A. Nastia and Nearly Good Numbers

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

Nastia has 2 positive integers A and B. She defines that:

- The integer is good if it is divisible by $A \cdot B$;
- Otherwise, the integer is nearly good, if it is divisible by A.

For example, if $A = 6$ and $B = 4$, the integers 24 and 72 are good, the integers 6, 660 and 12 are nearly good, the integers 16, 7 are neither good nor nearly good.

Find 3 **different** positive integers x, y, and z such that **exactly one** of them is good and the **other** 2 are nearly good, and $x + y = z$.

Input

The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases.

The first line of each test case contains two integers A and B ($1 \le A \le 10^6$, $1 \le B \le 10^6$) — numbers that Nastia has.

Output

For each test case print:

- "YES" and 3 **different** positive integers x, y, and z ($1 \le x, y, z \le 10^{18}$) such that **exactly one** of them is good and the **other** 2 are nearly good, and $x + y = z$.
- "NO" if no answer exists.

You can print each character of "YES" or "NO" in any case.
If there are multiple answers, print any.

Example

input

```
3
5 3
13 2
7 11
```

output

```
YES
10 50 60
YES
169 39 208
YES
28 154 182
```

Note

In the first test case: 60 — good number; 10 and 50 — nearly good numbers.

In the second test case: 208 — good number; 169 and 39 — nearly good numbers.

In the third test case: 154 — good number; 28 and 182 — nearly good numbers.

B. Nastia and a Good Array

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

Nastia has received an array of n positive integers as a gift.

She calls such an array a good that for all i ($2 \le i \le n$) takes place $gcd(a_{i-1}, a_i) = 1$, where $gcd(u, v)$ denotes the greatest common divisor (GCD) of integers u and v.

You can perform the operation: select two **different** indices i, j ($1 \le i, j \le n$, $i \ne j$) and two integers x, y ($1 \le x, y \le 2 \cdot 10^9$) so that $\min{(a_i, a_j)} = \min{(x, y)}$. Then change a_i to x and a_j to y.

The girl asks you to make the array good using **at most** n operations.

It can be proven that this is always possible.

Input

The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the array which Nastia has received as a gift.

It's guaranteed that the sum of n in one test doesn't exceed $2 \cdot 10^5$.

Output

For each of t test cases print a single integer k ($0 \le k \le n$) — the number of operations. You don't need to minimize this number.

In each of the next k lines print 4 integers i, j, x, y ($1 \le i \ne j \le n$, $1 \le x, y \le 2 \cdot 10^9$) so that $\min{(a_i, a_j)} = \min{(x, y)}$ — in this manner you replace a_i with x and a_j with y.

If there are multiple answers, print any.

Example

input

```
2
5
9 6 3 11 15
3
7 5 13
```

output

```
2
1 5 11 9
2 5 7 6
0
```

Note

Consider the first test case.

Initially $a = [9, 6, 3, 11, 15]$.

In the first operation replace a_1 with 11 and a_5 with 9. It's valid, because $\min{(a_1, a_5)} = \min{(11, 9)} = 9$.

After this $a = [11, 6, 3, 11, 9]$.

In the second operation replace a_2 with 7 and a_5 with 6. It's valid, because $\min{(a_2, a_5)} = \min{(7, 6)} = 6$.

After this $a = [11, 7, 3, 11, 6]$ — a good array.

In the second test case, the initial array is already good.

C. Nastia and a Hidden Permutation

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

This is an interactive problem!

Nastia has a hidden permutation p of length n consisting of integers from 1 to n. You, for some reason, want to figure out the permutation. To do that, you can give her an integer t ($1 \le t \le 2$), two **different** indices i and j ($1 \le i, j \le n$, $i \neq j$), and an integer x ($1 \le x \le n - 1$).

Depending on t, she will answer:

- $t = 1$: $\max{(\min{(x, p_i)}, \min{(x + 1, p_j)})}$;
- $t = 2$: $\min{(\max{(x, p_i)}, \max{(x + 1, p_j)})}$.

You can ask Nastia **at most** $\lfloor \frac{3 \cdot n}{2} \rfloor + 30$ times. It is guaranteed that she will **not** change her permutation depending on your queries. Can you guess the permutation?

Input

The input consists of several test cases. In the beginning, you receive the integer T ($1 \le T \le 10\,000$) — the number of test cases.

At the beginning of each test case, you receive an integer n ($3 \le n \le 10^4$) — the length of the permutation p.

It's guaranteed that the permutation is fixed beforehand and that the sum of n in one test doesn't exceed $2 \cdot 10^4$.

Interaction

To ask a question, print "? t i j x" ($t = 1$ or $t = 2$, $1 \le i, j \le n$, $i \ne j$, $1 \le x \le n - 1$) Then, you should read the answer.

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 the 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 $\lfloor \frac{3 \cdot n}{2} \rfloor + 30$ 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

To hack the solution, use the following test format.

The first line should contain a single integer T ($1 \le T \le 10\,000$) — the number of test cases.

For each test case in the first line print a single integer n ($3 \le n \le 10^4$) — the length of the hidden permutation p.

In the second line print n space-separated integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$), where p is permutation.

Note that the sum of n over all test cases should not exceed $2 \cdot 10^4$.

Example

input

```
2
4

3

2

5

3
```

output

```
? 2 4 1 3

? 1 2 4 2

! 3 1 4 2

? 2 3 4 2

! 2 5 3 4 1
```

Note

Consider the first test case.

The hidden permutation is $[3, 1, 4, 2]$.

We print: "? 2 4 1 3" and get back $\min(\max(3, p_4), \max(4, p_1)) = 3$.

We print: "? 1 2 4 2" and get back $\max(\min(2, p_2), \min(3, p_4)) = 2$.

Consider the second test case.

The hidden permutation is $[2, 5, 3, 4, 1]$.

We print: "? 2 3 4 2" and get back $\min(\max(2, p_3), \max(3, p_4)) = 3$.

D. Nastia Plays with a Tree

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

Nastia has an unweighted tree with n vertices and wants to play with it!

The girl will perform the following operation with her tree, as long as she needs:

1. Remove any existing edge.
2. Add an edge between any pair of vertices.

What is the **minimum** number of operations Nastia needs to get a bamboo from a tree? A bamboo is a tree in which no node has a degree greater than 2.

Input

The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases.

The first line of each test case contains a single integer n ($2 \le n \le 10^5$) — the number of vertices in the tree.

Next $n-1$ lines of each test cases describe the edges of the tree in form a_i, b_i ($1 \le a_i, b_i \le n$, $a_i \neq b_i$).

It's guaranteed the given graph is a tree and the sum of n in one test doesn't exceed $2 \cdot 10^5$.

Output

For each test case in the first line print a single integer k — the minimum number of operations required to obtain a bamboo from the initial tree.

In the next k lines print 4 integers x_1, y_1, x_2, y_2 ($1 \le x_1, y_1, x_2, y_2 \le n$, $x_1 \neq y_1$, $x_2 \neq y_2$) — this way you remove the edge (x_1, y_1) and add an undirected edge (x_2, y_2).

Note that the edge (x_1, y_1) must be present in the graph at the moment of removing.

Example

input

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

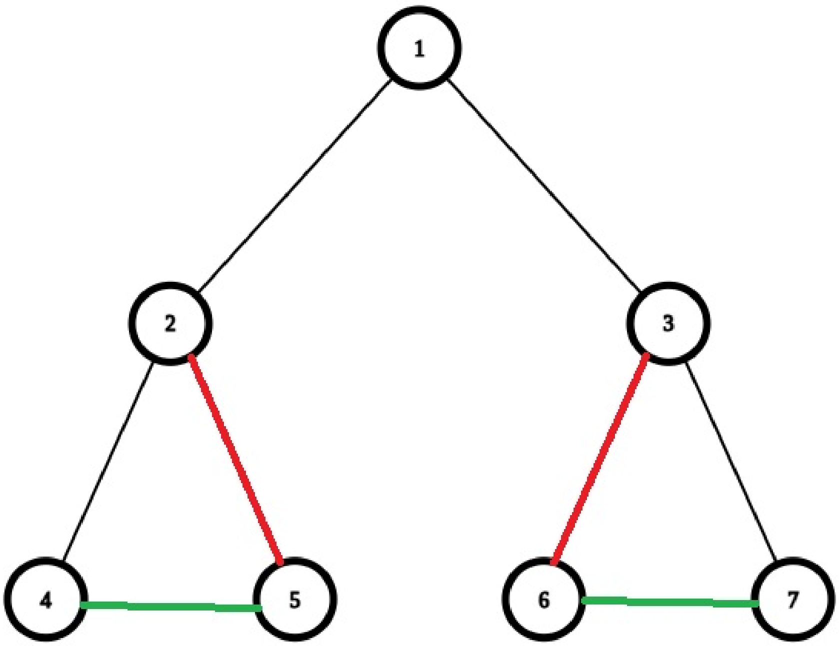
output

```
2
2 5 6 7
3 6 4 5
0
```

Note

Note the graph can be **unconnected** after a certain operation.

Consider the first test case of the example:



The red edges are removed, and the green ones are added.

E. Nastia and a Beautiful Matrix

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

You like numbers, don't you? Nastia has a lot of numbers and she wants to share them with you! Isn't it amazing?

Let a_i be how many numbers i ($1 \le i \le k$) you have.

An $n \times n$ matrix is called beautiful if it contains **all** the numbers you have, and for **each** 2×2 submatrix of the original matrix is satisfied:

1. The number of occupied cells doesn't exceed 3;
2. The numbers on each diagonal are distinct.

Make a beautiful matrix of **minimum** size.

Input

The first line contains a single integer t ($1 \le t \le 10\,000$) — the number of test cases.

The first line of each test case contains 2 integers m and k ($1 \le m, k \le 10^5$) — how many numbers Nastia gave you and the length of the array a, respectively.

The second line of each test case contains k integers $a_1, a_2, \ldots, a_k$ ($0 \le a_i \le m$, $a_1 + a_2 + \ldots + a_k = m$), where a_i is how many numbers i you have.

It's guaranteed that the sum of m and k in one test doesn't exceed $2 \cdot 10^5$.

Output

For each t test case print a single integer n — the size of the beautiful matrix.

In the next n lines print n integers $b_{i,j}$ ($0 \le b_{i,j} \le k$; if position is empty, print $b_{i,j} = 0$) — the beautiful matrix b you made up.

Example

input

```
2
3 4
2 0 0 1
15 4
2 4 8 1
```

output

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

Note

Note that 0 in this problem represents a blank, not a number.

Examples of possible answers for the first test case:

$$\begin{matrix} 1 & 1 \\ 4 & 0 \end{matrix} \quad \begin{matrix} 1 & 4 \\ 1 & 0 \end{matrix} \quad \begin{matrix} 4 & 0 \\ 1 & 1 \end{matrix}$$

Examples of **not** beautiful matrices for the first test case:

$$\begin{matrix} 1 & 0 \\ 4 & 1 \end{matrix} \quad \begin{matrix} 4 & 1 \\ 7 & 1 \end{matrix} \quad \begin{matrix} 1 & 0 \\ 4 & 0 \end{matrix}$$

The example of the **not** beautiful matrix for the second test case:

$$\begin{matrix} 3 & 4 & 0 & 2 & 2 \\ 3 & 2 & 3 & 3 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \\ 2 & 1 & 3 & 3 & 3 \end{matrix}$$

Everything is okay, except the left-top submatrix contains 4 numbers.