Codeforces Round #682 (Div. 2)

A. Specific Tastes of Andre

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

Andre has very specific tastes. Recently he started falling in love with arrays.

Andre calls an nonempty array b **good**, if sum of its elements is divisible by the length of this array. For example, array $[2, 3, 1]$ is good, as sum of its elements — 6 — is divisible by 3, but array $[1, 1, 2, 3]$ isn't good, as 7 isn't divisible by 4.

Andre calls an array a of length n **perfect** if the following conditions hold:

- Every nonempty subarray of this array is **good**.
- For every i ($1 \le i \le n$), $1 \le a_i \le 100$.

Given a positive integer n, output any **perfect** array of length n. We can show that for the given constraints such an array always exists.

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

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first and only line of every test case contains a single integer n ($1 \le n \le 100$).

Output

For every test, output any **perfect** array of length n on a separate line.

Example

input

```
3
1
2
4
```

output

```
24
19 33
7 37 79 49
```

Note

Array $[19, 33]$ is perfect as all 3 its subarrays: $[19]$, $[33]$, $[19, 33]$, have sums divisible by their lengths, and therefore are good.

B. Valerii Against Everyone

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

You're given an array b of length n. Let's define another array a, also of length n, for which $a_i = 2^{b_i}$ ($1 \le i \le n$).

Valerii says that every two non-intersecting subarrays of a have different sums of elements. You want to determine if he is wrong. More formally, you need to determine if there exist four integers l_1, r_1, l_2, r_2 that satisfy the following conditions:

- $1 \le l_1 \le r_1 < l_2 \le r_2 \le n$;
- $a_{l_1} + a_{l_1+1} + \ldots + a_{r_1-1} + a_{r_1} = a_{l_2} + a_{l_2+1} + \ldots + a_{r_2-1} + a_{r_2}$.

If such four integers exist, you will prove Valerii wrong. Do they exist?

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

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). Description of the test cases follows.

The first line of every test case contains a single integer n ($2 \le n \le 1000$).

The second line of every test case contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^9$).

Output

For every test case, if there exist two non-intersecting subarrays in a that have the same sum, output YES on a separate line. Otherwise, output NO on a separate line.

Also, note that each letter can be in any case.

Example

input

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

output

```
YES
NO
```

Note

In the first case, $a = [16, 8, 1, 2, 4, 1]$. Choosing $l_1 = 1$, $r_1 = 1$, $l_2 = 2$ and $r_2 = 6$ works because $16 = (8 + 1 + 2 + 4 + 1)$.

In the second case, you can verify that there is no way to select to such subarrays.

C. Engineer Artem

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

Artem is building a new robot. He has a matrix a consisting of n rows and m columns. The cell located on the i-th row from the top and the j-th column from the left has a value $a_{i,j}$ written in it.

If two adjacent cells contain the same value, the robot will break. A matrix is called **good** if no two adjacent cells contain the same value, where two cells are called adjacent if they share a side.

Artem wants to **increment the values in some cells by one** to make a good.

More formally, find a good matrix b that satisfies the following condition —

- For all valid (i, j), either $b_{i,j} = a_{i,j}$ or $b_{i,j} = a_{i,j} + 1$.

For the constraints of this problem, it can be shown that such a matrix b always exists. If there are several such tables, you can output any of them. Please note that you do not have to minimize the number of increments.

Input

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

The first line of each test case contains two integers n, m ($1 \le n \le 100$, $1 \le m \le 100$) — the number of rows and columns, respectively.

The following n lines each contain m integers. The j-th integer in the i-th line is $a_{i,j}$ ($1 \le a_{i,j} \le 10^9$).

Output

For each case, output n lines each containing m integers. The j-th integer in the i-th line is $b_{i,j}$.

Example

input

```
3
3 2
1 2
4 5
7 8
2 2
1 1
3 3
2 2
1 3
2 2
```

output

```
1 2
5 6
7 8
2 1
```

```
4 3
2 4
3 2
```

Note

In all the cases, you can verify that no two adjacent cells have the same value and that b is the same as a with some values incremented by one.

D. Powerful Ksenia

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

Ksenia has an array a consisting of n positive integers $a_1, a_2, \ldots, a_n$.

In one operation she can do the following:

- choose three distinct indices i, j, k, and then
- change all of a_i, a_j, a_k to $a_i \oplus a_j \oplus a_k$ simultaneously, where $\oplus$ denotes the bitwise XOR operation.

She wants to make all a_i equal **in at most n operations**, or to determine that it is impossible to do so. She wouldn't ask for your help, but please, help her!

Input

The first line contains one integer n ($3 \le n \le 10^5$) — the length of a.

The second line contains n integers, $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — elements of a.

Output

Print YES or NO in the first line depending on whether it is possible to make all elements equal in at most n operations.

If it is possible, print an integer m ($0 \le m \le n$), which denotes the number of operations you do.

In each of the next m lines, print three distinct integers i, j, k, representing one operation.

If there are many such operation sequences possible, print any. Note that you do **not** have to minimize the number of operations.

Examples

input

```
5
4 2 1 7 2
```

output

```
YES
1
1 3 4
```

input

```
4
10 4 49 22
```

output

```
NO
```

Note

In the first example, the array becomes $[4 \oplus 1 \oplus 7, 2, 4 \oplus 1 \oplus 7, 4 \oplus 1 \oplus 7, 2] = [2, 2, 2, 2, 2]$.

E. Yurii Can Do Everything

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

Yurii is sure he can do everything. Can he solve this task, though?

He has an array a consisting of n positive integers. Let's call a subarray $a[l \ldots r]$ **good** if the following conditions are simultaneously satisfied:

- $l + 1 \le r - 1$, i. e. the subarray has length at least 3;
- $(a_l \oplus a_r) = (a_{l+1} + a_{l+2} + \ldots + a_{r-2} + a_{r-1})$, where $\oplus$ denotes the bitwise XOR operation.

In other words, a subarray is good if the bitwise XOR of the two border elements is equal to the sum of the rest of the elements.

Yurii wants to calculate the total number of good subarrays. What is it equal to?

An array c is a subarray of an array d if c can be obtained from d 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 a single integer n ($3 \leq n \leq 2 \cdot 10^5$) — the length of a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i < 2^{30}$) — elements of a.

Output

Output a single integer — the number of good subarrays.

Examples

input

```
8
3 1 2 3 1 2 3 15
```

output

```
6
```

input

```
10
997230370 58052053 240970544 715275815 250707702 156801523 44100666 64791577 43523002 480196854
```

output

```
2
```

Note

There are 6 good subarrays in the example:

- $[3, 1, 2]$ (twice) because $(3 \oplus 2) = 1$;
- $[1, 2, 3]$ (twice) because $(1 \oplus 3) = 2$;
- $[2, 3, 1]$ because $(2 \oplus 1) = 3$;
- $[3, 1, 2, 3, 1, 2, 3, 15]$ because $(3 \oplus 15) = (1 + 2 + 3 + 1 + 2 + 3)$.

F. Olha and Igor

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

This is an interactive problem.

Igor wants to find the key to Olha's heart. The problem is, that it's at the root of a binary tree.

There is a perfect binary tree of height h consisting of $n = 2^h - 1$ nodes. The nodes have been assigned distinct labels from 1 to n. However, **Igor only knows h and does not know which label corresponds to which node**.

To find key to Olha's heart he needs to find the label assigned to the root by making queries of the following type **at most $n + 420$ times**:

- Select three **distinct** labels u, v and w ($1 \leq u, v, w \leq n$).
- In response, Olha (the grader) will tell him the label of the **lowest common ancestor** of nodes labelled u and v, if the tree was **rooted** at the node labelled w instead.

Help Igor to find the root!

Note: the grader is not adaptive: the labels are fixed before any queries are made.

Input

The first and only line contains a single integer h ($3 \leq h \leq 18$) — the height of the tree.

Interaction

You begin the interaction by reading h.

To make a query for labels u, v, w, in a separate line output "`? u v w`".

Numbers in the query have to satisfy $1 \leq u, v, w \leq n$. Additionally, $u \neq v$, $u \neq w$, and $v \neq w$.

In response, you will receive $1 \leq x \leq n$, the label of the lowest common ancestor of u and v, if the tree was rooted at w.

In case your query is invalid or you asked more than $n + 420$ queries, program will print -1 and will finish interaction. You will receive **Wrong answer** verdict. Make sure to exit immediately to avoid getting other verdicts.

When you determine the label assigned to the root, output "! r", where r is the label of the root.

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

To hack, use the following format.

The first line should contain a single integer h (height of the binary tree).

On the next line, output a permutation p of size $n = 2^h - 1$. This represents a binary tree where the root is labelled p_1 and for $1 < i \le n$, the parent of p_i is $p_{\lfloor \frac{i}{2} \rfloor}$.

Example

input

```
3

2

7

4
```

output

```
? 7 3 5

? 1 6 4

? 1 5 4

! 4
```

Note

The labels corresponding to the tree in the example are [4,7,2,6,1,5,3], meaning the root is labelled 4, and for $1 < i \le n$, the parent of p_i is $p_{\lfloor \frac{i}{2} \rfloor}$.