Codeforces Round #686 (Div. 3)

A. Special Permutation

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

You are given one integer n ($n > 1$).

Recall that 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 of length 5, 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).

Your task is to find a permutation p of length n that there is no index i ($1 \le i \le n$) such that $p_i = i$ (so, for all i from 1 to n the condition $p_i \ne i$ should be satisfied).

You have to answer t independent test cases.

If there are several answers, you can print any. It can be proven that the answer exists for each $n > 1$.

Input

The first line of the input contains one integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

The only line of the test case contains one integer n ($2 \le n \le 100$) — the length of the permutation you have to find.

Output

For each test case, print n distinct integers $p_1, p_2, \ldots, p_n$ — a permutation that there is no index i ($1 \le i \le n$) such that $p_i = i$ (so, for all i from 1 to n the condition $p_i \ne i$ should be satisfied).

If there are several answers, you can print any. It can be proven that the answer exists for each $n > 1$.

Example

input

```
2
2
5
```

output

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

B. Unique Bid Auction

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

There is a game called "Unique Bid Auction". You can read more about it here: https://en.wikipedia.org/wiki/Unique_bid_auction (though you don't have to do it to solve this problem).

Let's simplify this game a bit. Formally, there are n participants, the i-th participant chose the number a_i. The winner of the game is such a participant that the number he chose is **unique** (i. e. nobody else chose this number except him) and is **minimal** (i. e. among all unique values of a the minimum one is the winning one).

Your task is to find the **index** of the participant who won the game (or -1 if there is no winner). Indexing is 1-based, i. e. the participants are numbered from 1 to n.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of participants. The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$), where a_i is the i-th participant chosen number.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

For each test case, print the answer — the **index** of the participant who won the game (or -1 if there is no winner). **Note that the answer is always unique**.

Example

input

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

output

```
-1
2
4
1
2
-1
```

C. Sequence Transformation

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given a sequence a, initially consisting of n integers.

You want to transform this sequence so that all elements in it *are equal* (i. e. it contains several occurrences of the same element).

To achieve this, you choose some integer x **that occurs at least once in** a, and then perform the following operation any number of times (possibly zero): choose some segment $[l, r]$ of the sequence and remove it. But there is one exception: **you are not allowed to choose a segment that contains** x. More formally, you choose some contiguous subsequence $[a_l, a_{l+1}, \dots, a_r]$ such that $a_i \neq x$ if $l \leq i \leq r$, and remove it. After removal, the numbering of elements to the right of the removed segment changes: the element that was the $(r+1)$-th is now l-th, the element that was $(r+2)$-th is now $(l+1)$-th, and so on (i. e. the remaining sequence just collapses).

Note that you **can not change** x after you chose it.

For example, suppose $n = 6$, $a = [1, 3, 2, 4, 1, 2]$. Then one of the ways to transform it in two operations is to choose $x = 1$, then:

1. choose $l = 2$, $r = 4$, so the resulting sequence is $a = [1, 1, 2]$;
2. choose $l = 3$, $r = 3$, so the resulting sequence is $a = [1, 1]$.

Note that choosing x is not an operation. Also, note that you **can not** remove any occurrence of x.

Your task is to find the **minimum** number of operations required to transform the sequence in a way described above.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \leq t \leq 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of elements in a. The second line of the test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \leq a_i \leq n$), where a_i is the i-th element of a.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ ($\sum n \leq 2 \cdot 10^5$).

Output

For each test case, print the answer — the **minimum** number of operations required to transform the given sequence in a way described in the problem statement. It can be proven that it is always possible to perform a finite sequence of operations so the sequence is transformed in the required way.

Example

input

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

```
7
1 2 3 1 2 3 1
11
2 2 1 2 3 2 1 2 3 1 2
```

output

```
0
1
1
2
3
```

D. Number into Sequence

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

You are given an integer n ($n > 1$).

Your task is to find a sequence of integers $a_1, a_2, \ldots, a_k$ such that:

- each a_i is strictly greater than 1;
- $a_1 \cdot a_2 \cdot \ldots \cdot a_k = n$ (i. e. the product of this sequence is n);
- a_{i+1} is divisible by a_i for each i from 1 to $k-1$;
- k is the **maximum** possible (i. e. the length of this sequence is the **maximum** possible).

If there are several such sequences, any of them is acceptable. It can be proven that at least one valid sequence always exists for any integer $n > 1$.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 5000$) — the number of test cases. Then t test cases follow.

The only line of the test case contains one integer n ($2 \le n \le 10^{10}$).

It is guaranteed that the sum of n does not exceed 10^{10} ($\sum n \le 10^{10}$).

Output

For each test case, print the answer: in the first line, print one positive integer k — the **maximum** possible length of a. In the second line, print k integers $a_1, a_2, \ldots, a_k$ — the sequence of length k satisfying the conditions from the problem statement.

If there are several answers, you can print any. It can be proven that at least one valid sequence always exists for any integer $n > 1$.

Example

input

```
4
2
360
4999999937
4998207083
```

output

```
1
2
3
2 2 90
1
4999999937
1
4998207083
```

E. Number of Simple Paths

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

You are given an **undirected** graph consisting of n vertices and n edges. It is guaranteed that the given graph is **connected** (i. e. it is possible to reach any vertex from any other vertex) and there are no self-loops and multiple edges in the graph.

Your task is to calculate the number of **simple paths** of length **at least** 1 in the given graph. Note that paths that differ only by their direction are considered the same (i. e. you have to calculate the number of *undirected* paths). For example, paths $[1, 2, 3]$ and $[3, 2, 1]$ are considered the same.

You have to answer t independent test cases.

Recall that a path in the graph is a sequence of vertices $v_1, v_2, \ldots, v_k$ such that each pair of adjacent (consecutive) vertices in this sequence is connected by an edge. The length of the path is the number of edges in it. A **simple path** is such a path that all vertices in it are distinct.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($3 \le n \le 2 \cdot 10^5$) — the number of vertices (and the number of edges) in the graph.

The next n lines of the test case describe edges: edge i is given as a pair of vertices u_i, v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), where u_i and v_i are vertices the i-th edge connects. For each pair of vertices (u, v), there is at most one edge between u and v. There are no edges from the vertex to itself. So, there are no self-loops and multiple edges in the graph. The graph is **undirected**, i. e. all its edges are bidirectional. The graph is connected, i. e. it is possible to reach any vertex from any other vertex by moving along the edges of the graph.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

For each test case, print one integer: the number of **simple paths** of length **at least** 1 in the given graph. Note that paths that differ only by their direction are considered the same (i. e. you have to calculate the number of *undirected* paths).

Example

input

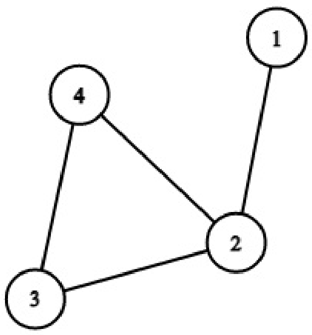
```
3
3
1 2
2 3
1 3
4
1 2
2 3
3 4
4 2
5
1 2
2 3
1 3
2 5
4 3
```

output

```
6
11
18
```

Note

Consider the second test case of the example. It looks like that:



There are 11 different simple paths:

1. $[1, 2]$;
2. $[2, 3]$;
3. $[3, 4]$;
4. $[2, 4]$;
5. $[1, 2, 4]$;
6. $[1, 2, 3]$;
7. $[2, 3, 4]$;
8. $[2, 4, 3]$;
9. $[3, 2, 4]$;
10. $[1, 2, 3, 4]$;

11. $[1, 2, 4, 3]$.

F. Array Partition

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

You are given an array a consisting of n integers.

Let $min(l, r)$ be the minimum value among $a_l, a_{l+1}, \ldots, a_r$ and $max(l, r)$ be the maximum value among $a_l, a_{l+1}, \ldots, a_r$.

Your task is to choose three **positive** (greater than 0) integers x, y and z such that:

- $x + y + z = n$;
- $max(1, x) = min(x + 1, x + y) = max(x + y + 1, n)$.

In other words, you have to split the array a into three consecutive non-empty parts that cover the whole array and the maximum in the first part equals the minimum in the second part and equals the maximum in the third part (or determine it is impossible to find such a partition).

Among all such triples (partitions), you can choose any.

You have to answer t independent test cases.

Input

The first line of the input contains one integer t ($1 \le t \le 2 \cdot 10^4$) — the number of test cases. Then t test cases follow.

The first line of the test case contains one integer n ($3 \le n \le 2 \cdot 10^5$) — the length of a.

The second line of the test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the i-th element of a.

It is guaranteed that the sum of n does not exceed $2 \cdot 10^5$ ($\sum n \le 2 \cdot 10^5$).

Output

For each test case, print the answer: NO in the only line if there is no such partition of a that satisfies the conditions from the problem statement. Otherwise, print YES in the first line and three integers x, y and z ($x + y + z = n$) in the second line.

If there are several answers, you can print any.

Example

input

```
6
11
1 2 3 3 3 4 4 3 4 2 1
8
2 9 1 7 3 9 4 1
9
2 1 4 2 4 3 3 1 2
7
4 2 1 1 4 1 4
5
1 1 1 1 1
7
4 3 4 3 3 3 4
```

output

```
YES
6 1 4
NO
YES
2 5 2
YES
4 1 2
YES
1 1 3
YES
2 1 4
```