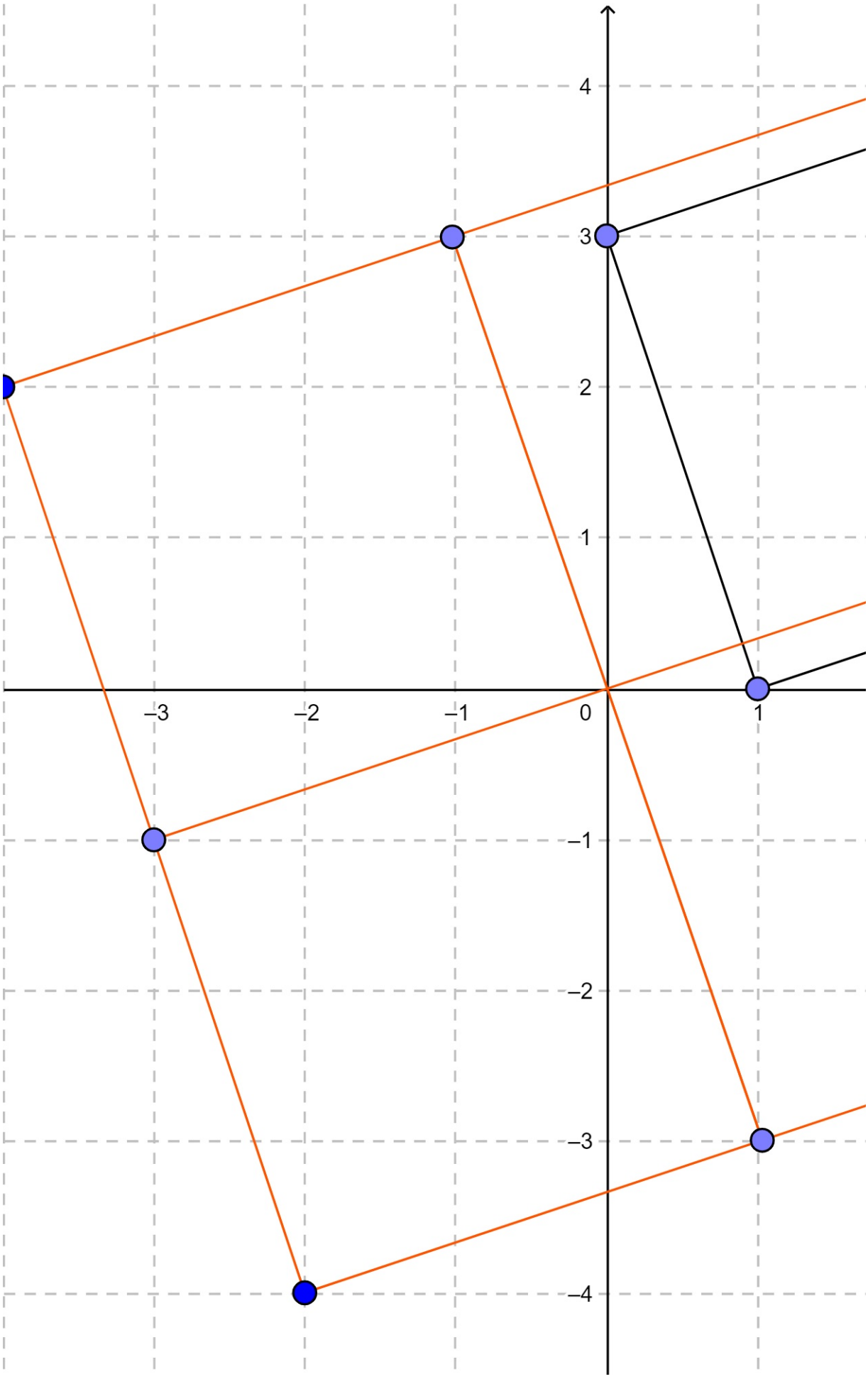
```
8
0 0
1 0
2 1
3 3
4 6
3 6
2 5
1 3
```

output

```
YES
```

Note

The following image shows the first sample: both P and T are squares. The second sample was shown in the statements.



C. Water Balance

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n water tanks in a row, i-th of them contains a_i liters of water. The tanks are numbered from 1 to n from left to right.

You can perform the following operation: choose some subsegment $[l, r]$ $(1 \le l \le r \le n)$, and redistribute water in tanks $l, l+1, \ldots, r$ evenly. In other words, replace each of $a_l, a_{l+1}, \ldots, a_r$ by $\frac{a_l + a_{l+1} + \cdots + a_r}{r-l+1}$. For example, if for volumes $[1, 3, 6, 7]$ you choose $l = 2, r = 3$, new volumes of water will be $[1, 4.5, 4.5, 7]$. **You can perform this operation any number of times**.

What is the lexicographically smallest sequence of volumes of water that you can achieve?

As a reminder:

A sequence a is lexicographically smaller than a sequence b of the same length if and only if the following holds: in the first (leftmost) position where a and b differ, the sequence a has a smaller element than the corresponding element in b.

Input
The first line contains an integer n ($1 \le n \le 10^6$) — the number of water tanks.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — initial volumes of water in the water tanks, in liters.

Because of large input, reading input as doubles **is not recommended**.

Output
Print the lexicographically smallest sequence you can get. In the i-th line print the final volume of water in the i-th tank.

Your answer is considered correct if the absolute or relative error of each a_i does not exceed 10^{-9}.

Formally, let your answer be $a_1, a_2, \ldots, a_n$, and the jury's answer be $b_1, b_2, \ldots, b_n$. Your answer is accepted if and only if $\frac{|a_i - b_i|}{\max(1, |b_i|)} \le 10^{-9}$ for each i.

Examples

input
4 7 5 5 7

output
5.666666667 5.666666667 5.666666667 7.000000000

input
5 7 8 8 10 12

output
7.000000000 8.000000000 8.000000000 10.000000000 12.000000000

input
10 3 9 5 5 1 7 5 3 8 7

output
3.000000000 5.000000000 5.000000000 5.000000000 5.000000000 5.000000000 5.000000000 5.000000000 7.500000000 7.500000000

Note
In the first sample, you can get the sequence by applying the operation for subsegment $[1, 3]$.

In the second sample, you can't get any lexicographically smaller sequence.

D. Around the World

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

Guy-Manuel and Thomas are planning 144 *trips around the world.*

You are given a simple weighted undirected connected graph with n vertexes and m edges with the following restriction: there isn't any simple cycle (i. e. a cycle which doesn't pass through any vertex more than once) of length greater than 3 which passes through the vertex 1. The cost of a path (not necessarily simple) in this graph is defined as the XOR of weights of all edges in that path with each edge being counted as many times as the path passes through it.

But the trips with cost 0 *aren't exciting.*

You may choose any subset of edges incident to the vertex 1 and remove them. How many are there such subsets, that, when removed, there is not any nontrivial cycle with the cost equal to 0 which passes through the vertex 1 in the resulting graph? A cycle is called *nontrivial* if it passes through some edge odd number of times. As the answer can be very big, output it modulo $10^9 + 7$.

Input
The first line contains two integers n and m ($1 \le n, m \le 10^5$) — the number of vertexes and edges in the graph. The i-th of the next m lines contains three integers a_i, b_i and w_i ($1 \le a_i, b_i \le n, a_i \ne b_i, 0 \le w_i < 32$) — the endpoints of the i-th edge and its weight. It's guaranteed there aren't any multiple edges, the graph is connected and there isn't any simple cycle of length greater than 3 which passes through the vertex 1.

Output
Output the answer modulo $10^9 + 7$.

Examples

input
6 8 1 2 0 2 3 1 2 4 3 2 6 2 3 4 8 3 5 4 5 4 5 5 6 6

output
2

input
7 9 1 2 0 1 3 1 2 3 9 2 4 3 2 5 4 4 5 7 3 6 6 3 7 7 6 7 8

output
1

input
4 4 1 2 27

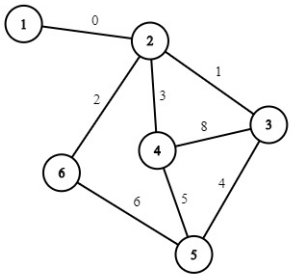
```
1 3 1
1 4 1
3 4 0
```

output

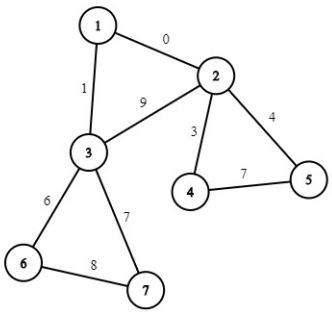
```
6
```

Note

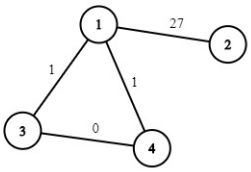
The pictures below represent the graphs from examples.



In the first example, there aren't any nontrivial cycles with cost 0, so we can either remove or keep the only edge incident to the vertex 1.



In the second example, if we don't remove the edge $1-2$, then there is a cycle $1-2-4-5-2-1$ with cost 0; also if we don't remove the edge $1-3$, then there is a cycle $1-3-2-4-5-2-3-1$ of cost 0. The only valid subset consists of both edges.



In the third example, all subsets are valid except for those two in which both edges $1-3$ and $1-4$ are kept.

E. So Mean

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is interactive.

We have hidden a permutation $p_1, p_2, \ldots, p_n$ of numbers from 1 to n from you, where n **is even**. You can try to guess it using the following queries:

$?\ k\ a_1\ a_2\ \ldots\ a_k$.

In response, you will learn if the average of elements with indexes $a_1, a_2, \ldots, a_k$ is an integer. In other words, you will receive 1 if $\frac{p_{a_1}+p_{a_2}+\cdots+p_{a_k}}{k}$ is integer, and 0 otherwise.

You have to guess the permutation. You can ask **not more than** $18n$ **queries**.

Note that permutations $[p_1, p_2, \ldots, p_k]$ and $[n+1-p_1, n+1-p_2, \ldots, n+1-p_k]$ are indistinguishable. Therefore, **you are guaranteed that** $p_1 \le \frac{n}{2}$.

Note that the permutation p is fixed before the start of the interaction and doesn't depend on your queries. In other words, **interactor is not adaptive**.

Note that you don't have to minimize the number of queries.

Input

The first line contains a single integer n ($2 \le n \le 800$, n **is even**).

Interaction

You begin the interaction by reading n.

To ask a question about elements on positions $a_1, a_2, \ldots, a_k$, in a separate line output

$?\ k\ a_1\ a_2\ ...\ a_k$

Numbers in the query have to satisfy $1 \le a_i \le n$, and all a_i have to be different. Don't forget to 'flush', to get the answer.

In response, you will receive 1 if $\frac{p_{a_1}+p_{a_2}+\cdots+p_{a_k}}{k}$ is integer, and 0 otherwise.

In case your query is invalid or you asked more than $18n$ queries, the program will print -1 and will finish interaction. You will receive a **Wrong answer** verdict. Make sure to exit immediately to avoid getting other verdicts.

When you determine permutation, output

! p_1 p_2 ... p_n

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.

Hack format

For the hacks use the following format:

The first line has to contain a single integer n ($2 \le n \le 800$, n **is even**).

In the next line output n integers $p_1, p_2, \ldots, p_n$ — the valid permutation of numbers from 1 to n. $p_1 \le \frac{n}{2}$ must hold.

Example

input
2 1 2
output
? 1 2 ? 1 1 ! 1 2