A. Brick Wall

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

A brick is a strip of size $1 \times k$, placed horizontally or vertically, where k can be an arbitrary number that is at least 2 ($k \geq 2$).

A brick wall of size $n \times m$ is such a way to place several bricks inside a rectangle $n \times m$, that all bricks lie either horizontally or vertically in the cells, do not cross the border of the rectangle, and that each cell of the $n \times m$ rectangle belongs to exactly one brick. Here n is the height of the rectangle $n \times m$ and m is the width. **Note** that there can be bricks with different values of k in the same brick wall.

The wall stability is the difference between the number of horizontal bricks and the number of vertical bricks. **Note** that if you used 0 horizontal bricks and 2 vertical ones, then the stability will be -2**, not** 2.

What is the maximal possible stability of a wall of size $n \times m$?

It is guaranteed that under restrictions in the statement at least one $n \times m$ wall exists.

Input

The first line of the input contains one integer t ($1 \leq t \leq 10\,000$), the number of test cases.

The only line of each test case contains two integers n and m ($2 \leq n,\ m \leq 10^4$).

Output

For each test case, print one integer, the maximum stability of a wall of size $n \times m$.

Standard Input	Standard Output
5 2 2 7 8 16 9 3 5 10000 10000	2 28 64 6 50000000

Note

In the 1st test case, the maximum stability of 2 is obtained by placing two horizontal bricks 1×2 one on top of the other.

In the 2nd test case, one can get the maximum stability of 28 by placing 4 horizontal bricks 1×2 in each of the 7 rows.

,

B. Minimize Inversions

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You are given two permutations a and b of length n. A permutation is an array of n elements from 1 to n where all elements are distinct. For example, an array $[2, 1, 3]$ is a permutation, but $[0, 1]$ and $[1, 3, 1]$ aren't.

You can (as many times as you want) choose two indices i and j, then swap a_i with a_j and b_i with b_j simultaneously.

You hate inversions, so you want to minimize the total number of inversions in both permutations.

An inversion in a permutation p is a pair of indices (i, j) such that $i < j$ and $p_i > p_j$. For example, if $p = [3, 1, 4, 2, 5]$ then there are 3 inversions in it (the pairs of indices are $(1, 2)$, $(1, 4)$ and $(3, 4)$).

Input

The first line contains an integer t ($1 \le t \le 20\,000$) — the number of test cases.

Each test case consists of three lines. The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the permutations a and b. The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — permutation a. The third line contains b in a similar format.

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 two permutations a' and b' (in the same format as in the input) — the permutations after all operations. The total number of inversions in a' and b' should be the minimum possible among all pairs of permutations that can be obtained using operations from the statement.

If there are multiple solutions, print any of them.

Standard Input	Standard Output
3 5 1 2 3 4 5 5 4 3 2 1 3 3 1 2 3 1 2 6 2 5 6 1 3 4 1 5 3 6 2 4	3 2 5 1 4 3 4 1 5 2 1 2 3 1 2 3 2 3 4 6 5 1 1 2 4 3 5 6

Note

In the first test case, the minimum possible number of inversions is 10.

In the second test case, we can sort both permutations at the same time. For this, the following operations can be done:

- Swap the elements in the positions 1 and 3 in both permutations. After the operation, $a = [2, 1, 3]$, $b = [2, 1, 3]$.
- Swap the elements in the positions 1 and 2. After the operations, a and b are sorted.

In the third test case, the minimum possible number of inversions is 7.

,

C. XOR-distance

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

You are given integers a, b, r. Find the smallest value of $|(a \oplus x) - (b \oplus x)|$ among all $0 \le x \le r$.

$\oplus$ is the operation of bitwise XOR, and $|y|$ is absolute value of y.

Input

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

Each test case contains integers a, b, r ($0 \le a, b, r \le 10^{18}$).

Output

For each test case, output a single number — the smallest possible value.

Standard Input	Standard Output
10 4 6 0 0 3 2 9 6 10 92 256 23 165 839 201 1 14 5 2 7 2 96549 34359 13851 853686404475946 283666553522252166 127929199446003072 735268590557942972 916721749674600979 895150420120690183	2 1 1 164 542 5 3 37102 27934920819538516 104449824168870225

Note

In the first test, when $r = 0$, then x is definitely equal to 0, so the answer is $|4 \oplus 0 - 6 \oplus 0| = |4 - 6| = 2$.

In the second test:

- When $x = 0$, $|0 \oplus 0 - 3 \oplus 0| = |0 - 3| = 3$.
- When $x = 1$, $|0 \oplus 1 - 3 \oplus 1| = |1 - 2| = 1$.
- When $x = 2$, $|0 \oplus 2 - 3 \oplus 2| = |2 - 1| = 1$.

Therefore, the answer is 1.

In the third test, the minimum is achieved when $x = 1$.

,

D. Blocking Elements

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 256 megabytes

You are given an array of numbers $a_1, a_2, \ldots, a_n$. Your task is to block some elements of the array in order to minimize its cost. Suppose you block the elements with indices $1 \le b_1 < b_2 < \ldots < b_m \le n$. Then the cost of the array is calculated as the maximum of:

- the sum of the blocked elements, i.e., $a_{b_1} + a_{b_2} + \ldots + a_{b_m}$.
- the maximum sum of the segments into which the array is divided when the blocked elements are removed. That is, the maximum sum of the following $(m + 1)$ subarrays: $[1, b_1 - 1]$, $[b_1 + 1, b_2 - 1]$, $[\ldots]$, $[b_{m-1} + 1, b_m - 1]$, $[b_m + 1, n]$ (the sum of numbers in a subarray of the form $[x, x - 1]$ is considered to be 0).

For example, if $n = 6$, the original array is $[1, 4, 5, 3, 3, 2]$, and you block the elements at positions 2 and 5, then the cost of the array will be the maximum of the sum of the blocked elements ($4 + 3 = 7$) and the sums of the subarrays (1, $5 + 3 = 8$, 2), which is $\max(7, 1, 8, 2) = 8$.

You need to output the minimum cost of the array after blocking.

Input

The first line of the input contains a single integer t ($1 \le t \le 30\,000$) — the number of queries.

Each test case consists of two lines. The first line contains an integer n ($1 \le n \le 10^5$) — the length of the array a. The second line contains n elements $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — 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 number — the minimum cost of blocking the array.

Standard Input	Standard Output
3 6 1 4 5 3 3 2 5 1 2 3 4 5 6 4 1 6 3 10 7	7 5 11

Note

The first test case matches with the array from the statement. To obtain a cost of 7, you need to block the elements at positions 2 and 4. In this case, the cost of the array is calculated as the maximum of:

- the sum of the blocked elements, which is $a_2 + a_4 = 7$.
- the maximum sum of the segments into which the array is divided when the blocked elements are removed, i.e., the maximum of a_1, a_3, $a_5 + a_6 = \max(1, 5, 5) = 5$.

So the cost is $\max(7, 5) = 7$.

In the second test case, you can block the elements at positions 1 and 4.

In the third test case, to obtain the answer 11, you can block the elements at positions 2 and 5. There are other ways to get this answer, for example, blocking positions 4 and 6.

,

E. ace5 and Task Order

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

`This is an interactive problem!`

In the new round, there were n tasks with difficulties from 1 to n. The coordinator, who decided to have the first round with tasks in unsorted order of difficulty, rearranged the tasks, resulting in a permutation of difficulties from 1 to n. After that, the coordinator challenged ace5 to guess the permutation in the following way.

Initially, the coordinator chooses a number x from 1 to n.

ace5 can make queries of the form: $?\ i$. The answer will be:

- $>$, if $a_i > x$, after which x increases by 1.
- $<$, if $a_i < x$, after which x decreases by 1.
- $=$, if $a_i = x$, after which x remains unchanged.

The task for ace5 is to guess the permutation in no more than $40n$ queries. Since ace5 is too busy writing the announcement, he has entrusted this task to you.

Input

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

Interaction

The interaction between your program and the jury's program begins with reading a positive integer n ($1 \le n \le 2000$) — the length of the hidden permutation.

To make a query, output a line in the format "`? i`", where $1 \le i \le n$.

As an answer, you will receive:

- ">", if a_i > x, after which x will increase by 1.
- "<", if a_i < x, after which x will decrease by 1.
- "=", if a_i = x, after which x will remain unchanged.

You can make no more than $40n$ queries. To output the answer, you need to print "`! a_1 a_2 ... a_n`", where $1 \le a_i \le n$, and all of them are distinct. Outputting the answer **does not count** as a query.

If your program makes more than $40n$ queries for one test case, or makes an invalid query, then the **response to the query will be** `-1`. After receiving such a response, your program should immediately terminate to receive the verdict `Wrong Answer`. Otherwise, it may receive any other verdict.

After outputting a query, do not forget to print a newline and flush the output buffer. Otherwise, you will receive the verdict `Presentation Error`. To flush the buffer, 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.

It is guaranteed that the sum of n over all test cases does not exceed 2000.

The interactor in this problem is **not adaptive**.

Hacks:

To make a hack, use the following format:

The first line contains a single integer t — the number of test cases.

The description of each test case should consist of two lines. The first line contains the numbers n and x ($1 \le x \le n \le 2000$) — the length of the hidden permutation and the initial value of the number x. The second line contains n distinct numbers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the permutation that the jury should choose in this test case.

Sum of n over all test cases should not exceed 2000.

Standard Input	Standard Output
2 5 > = < = < < 2 >	? 4 ? 2 ? 1 ? 5 ? 1 ? 3 ! 2 4 1 5 3 ? 1 ! 2 1

Note

In the first test, the hidden permutation is $a = [2, 4, 1, 5, 3]$, and the initial value of x is 3.

In the second test, the hidden permutation is $a = [2, 1]$, and the initial value of x is 1.

,

F. Caterpillar on a Tree

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

The caterpillar decided to visit every node of the tree. Initially, it is sitting at the root.

The tree is represented as a rooted tree with the root at the node 1. Each crawl to a neighboring node takes 1 minute for the caterpillar.

And there is a trampoline under the tree. If the caterpillar detaches from the tree and falls onto the trampoline, it will end up at the root of the tree in 0 seconds. But the trampoline is old and can withstand no more than k caterpillar's falls.

What is the minimum time the caterpillar can take to visit all the nodes of the tree?

More formally, we need to find the minimum time required to visit all the nodes of the tree, if the caterpillar starts at the root (node 1) and moves using two methods.

1. Crawl along an edge to one of the neighboring nodes: takes 1 minute.
2. Teleport to the root: takes no time, no new nodes become visited.

The second method (teleportation) can be used at most k times. The caterpillar can finish the journey at any node.

Input

The first line of the input contains two integers: n ($2 \le n \le 2 \cdot 10^5$) — the number of nodes in the tree, and k ($0 \le k \le 10^9$) — the maximum number of teleports to the root.

The second line contains $n - 1$ integers p_2, p_3, ..., p_n ($1 \le p_i \le n$) — the ancestors in the tree for nodes $2, 3, \ldots, n$; node 1 is the root.

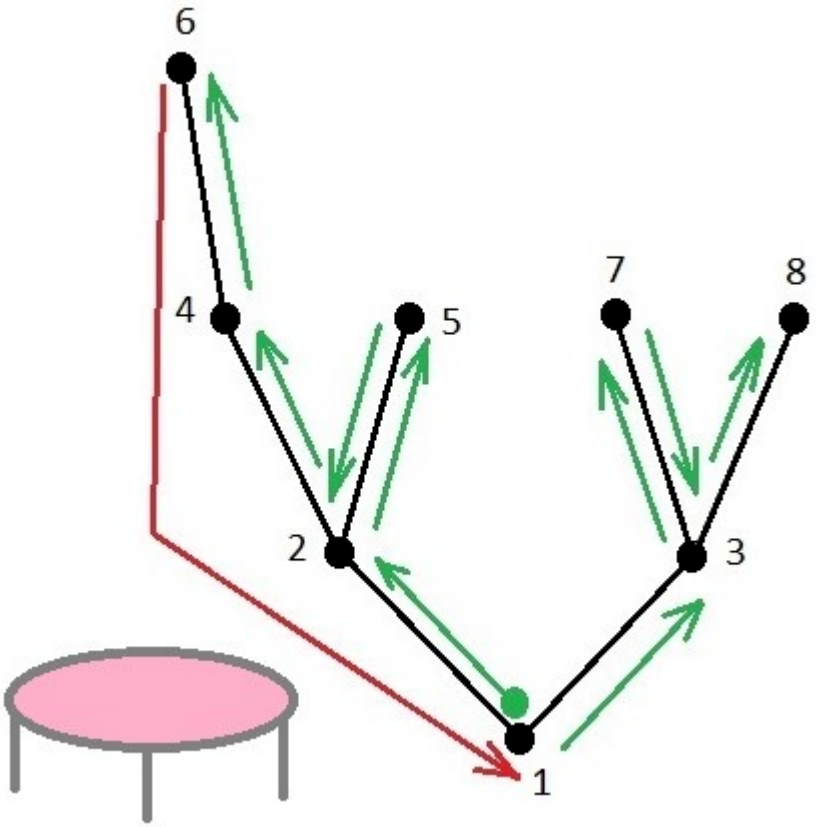
Output

Print a single number in a single line — the minimum number of minutes required to visit all the nodes of the tree.

Standard Input	Standard Output
8 1 1 1 2 2 4 3 3	9
4 0 4 4 1	4

Note

The figure shows one of the ways to traverse the tree from the first test in 9 minutes.


6
4
5
7
8
2
3
1

,

G. Permutation of Given

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

You were given only one number, n. It didn't seem interesting to you, so you wondered if it's possible to come up with an array of length n consisting of non-zero integers, such that if each element of the array is replaced by the sum of its neighbors (the elements on the ends are replaced by their only neighbors), you obtain a permutation of the numbers in the original array.

Input

Each test case contains only one number, n ($2 \le n \le 10^6$).

Output

If a solution exists, output `"YES"` (without quotes), followed by an array a ($-10^9 \le a_i \le 10^9, a_i \neq 0$) that satisfies the condition of the problem. If there are multiple possible answers, output any of them.

If there is no suitable array, output `"NO"` (without quotes).

The words `"YES"` and `"NO"` can be output in any case, for example, `"YES"`, `"Yes"`, `"yEs"`, and so on.

Standard Input	Standard Output
4	YES 1 2 -2 -1
5	NO

Note

In the first test, the array $[1, 2, -2, -1]$ is suitable, because if each element is replaced by the sum of its neighbors, the resulting array is $[2, -1, 1, -2]$, which is a permutation of the original array.

In the second test, it can be shown that there is no solution.