A. Yet Another String Game

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

Homer has two friends Alice and Bob. Both of them are string fans.

One day, Alice and Bob decide to play a game on a string $s = s_1 s_2 \dots s_n$ of length n consisting of lowercase English letters. They move in turns alternatively and **Alice makes the first move**.

In a move, a player **must** choose an index i ($1 \leq i \leq n$) that has not been chosen before, and change s_i to any other lowercase English letter c that $c \neq s_i$.

When all indices have been chosen, the game ends.

The goal of Alice is to make the final string lexicographically as small as possible, while the goal of Bob is to make the final string lexicographically as large as possible. Both of them are game experts, so they always play games optimally. Homer is not a game expert, so he wonders what the final string will be.

A string a is lexicographically smaller than a string 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 string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

Input

Each test contains multiple test cases. The first line contains t $(1 \leq t \leq 1000)$ — the number of test cases. Description of the test cases follows.

The only line of each test case contains a single string s ($1 \leq |s| \leq 50$) consisting of lowercase English letters.

Output

For each test case, print the final string in a single line.

Example

input

```
3
a
bbbb
az
```

output

```
b
azaz
by
```

Note

In the first test case: Alice makes the first move and must change the only letter to a different one, so she changes it to 'b'.

In the second test case: Alice changes the first letter to 'a', then Bob changes the second letter to 'z', Alice changes the third letter to 'a' and then Bob changes the fourth letter to 'z'.

In the third test case: Alice changes the first letter to 'b', and then Bob changes the second letter to 'y'.

B. The Great Hero

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

The great hero guards the country where Homer lives. The hero has attack power A and initial health value B. There are n monsters in front of the hero. The i-th monster has attack power a_i and initial health value b_i.

The hero or a monster is said to be living, if his or its health value is positive (greater than or equal to 1); and he or it is said to be dead, if his or its health value is non-positive (less than or equal to 0).

In order to protect people in the country, the hero will fight with monsters until either the hero is dead or all the monsters are dead.

- In each fight, the hero can select an arbitrary living monster and fight with it. Suppose the i-th monster is selected, and the health values of the hero and the i-th monster are x and y before the fight, respectively. After the fight, the health values of the hero and the i-th monster become $x - a_i$ and $y - A$, respectively.

Note that the hero can fight the same monster more than once.

For the safety of the people in the country, please tell them whether the great hero can kill all the monsters (even if the great hero himself is dead after killing the last monster).

Input

Each test contains multiple test cases. The first line contains t ($1 \le t \le 10^5$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains three integers A ($1 \le A \le 10^6$), B ($1 \le B \le 10^6$) and n ($1 \le n \le 10^5$) — the attack power of the great hero, the initial health value of the great hero, and the number of monsters.

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^6$), where a_i denotes the attack power of the i-th monster.

The third line of each test case contains n integers $b_1, b_2, \dots, b_n$ ($1 \le b_i \le 10^6$), where b_i denotes the initial health value of the i-th monster.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case print the answer: "YES" (without quotes) if the great hero can kill all the monsters. Otherwise, print "NO" (without quotes).

Example

input

```
5
3 17 1
2
16
10 999 3
10 20 30
100 50 30
1000 1000 4
200 300 400 500
1000 1000 1000 1000
999 999 1
1000
1000
999 999 1
1000000
999
```

output

```
YES
YES
YES
NO
YES
```

Note

In the first example: There will be 6 fights between the hero and the only monster. After that, the monster is dead and the health value of the hero becomes $17 - 6 \times 2 = 5 > 0$. So the answer is "YES", and moreover, the hero is still living.

In the second example: After all monsters are dead, the health value of the hero will become 709, regardless of the order of all fights. So the answer is "YES".

In the third example: A possible order is to fight with the 1-st, 2-nd, 3-rd and 4-th monsters. After all fights, the health value of the hero becomes -400. Unfortunately, the hero is dead, but all monsters are also dead. So the answer is "YES".

In the fourth example: The hero becomes dead but the monster is still living with health value $1000 - 999 = 1$. So the answer is "NO".

C. Searching Local Minimum

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

This is an interactive problem.

Homer likes arrays a lot and he wants to play a game with you.

Homer has hidden from you a permutation $a_1, a_2, \dots, a_n$ of integers 1 to n. You are asked to find any index k ($1 \le k \le n$) which is a local minimum.

For an array $a_1, a_2, \dots, a_n$, an index i ($1 \le i \le n$) is said to be a *local minimum* if $a_i < \min\{a_{i-1}, a_{i+1}\}$, where $a_0 = a_{n+1} = +\infty$. An array is said to be a permutation of integers 1 to n, if it contains all integers from 1 to n exactly once.

Initially, you are only given the value of n without any other information about this permutation.

At each interactive step, you are allowed to choose any i ($1 \le i \le n$) and make a query with it. As a response, you will be given the value of a_i.

You are asked to find any index k which is a local minimum **after at most 100 queries**.

Interaction

You begin the interaction by reading an integer n ($1 \le n \le 10^5$) on a separate line.

To make a query on index i ($1 \le i \le n$), you should output "? i" in a separate line. Then read the value of a_i in a separate line. The number of the "?" queries is limited within 100.

When you find an index k ($1 \le k \le n$) which is a local minimum, output "! k" in a separate line and terminate your program.

In case your query format is invalid, or you have made more than 100 "?" queries, you will receive **Wrong Answer** verdict.

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

The first line of the hack should contain a single integer n ($1 \le n \le 10^5$).

The second line should contain n distinct integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$).

Example

input

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

output

```
? 1
? 2
? 3
? 4
? 5
! 3
```

Note

In the example, the first line contains an integer 5 indicating that the length of the array is $n = 5$.

The example makes five "?" queries, after which we conclude that the array is $a = [3, 2, 1, 4, 5]$ and $k = 3$ is local minimum.

D1. Painting the Array I

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

The only difference between the two versions is that this version asks the *maximal* possible answer.

Homer likes arrays a lot. Today he is painting an array $a_1, a_2, \dots, a_n$ with two kinds of colors, **white** and **black**. A painting assignment for $a_1, a_2, \dots, a_n$ is described by an array $b_1, b_2, \dots, b_n$ that b_i indicates the color of a_i (0 for white and 1 for black).

According to a painting assignment $b_1, b_2, \dots, b_n$, the array a is split into two new arrays $a^{(0)}$ and $a^{(1)}$, where $a^{(0)}$ is the sub-sequence of all white elements in a and $a^{(1)}$ is the sub-sequence of all black elements in a. For example, if $a = [1, 2, 3, 4, 5, 6]$ and $b = [0, 1, 0, 1, 0, 0]$, then $a^{(0)} = [1, 3, 5, 6]$ and $a^{(1)} = [2, 4]$.

The number of segments in an array $c_1, c_2, \dots, c_k$, denoted $seg(c)$, is the number of elements if we merge all adjacent elements with the same value in c. For example, the number of segments in $[1, 1, 2, 2, 3, 3, 3, 2]$ is 4, because the array will become $[1, 2, 3, 2]$ after merging adjacent elements with the same value. Especially, the number of segments in an empty array is 0.

Homer wants to find a painting assignment b, according to which the number of segments in both $a^{(0)}$ and $a^{(1)}$, i.e.

$seg(a^{(0)}) + seg(a^{(1)})$, is as **large** as possible. Find this number.

Input

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

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$).

Output

Output a single integer, indicating the **maximal** possible total number of segments.

Examples

input

```
7
1 1 2 2 3 3 3
```

output

```
6
```

input

```
7
1 2 3 4 5 6 7
```

output

```
7
```

Note

In the first example, we can choose $a^{(0)} = [1, 2, 3, 3]$, $a^{(1)} = [1, 2, 3]$ and $seg(a^{(0)}) = seg(a^{(1)}) = 3$. So the answer is $3 + 3 = 6$.

In the second example, we can choose $a^{(0)} = [1, 2, 3, 4, 5, 6, 7]$ and $a^{(1)}$ is empty. We can see that $seg(a^{(0)}) = 7$ and $seg(a^{(1)}) = 0$. So the answer is $7 + 0 = 7$.

D2. Painting the Array II

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

The only difference between the two versions is that this version asks the *minimal* possible answer.

Homer likes arrays a lot. Today he is painting an array $a_1, a_2, \dots, a_n$ with two kinds of colors, **white** and **black**. A painting assignment for $a_1, a_2, \dots, a_n$ is described by an array $b_1, b_2, \dots, b_n$ that b_i indicates the color of a_i (0 for white and 1 for black).

According to a painting assignment $b_1, b_2, \dots, b_n$, the array a is split into two new arrays $a^{(0)}$ and $a^{(1)}$, where $a^{(0)}$ is the sub-sequence of all white elements in a and $a^{(1)}$ is the sub-sequence of all black elements in a. For example, if $a = [1, 2, 3, 4, 5, 6]$ and $b = [0, 1, 0, 1, 0, 0]$, then $a^{(0)} = [1, 3, 5, 6]$ and $a^{(1)} = [2, 4]$.

The number of segments in an array $c_1, c_2, \dots, c_k$, denoted $seg(c)$, is the number of elements if we merge all adjacent elements with the same value in c. For example, the number of segments in $[1, 1, 2, 2, 3, 3, 3, 2]$ is 4, because the array will become $[1, 2, 3, 2]$ after merging adjacent elements with the same value. Especially, the number of segments in an empty array is 0.

Homer wants to find a painting assignment b, according to which the number of segments in both $a^{(0)}$ and $a^{(1)}$, i.e. $seg(a^{(0)}) + seg(a^{(1)})$, is as **small** as possible. Find this number.

Input

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

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$).

Output

Output a single integer, indicating the **minimal** possible total number of segments.

Examples

input

```
6
1 2 3 1 2 2
```

output

```
4
```

input

```
7
1 2 1 2 1 2 1
```

output

```
2
```

Note

In the first example, we can choose $a^{(0)} = [1, 1, 2, 2]$, $a^{(1)} = [2, 3]$ and $seg(a^{(0)}) = seg(a^{(1)}) = 2$. So the answer is $2 + 2 = 4$.

In the second example, we can choose $a^{(0)} = [1, 1, 1, 1]$, $a^{(1)} = [2, 2, 2]$ and $seg(a^{(0)}) = seg(a^{(1)}) = 1$. So the answer is $1 + 1 = 2$.

E. Continuous City

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

Some time ago Homer lived in a beautiful city. There were n blocks numbered from 1 to n and m directed roads between them. Each road had a positive length, and each road went from the block with the smaller index to the block with the larger index. For every two (different) blocks, there was at most one road between them.

Homer discovered that for some two numbers L and R the city was (L, R)-*continuous*.

The city is said to be (L, R)-continuous, if

1. all paths from block 1 to block n are of length between L and R (inclusive); and
2. for every $L \leq d \leq R$, there is **exactly one** path from block 1 to block n whose length is d.

A path from block u to block v is a sequence $u = x_0 \to x_1 \to x_2 \to \cdots \to x_k = v$, where there is a road from block x_{i-1} to block x_i for every $1 \leq i \leq k$. The length of a path is the sum of lengths over all roads in the path. Two paths $x_0 \to x_1 \to \cdots \to x_k$ and $y_0 \to y_1 \to \cdots \to y_l$ are different, if $k \neq l$ or $x_i \neq y_i$ for some $0 \leq i \leq \min\{k, l\}$.

After moving to another city, Homer only remembers the two special numbers L and R but forgets the numbers n and m of blocks and roads, respectively, and how blocks are connected by roads. However, he believes the number of blocks should be no larger than 32 (because the city was small).

As the best friend of Homer, please tell him whether it is possible to find a (L, R)-continuous city or not.

Input

The single line contains two integers L and R ($1 \leq L \leq R \leq 10^6$).

Output

If it is impossible to find a (L, R)-continuous city within 32 blocks, print "NO" in a single line.

Otherwise, print "YES" in the first line followed by a description of a (L, R)-continuous city.

The second line should contain two integers n ($2 \leq n \leq 32$) and m ($1 \leq m \leq \frac{n(n-1)}{2}$), where n denotes the number of blocks and m denotes the number of roads.

Then m lines follow. The i-th of the m lines should contain three integers a_i, b_i ($1 \leq a_i < b_i \leq n$) and c_i ($1 \leq c_i \leq 10^6$) indicating that there is a directed road from block a_i to block b_i of length c_i.

It is required that for every two blocks, there should be **no more than 1** road connecting them. That is, for every $1 \leq i < j \leq m$, either $a_i \neq a_j$ or $b_i \neq b_j$.

Examples

input

```
1 1
```

output

```
YES
2 1
1 2 1
```

input

```
4 6
```

output

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

Note

In the first example there is only one path from block 1 to block $n = 2$, and its length is 1.

In the second example there are three paths from block 1 to block $n = 5$, which are $1 \to 2 \to 5$ of length 4, $1 \to 3 \to 5$ of length 5 and $1 \to 4 \to 5$ of length 6.