Codeforces Round #781 (Div. 2)

A. GCD vs LCM

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

You are given a positive integer n. You have to find 4 **positive** integers a, b, c, d such that

- $a + b + c + d = n$, and
- $\gcd(a, b) = \operatorname{lcm}(c, d)$.

If there are several possible answers you can output any of them. It is possible to show that the answer always exists.

In this problem $\gcd(a, b)$ denotes the greatest common divisor of a and b, and $\operatorname{lcm}(c, d)$ denotes the least common multiple of c and d.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Description of the test cases follows.

Each test case contains a single line with integer n ($4 \le n \le 10^9$) — the sum of a, b, c, and d.

Output

For each test case output 4 **positive** integers a, b, c, d such that $a + b + c + d = n$ and $\gcd(a, b) = \operatorname{lcm}(c, d)$.

Example

input
5 4 7 8 9 10

output
1 1 1 1 2 2 2 1 2 2 2 2 2 4 2 1 3 5 1 1

Note

In the first test case $\gcd(1, 1) = \operatorname{lcm}(1, 1) = 1$, $1 + 1 + 1 + 1 = 4$.

In the second test case $\gcd(2, 2) = \operatorname{lcm}(2, 1) = 2$, $2 + 2 + 2 + 1 = 7$.

In the third test case $\gcd(2, 2) = \operatorname{lcm}(2, 2) = 2$, $2 + 2 + 2 + 2 = 8$.

In the fourth test case $\gcd(2, 4) = \operatorname{lcm}(2, 1) = 2$, $2 + 4 + 2 + 1 = 9$.

In the fifth test case $\gcd(3, 5) = \operatorname{lcm}(1, 1) = 1$, $3 + 5 + 1 + 1 = 10$.

B. Array Cloning Technique

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

You are given an array a of n integers. Initially there is only one copy of the given array.

You can do operations of two types:

1. Choose any array and clone it. After that there is one more copy of the chosen array.
2. Swap two elements from **any** two copies (maybe in the same copy) on any positions.

You need to find the minimal number of operations needed to obtain a copy where all elements are equal.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Description of the test cases follows.

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the elements of the array a.

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

Output

For each test case output a single integer — the minimal number of operations needed to create at least one copy where all elements are equal.

Example

input
6 1 1789 6 0 1 3 3 7 0 2 -1000000000 1000000000 4 4 3 2 1 5 2 5 7 6 3 7 1 1 1 1 1 1 1

output
0 6 2 5 7 0

Note

In the first test case all elements in the array are already equal, that's why the answer is 0.

In the second test case it is possible to create a copy of the given array. After that there will be two identical arrays:

$[\ 0\ 1\ 3\ 3\ 7\ 0\ ]$ and $[\ 0\ 1\ 3\ 3\ 7\ 0\ ]$

After that we can swap elements in a way so all zeroes are in one array:

$[\ 0\ \underline{0}\ \underline{0}\ 3\ 7\ 0\ ]$ and $[\ \underline{1}\ 1\ 3\ 3\ 7\ \underline{3}\ ]$

Now let's create a copy of the first array:

$[\,0\,0\,0\,3\,7\,0\,]$, $[\,0\,0\,0\,3\,7\,0\,]$ and $[\,1\,1\,3\,3\,7\,3\,]$

Let's swap elements in the first two copies:

$[\,0\,0\,0\,\underline{0}\,\underline{0}\,0\,]$, $[\,\underline{3}\,\underline{7}\,0\,3\,7\,0\,]$ and $[\,1\,1\,3\,3\,7\,3\,]$.

Finally, we made a copy where all elements are equal and made 6 operations.

It can be proven that no fewer operations are enough.

C. Tree Infection

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

A tree is a connected graph without cycles. A rooted tree has a special vertex called the root. The parent of a vertex v (different from root) is the previous to v vertex on the shortest path from the root to the vertex v. Children of the vertex v are all vertices for which v is the parent.

You are given a rooted tree with n vertices. The vertex 1 is the root. Initially, all vertices are healthy.

Each second you do **two** operations, the *spreading* operation and, after that, the *injection* operation:

1. Spreading: for **each** vertex v, if at least one child of v is infected, you can spread the disease by infecting at most one other child of v of your choice.
2. Injection: you can choose any healthy vertex and infect it.

This process repeats each second until the whole tree is infected. You need to find the minimal number of seconds needed to infect the whole tree.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Description of the test cases follows.

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

The second line of each test case contains $n-1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i \le n$), where p_i is the ancestor of the i-th vertex in the tree.

It is guaranteed that the given graph is a tree.

It is guaranteed that the sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case you should output a single integer — the minimal number of seconds needed to infect the whole tree.

Example

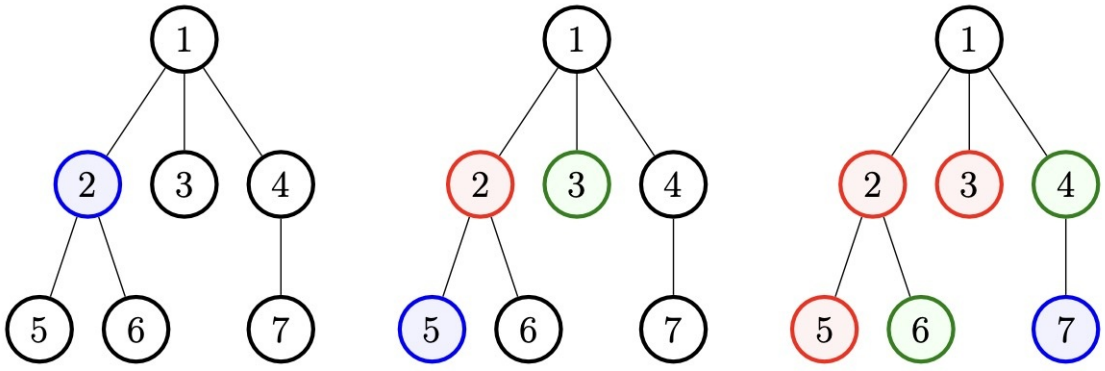
input
5 7 1 1 1 2 2 4 5 5 5 1 4 2 1 3 3 1 6 1 1 1 1 1

output
4 4 2 3 4

Note

The image depicts the tree from the first test case during each second.

A vertex is black if it is not infected. A vertex is blue if it is infected by *injection* during the previous second. A vertex is green if it is infected by *spreading* during the previous second. A vertex is red if it is infected earlier than the previous second.



Note that you are able to choose which vertices are infected by *spreading* and by *injections*.

D. GCD Guess

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

This is an interactive problem.

There is a positive integer $1 \le x \le 10^9$ that you have to guess.

In one query you can choose two positive integers $a \neq b$. As an answer to this query you will get $\gcd(x+a, x+b)$, where $\gcd(n, m)$ is the greatest common divisor of the numbers n and m.

To guess one hidden number x you are allowed to make no more than 30 queries.

Input

The first line of input contains a single integer t ($1 \le t \le 1000$) denoting the number of test cases.

The integer x that you have to guess satisfies the constraints: ($1 \le x \le 10^9$).

Interaction

The hidden number x is fixed before the start of the interaction and does not depend on your queries.

To guess each x you can make no more than 30 queries in the following way:

- "? a b" ($1 \le a, b \le 2 \cdot 10^9$, $a \neq b$).

For this query you will get $\gcd(x + a, x + b)$.

When you know x, print a single line in the following format.

- "! x" ($1 \le x \le 10^9$).

After that continue to solve the next test case.

If you ask more than 30 queries for one x or make an invalid query, the interactor will terminate immediately and your program will receive verdict `Wrong Answer`.

After printing each query do not forget to output end of line and flush the output buffer. Otherwise, you will get the `Idleness limit exceeded` verdict. To do flush use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- Read documentation for other languages.

Hacks

To use hacks, use the following format of tests:

The first line should contain a single integer t ($1 \le t \le 1000$) — the number of test cases.

The first and only line of each test case should contain a single integer x ($1 \le x \le 10^9$) denoting the integer x that should be guessed.

Example

input
2 1 8   1

output
? 1 2 ? 12 4 ! 4 ? 2000000000 1999999999 ! 1000000000

Note

The first hidden number is 4, that's why the answers for the queries are:

"? 1 2" — $\gcd(4 + 1, 4 + 2) = \gcd(5, 6) = 1$.

"? 12 4" — $\gcd(4 + 12, 4 + 4) = \gcd(16, 8) = 8$.

The second hidden number is 10^9, that's why the answer for the query is:

"? 2000000000 1999999999" — $\gcd(3 \cdot 10^9, 3 \cdot 10^9 - 1) = 1$.

These queries are made only for understanding the interaction and are not enough for finding the true x.

E. MinimizOR

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

You are given an array a of n non-negative integers, numbered from 1 to n.

Let's define the *cost* of the array a as $\min\limits_{i \neq j} a_i | a_j$, where $|$ denotes the bitwise OR operation.

There are q queries. For each query you are given two integers l and r ($l < r$). For each query you should find the cost of the subarray $a_l, a_{l+1}, \ldots, a_r$.

Input

Each test case consists of several test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains an integer n ($2 \le n \le 10^5$) — the length array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^{30}$) — the elements of a.

The third line of each test case contains an integer q ($1 \le q \le 10^5$) — the number of queries.

Each of the next q lines contains two integers l_j, r_j ($1 \le l_j < r_j \le n$) — the description of the j-th query.

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

Output

For each test case print q numbers, where the j-th number is the cost of array $a_{l_j}, a_{l_j+1}, \ldots, a_{r_j}$.

Example

input
2 5 6 1 3 2 1 4 1 2 2 3 2 4 2 5 4 0 2 1 1073741823 4 1 2 2 3 1 3 3 4

output
7 3 3 1 2 3 1 1073741823

Note

In the first test case the array a is

$110_2, 001_2, 011_2, 010_2, 001_2$.

That's why the answers for the queries are:

- $[1; 2]$: $a_1 | a_2 = 110_2 | 001_2 = 111_2 = 7$;
- $[2; 3]$: $a_2 | a_3 = 001_2 | 011_2 = 011_2 = 3$;
- $[2; 4]$: $a_2 | a_3 = a_3 | a_4 = a_2 | a_4 = 011_2 = 3$;
- $[2; 5]$: $a_2 | a_5 = 001_2 = 1$.

In the second test case the array a is

$00_2, 10_2, 01_2, \underbrace{11\ldots1_2}_{30}$ ($a_4 = 2^{30} - 1$).

That's why the answers for the queries are:

- $[1; 2]$: $a_1 | a_2 = 10_2 = 2$;
- $[2; 3]$: $a_2 | a_3 = 11_2 = 3$;
- $[1; 3]$: $a_1 | a_3 = 01_2 = 1$;
- $[3; 4]$: $a_3 | a_4 = 01_2 | \underbrace{11\ldots1_2}_{30} = 2^{30} - 1 = 1073741823$.