Codeforces Round 987 (Div. 2)

A. Penchick and Modern Monument

1 second, 256 megabytes

Amidst skyscrapers in the bustling metropolis of Metro Manila, the newest Noiph mall in the Philippines has just been completed! The construction manager, Penchick, ordered a state-of-the-art monument to be built with n pillars.

The heights of the monument's pillars can be represented as an array h of n positive integers, where h_i represents the height of the i-th pillar for all i between 1 and n.

Penchick wants the heights of the pillars to be in **non-decreasing** order, i.e. $h_i \leq h_{i+1}$ for all i between 1 and $n-1$. However, due to confusion, the monument was built such that the heights of the pillars are in **non-increasing** order instead, i.e. $h_i \geq h_{i+1}$ for all i between 1 and $n-1$.

Luckily, Penchick can modify the monument and do the following operation on the pillars as many times as necessary:

- Modify the height of a pillar to any positive integer. Formally, choose an index $1 \leq i \leq n$ and a positive integer x. Then, assign $h_i := x$.

Help Penchick determine the minimum number of operations needed to make the heights of the monument's pillars **non-decreasing**.

Input

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

The first line of each test case contains a single integer n ($1 \leq n \leq 50$) — the number of pillars.

The second line of each test case contains n integers $h_1, h_2, \ldots, h_n$ ($1 \leq h_i \leq n$ and $h_i \geq h_{i+1}$) — the height of the pillars.

Please take note that the given array h is non-increasing.

Note that there are **no** constraints on the sum of n over all test cases.

Output

For each test case, output a single integer representing the minimum number of operations needed to make the heights of the pillars **non-decreasing**.

input
3 5 5 4 3 2 1 3 2 2 1 1 1

output
4 1 0

In the first test case, the initial heights of pillars are $h = [5, 4, 3, 2, 1]$.

- In the first operation, Penchick changes the height of pillar 1 to $h_1 := 2$.
- In the second operation, he changes the height of pillar 2 to $h_2 := 2$.
- In the third operation, he changes the height of pillar 4 to $h_4 := 4$.
- In the fourth operation, he changes the height of pillar 5 to $h_5 := 4$.

After the operation, the heights of the pillars are $h = [2, 2, 3, 4, 4]$, which is non-decreasing. It can be proven that it is not possible for Penchick to make the heights of the pillars non-decreasing in fewer than 4 operations.

In the second test case, Penchick can make the heights of the pillars non-decreasing by modifying the height of pillar 3 to $h_3 := 2$.

In the third test case, the heights of pillars are already non-decreasing, so no operations are required.

B. Penchick and Satay Sticks

1.5 seconds, 256 megabytes

Penchick and his friend Kohane are touring Indonesia, and their next stop is in Surabaya!

In the bustling food stalls of Surabaya, Kohane bought n satay sticks and arranged them in a line, with the i-th satay stick having length p_i. It is given that p is a permutation* of length n.

Penchick wants to sort the satay sticks in increasing order of length, so that $p_i = i$ for each $1 \leq i \leq n$. For fun, they created a rule: they can only swap neighboring satay sticks whose lengths differ by exactly 1. Formally, they can perform the following operation any number of times (including zero):

- Select an index i ($1 \leq i \leq n-1$) such that $|p_{i+1} - p_i| = 1$;
- Swap p_i and p_{i+1}.

Determine whether it is possible to sort the permutation p, thus the satay sticks, by performing the above operation.

*A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 2 \cdot 10^5$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of satay sticks.

The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ ($1 \leq p_i \leq n$) — the permutation p representing the length of the satay sticks.

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

Output

For each test case, output "YES" if it is possible to sort permutation p by performing the operation. Otherwise, output "NO".

You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

input
2 4 2 1 3 4 4 4 2 3 1

output
YES NO

In the first test case, we can sort permutation $p = [2, 1, 3, 4]$ by performing an operation on index 1 ($|p_2 - p_1| = |1 - 2| = 1$), resulting in $p = [1, 2, 3, 4]$.

In the second test case, it can be proven that it is impossible to sort permutation $p = [4, 2, 3, 1]$ by performing the operation. Here is an example of a sequence of operations that can be performed on the permutation:

- Select $i = 2$ ($|p_3 - p_2| = |3 - 2| = 1$). This results in $p = [4, 3, 2, 1]$.
- Select $i = 1$ ($|p_2 - p_1| = |3 - 4| = 1$). This results in $p = [3, 4, 2, 1]$.
- Select $i = 3$ ($|p_4 - p_3| = |1 - 2| = 1$). This results in $p = [3, 4, 1, 2]$.

Unfortunately, permutation p remains unsorted after performing the operations.

C. Penchick and BBQ Buns

2 seconds, 256 megabytes

Penchick loves two things: square numbers and Hong Kong-style BBQ buns! For his birthday, Kohane wants to combine them with a gift: n BBQ buns arranged from left to right. There are 10^6 available fillings of BBQ buns, numbered from 1 to 10^6. To ensure that Penchick would love this gift, Kohane has a few goals:

- No filling is used exactly once; that is, each filling must either not appear at all or appear at least twice.
- For any two buns i and j that have the same filling, the distance between them, which is $|i - j|$, must be a perfect square*.

Help Kohane find a valid way to choose the filling of the buns, or determine if it is impossible to satisfy her goals! If there are multiple solutions, print any of them.

*A positive integer x is a perfect square if there exists a positive integer y such that $x = y^2$. For example, 49 and 1 are perfect squares because $49 = 7^2$ and $1 = 1^2$ respectively. On the other hand, 5 is not a perfect square as no integer squared equals 5

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 2 \cdot 10^5$). The description of the test cases follows.

The only line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of BBQ buns.

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

Output

For each test case, if no valid choice of fillings exists, output -1. Otherwise, output n integers, where the i-th integer represents the filling of the i-th BBQ bun. If there are multiple solutions, print any of them.

input

```
2
3
12
```

output

```
-1
1 2 3 6 10 2 7 6 10 1 7 3
```

In the first test case, the choice of fillings "`1 1 1`" is not allowed because buns 1 and 3 have the same filling, but are distance 2 apart, which is not a perfect square. The choice of fillings "`1 1 2`" is also not allowed as filling 2 is only used once.

In the second test case, the solution is valid because no filling is used exactly once, and any two buns with the same filling are spaced at a distance equal to a perfect square. For example, buns 1 and 10 both have filling 1 and are spaced at a distance of $9 = 3^2$. Similarly, buns 5 and 9 both have filling 10 and are spaced at a distance of $4 = 2^2$.

D. Penchick and Desert Rabbit

3 seconds, 256 megabytes

Dedicated to pushing himself to his limits, Penchick challenged himself to survive the midday sun in the Arabian Desert!

While trekking along a linear oasis, Penchick spots a desert rabbit preparing to jump along a line of palm trees. There are n trees, each with a height denoted by a_i.

The rabbit can jump from the i-th tree to the j-th tree if exactly one of the following conditions is true:

- $j < i$ and $a_j > a_i$: the rabbit can jump backward to a taller tree.
- $j > i$ and $a_j < a_i$: the rabbit can jump forward to a shorter tree.

For each i from 1 to n, determine the maximum height among all trees that the rabbit can reach if it starts from the i-th tree.

Input

The first line contains the number of test cases t ($1 \le t \le 5 \cdot 10^5$). The description of the test cases follows.

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

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the height of the trees.

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

Output

For each test case, output n integers. The i-th integer should contain the maximum height among all trees that the rabbit can reach if it starts from the i-th tree.

input

```
5
4
2 3 1 4
5
5 4 3 2 1
4
2 1 1 3
4
1 1 3 1
8
2 4 1 6 3 8 5 7
```

output

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

In the first test case, the initial heights of trees are $a = [2, 3, 1, 4]$.

- If the rabbit starts from the first tree, it can jump to the third tree as $3 > 1$ and $1 < 2$. Then, the rabbit can jump to the second tree as $2 < 3$ and $3 > 1$. It can be proved that the rabbit cannot reach the fourth tree; hence, the maximum height of the tree that the rabbit can reach is $a_2 = 3$.
- If the rabbit starts from the fourth tree, it does not need to jump anywhere as it is already at the highest tree.

In the second test case, the rabbit can jump to the first tree regardless of which tree it starts from.

In the fifth test case, if the rabbit starts from the fifth tree, it can jump to the fourth tree. Then the rabbit can jump to the seventh tree and finally reach the sixth tree. Therefore, the maximum height of the tree that the rabbit can reach is 8.

E. Penchick and Chloe's Trees

3.5 seconds, 512 megabytes

With just a few hours left until Penchick and Chloe leave for Singapore, they could hardly wait to see the towering trees at the Singapore Botanic Gardens! Attempting to contain their excitement, Penchick crafted a rooted tree to keep Chloe and himself busy.

Penchick has a rooted tree* consisting of n vertices, numbered from 1 to n, with vertex 1 as the root, and Chloe can select a non-negative integer d to create a perfect binary tree† of depth d.

Since Penchick and Chloe are good friends, Chloe wants her tree to be isomorphic‡ to Penchick's tree. To meet this condition, Chloe can perform the following operation on her own tree any number of times:

- Select an edge (u, v), where u is the parent of v.
- Remove vertex v and all the edges connected to v, then connect all of v's previous children directly to u.

In particular, doing an operation on an edge (u, v) where v is a leaf will delete vertex v without adding any new edges.

Since constructing a perfect binary tree can be time-consuming, Chloe wants to choose the minimum d such that a perfect binary tree of depth d can be made isomorphic to Penchick's tree using the above operation. Note that she can't change the roots of the trees.

*A tree is a connected graph without cycles. A rooted tree is a tree where one vertex is special and called the root. The parent of vertex v is the first vertex on the simple path from v to the root. The root has no parent. A child of vertex v is any vertex u for which v is the parent. A leaf is any vertex without children.

†A full binary tree is rooted tree, in which each node has 0 or 2 children. A perfect binary tree is a full binary tree in which every leaf is at the same distance from the root. The depth of such a tree is the distance from the root to a leaf.

‡Two rooted trees, rooted at r_1 and r_2 respectively, are considered isomorphic if there exists a permutation p of the vertices such that an edge (u, v) exists in the first tree if and only if the edge (p_u, p_v) exists in the second tree, and $p_{r_1} = r_2$.

Input

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

The first line of each test case contains a single integer n ($2 \le n \le 10^6$) — the number of vertices in Penchick's tree.

The second line of each test case contains $n-1$ integers $p_2, p_3, \ldots, p_n$ ($1 \le p_i \le i-1$) — the parent of vertex i.

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, output a single integer on each line: the minimum depth of Chloe's perfect binary tree.

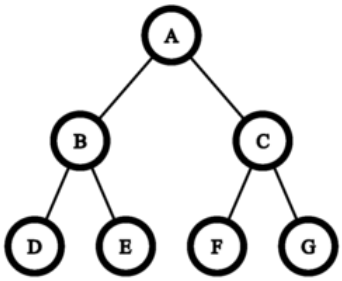
input

```
5
6
1 2 2 1 1
15
1 1 2 2 3 3 4 4 5 5 6 6 7 7
5
1 2 2 2
7
1 1 2 1 1 2
10
1 1 1 2 2 2 4 3 3
```

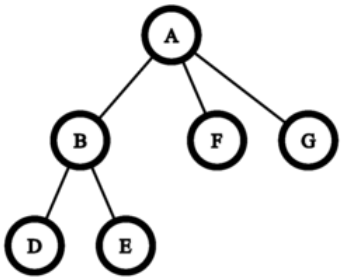
output

```
2
3
3
3
3
```

For the first test case, create a perfect binary tree with depth 2.



Consider carrying out the operation on edge AC. Then the edges AC, CF, and CG are removed, and edges AF and AG are added.



The resulting tree is isomorphic to the tree given in the input. It can be proven that no sequence of operations carried out on a binary tree of depth less than 2 can lead to a tree isomorphic to the tree given in the input.

In the second test case, the tree is already isomorphic to a perfect binary tree of depth 3.

F. Penchick and Even Medians

3 seconds, 256 megabytes

This is an interactive problem.

Returning from a restful vacation on Australia's Gold Coast, Penchick forgot to bring home gifts for his pet duck Duong Canh! But perhaps a beautiful problem crafted through deep thought on the scenic beaches could be the perfect souvenir.

There is a hidden permutation* p of length n, where n is even. You are allowed to make the following query:

- Choose a subsequence† of the permutation p with even length $4 \le k \le n$. The interactor will return the **value** of the two medians‡ in the chosen subsequence.

Find the **index** of the two medians in permutation p using at most 80 queries.

Note that the interactor is **non-adaptive**. This means that the permutation p is fixed at the beginning and will not change based on your queries.

*A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

†A sequence a is a subsequence of a sequence b if a can be obtained from b by the deletion of several (possibly, zero or all) element from arbitrary positions.

‡The two medians of an array a with even length k are defined as the $\frac{k}{2}$-th and $\left(\frac{k}{2}+1\right)$-th **smallest** element in the array (1-indexed).

Input

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

The only line of each test case contains a single integer n ($6 \le n \le 100$, n is even) — the length of the hidden permutation p.

For each test case, after reading the integer n, you should begin the interaction and find the answer before reading n for the next test case.

It is guaranteed that the sum of n over all test cases does not exceed 10^4.

Interaction

To make a query, print a single line in the following format:

- ? k x_1 x_2 ... x_{k-1} x_k ($4 \le k \le n$, k is even, $1 \le x_i \le n$, x_i is pairwise distinct) — the length of the chosen subsequence followed by the indices of the chosen subsequence.

After each query, you should read a line containing two integers m_1 and m_2 ($1 \le m_1 < m_2 \le n$) — the **value** of the two medians in array $[p_{x_1}, p_{x_2}, \ldots, p_{x_{k-1}}, p_{x_k}]$.

You can make at most 80 such queries in each test case.

To give the final answer, print a single line in the following format:

- `!` i_1 i_2 ($1 \le i_1, i_2 \le n$) — the **index** of the two medians.

Note that the order in which i_1 and i_2 is printed does not matter. In other words, your solution is valid as long as $p_{i_1} = \frac{n}{2}$ and $p_{i_2} = \frac{n}{2} + 1$, or $p_{i_1} = \frac{n}{2} + 1$ and $p_{i_2} = \frac{n}{2}$.

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

If, at any interaction step, you read -1 instead of valid data, your solution must exit immediately. This means that your solution will receive `Wrong answer` because of an invalid query or any other mistake. Failing to exit can result in an arbitrary verdict because your solution will continue to read from a closed stream.

Hack format

For hacks, use the following format.

The first line should contain t — the number of test cases.

The first line of each test case should contain a single even integer n.

The second line of each test case should contain a permutation $p_1, p_2, \ldots, p_n$ of length n.

As an example, the hack format for the example input is:

```
2
6
6 2 3 5 1 4
10
10 9 8 7 6 5 4 3 2 1
```

*To flush, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `sys.stdout.flush()` in Python;
- see the documentation for other languages.

input

```
2
6

3 4

3 4

2 3

10

3 4

6 7
```

output

```


? 6 1 2 3 4 5 6

? 4 3 6 1 5

? 4 3 6 2 5

! 3 6

? 6 1 3 7 8 9 10

? 8 1 2 3 4 5 6 7 8

! 6 5
```

In the first test case, the hidden permutation is $p = [6, 2, 3, 5, 1, 4]$.

1. The entire permutation was chosen for the first query. The two medians of the entire permutation p are 3 and 4.
2. The indices of the chosen subsequence in the second query are 3, 6, 1, and 5. The interactor returns the two medians of the subsequence $[p_3, p_6, p_1, p_5] = [3, 4, 6, 1]$, which are 3 and 4.
3. The indices of the chosen subsequence in the second query are 3, 6, 2, and 5. The interactor returns the two medians of the subsequence $[p_3, p_6, p_2, p_5] = [3, 4, 2, 1]$, which are 2 and 3.

The answer "`! 3 6`" is valid as $p_3 = 3$ and $p_6 = 4$.

In the second test case, the hidden permutation is $p = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$.

1. The indices of the chosen subsequence in the second query are 1, 3, 7, 8, 9, and 10. The interactor returns the two medians of the subsequence $[p_1, p_3, p_7, p_8, p_9, p_{10}] = [10, 8, 4, 3, 2, 1]$, which are 3 and 4.
2. The indices of the chosen subsequence in the second query are 1, 2, 3, 4, 5, 6, 7, and 8. The interactor returns the two medians of the subsequence $[p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8] = [10, 9, 8, 7, 6, 5, 4, 3]$, which are 6 and 7.

The answer "`! 5 6`" is valid as $p_5 = 6$ and $p_6 = 5$.