A. Shuffle Party

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

You are given an array $a_1, a_2, \ldots, a_n$. Initially, $a_i = i$ for each $1 \le i \le n$.

The operation $\texttt{swap}(k)$ for an integer $k \ge 2$ is defined as follows:

- Let d be the largest divisor† of k which is not equal to k itself. Then swap the elements a_d and a_k.

Suppose you perform $\texttt{swap}(i)$ for each $i = 2, 3, \ldots, n$ in this exact order. Find the position of 1 in the resulting array. In other words, find such j that $a_j = 1$ after performing these operations.

† An integer x is a divisor of y if there exists an integer z such that $y = x \cdot z$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The only line of each test case contains one integer n ($1 \le n \le 10^9$) — the length of the array a.

Output

For each test case, output the position of 1 in the resulting array.

Standard Input	Standard Output
4 1 4 5 120240229	1 4 4 67108864

Note

In the first test case, the array is $[1]$ and there are no operations performed.

In the second test case, a changes as follows:

- Initially, a is $[1, 2, 3, 4]$.
- After performing $\texttt{swap}(2)$, a changes to $[\underline{2}, \underline{1}, 3, 4]$ (the elements being swapped are underlined).
- After performing $\texttt{swap}(3)$, a changes to $[\underline{3}, 1, \underline{2}, 4]$.
- After performing $\texttt{swap}(4)$, a changes to $[3, \underline{4}, 2, \underline{1}]$.

Finally, the element 1 lies on index 4 (that is, $a_4 = 1$). Thus, the answer is 4.

,

B. Binary Path

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

You are given a $2 \times n$ grid filled with zeros and ones. Let the number at the intersection of the i-th row and the j-th column be a_{ij}.

There is a grasshopper at the top-left cell $(1, 1)$ that can only jump one cell right or downwards. It wants to reach the bottom-right cell $(2, n)$. Consider the binary string of length $n + 1$ consisting of numbers written in cells of the path without changing their order.

Your goal is to:

1. Find the lexicographically smallest[†] string you can attain by choosing any available path;
2. Find the number of paths that yield this lexicographically smallest string.

[†] If two strings s and t have the same length, then s is lexicographically smaller than t if and only if in the first position where s and t differ, the string s has a smaller element than the corresponding element in t.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The 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 second line of each test case contains a binary string $a_{11}a_{12}\ldots a_{1n}$ (a_{1i} is either 0 or 1).

The third line of each test case contains a binary string $a_{21}a_{22}\ldots a_{2n}$ (a_{2i} is either 0 or 1).

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

Output

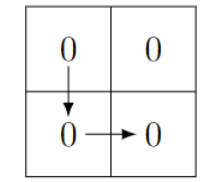
For each test case, output two lines:

1. The lexicographically smallest string you can attain by choosing any available path;
2. The number of paths that yield this string.

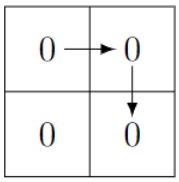
Standard Input	Standard Output
3 2 00 00 4 1101 1100 8 00100111 11101101	000 2 11000 1 001001101 4

Note

In the first test case, the lexicographically smallest string is `000`. There are two paths that yield this string:

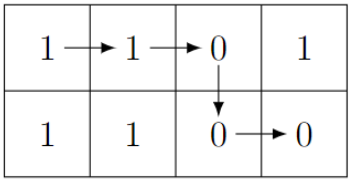


The first path



The second path

In the second test case, the lexicographically smallest string is `11000`. There is only one path that yields this string:



,

C. Bitwise Operation Wizard

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

This is an interactive problem.

There is a secret sequence $p_0, p_1, \ldots, p_{n-1}$, which is a permutation of $\{0, 1, \ldots, n-1\}$.

You need to find any two indices i and j such that $p_i \oplus p_j$ is maximized, where $\oplus$ denotes the bitwise XOR operation.

To do this, you can ask queries. Each query has the following form: you pick arbitrary indices a, b, c, and d ($0 \le a, b, c, d < n$). Next, the jury calculates $x = (p_a \mid p_b)$ and $y = (p_c \mid p_d)$, where $\mid$ denotes the bitwise OR operation. Finally, you receive the result of comparison between x and y. In other words, you are told if $x < y$, $x > y$, or $x = y$.

Please find any two indices i and j ($0 \le i, j < n$) such that $p_i \oplus p_j$ is maximum among all such pairs, using at most $3n$ queries. If there are multiple pairs of indices satisfying the condition, you may output any one of them.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

Interaction

The first line of each test case contains one integer n ($2 \le n \le 10^4$). At this moment, the permutation $p_0, p_1, \ldots, p_{n-1}$ is chosen. The interactor in this task is **not adaptive**. In other words, the sequence p is fixed in every test case and does not change during the interaction.

To ask a query, you need to pick four indices a, b, c, and d ($0 \le a, b, c, d < n$) and print the line of the following form:

- "`? a b c d`"

After that, you receive:

- "<" if $(p_a \mid p_b) < (p_c \mid p_d)$;
- "=" if $(p_a \mid p_b) = (p_c \mid p_d)$;
- ">" if $(p_a \mid p_b) > (p_c \mid p_d)$.

You can make at most $3n$ queries of this form.

Next, if your program has found a pair of indices i and j ($0 \le i, j < n$) such that $p_i \oplus p_j$ is maximized, print the line of the following form:

- "`! i j`"

Note that this line is **not** considered a query and is **not** taken into account when counting the number of queries asked.

After this, proceed to the next test case.

If you make more than $3n$ queries during an interaction, your program must terminate immediately, and you will receive the `Wrong Answer` verdict. Otherwise, you can get an arbitrary verdict because your solution will continue to read from a closed stream.

After printing a query or the answer for a test case, do not forget to output the end of line and flush the output. Otherwise, you will get the verdict `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.

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

Hacks

To hack, follow the test format below.

The first line contains the number of test cases t ($1 \le t \le 10^3$). The description of the test cases follows.

The first line of each test case contains one integer n ($2 \le n \le 10^4$).

The second line of each test case contains n integers $p_0, p_1, \ldots, p_{n-1}$, which represent a permutation of integers from 0 to $n - 1$.

The sum of n over all test cases should not exceed 10^4.

Standard Input	Standard Output
2 4 < = > 2	 ? 0 2 3 1 ? 1 1 2 3 ? 1 2 0 3 ! 3 2 ! 0 1

Note

In the first test case, the hidden permutation is $p = [0, 3, 1, 2]$.

For the query "? 0 2 3 1", the jury return "<" because $(p_0 \mid p_2) = (0 \mid 1) = 1 < (p_3 \mid p_1) = (2 \mid 3) = 3$.

For the query "? 1 1 2 3", the jury return "=" because $(p_1 \mid p_1) = (3 \mid 3) = 3 = (p_2 \mid p_3) = (1 \mid 2) = 3$.

For the query "? 1 2 0 3", the jury return ">" because $(p_1 \mid p_2) = (3 \mid 1) = 3 > (p_0 \mid p_3) = (0 \mid 2) = 2$.

The answer $i = 3$ and $j = 2$ is valid: $(p_3 \oplus p_2) = (2 \oplus 1) = 3$ is indeed equal to the maximum possible value of $p_i \oplus p_j$. Another valid answer would be $i = 0$ and $j = 1$. As the number of queries does not exceed $3n = 12$, the answer is considered correct.

In the second test case, $n = 2$, so p is either $[0, 1]$ or $[1, 0]$. In any case, $p_0 \oplus p_1 = 1$ is maximum possible.

,

D. Pinball

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

There is a one-dimensional grid of length n. The i-th cell of the grid contains a character s_i, which is either '<' or '>'.

When a pinball is placed on one of the cells, it moves according to the following rules:

- If the pinball is on the i-th cell and s_i is '<', the pinball moves one cell to the left in the next second. If s_i is '>', it moves one cell to the right.
- After the pinball has moved, the character s_i is inverted (i. e. if s_i used to be '<', it becomes '>', and vice versa).
- The pinball stops moving when it leaves the grid: either from the left border or from the right one.

You need to answer n **independent** queries. In the i-th query, a pinball will be placed on the i-th cell. Note that we always place a pinball on the initial grid.

For each query, calculate how many seconds it takes the pinball to leave the grid. It can be shown that the pinball will always leave the grid within a finite number of steps.

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 an integer n ($1 \le n \le 5 \cdot 10^5$).

The second line of each test case contains a string $s_1 s_2 \ldots s_n$ of length n consisting of characters '<' and '>'.

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

Output

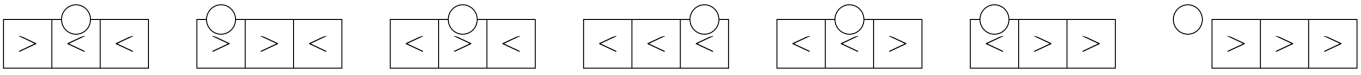
For each test case, for each i ($1 \le i \le n$) output the answer if a pinball is initially placed on the i-th cell.

Standard Input	Standard Output
3 3 ><< 4 <<<< 6 <><<<>	3 6 5 1 2 3 4 1 4 7 10 8 1

Note

In the first test case, the movement of the pinball for $i = 1$ is shown in the following pictures. It takes the pinball 3 seconds to leave the grid.

The movement of the pinball for $i = 2$ is shown in the following pictures. It takes the pinball 6 seconds to leave the grid.



,

E. Pokémon Arena

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 512 megabytes

You are at a dueling arena. You also possess n Pokémons. Initially, only the 1-st Pokémon is standing in the arena.

Each Pokémon has m attributes. The j-th attribute of the i-th Pokémon is $a_{i,j}$. Each Pokémon also has a cost to be hired: the i-th Pokémon's cost is c_i.

You want to have the n-th Pokémon stand in the arena. To do that, you can perform the following two types of operations any number of times in any order:

- Choose three integers i, j, k ($1 \le i \le n$, $1 \le j \le m$, $k > 0$), increase $a_{i,j}$ by k permanently. The cost of this operation is k.
- Choose two integers i, j ($1 \le i \le n$, $1 \le j \le m$) and hire the i-th Pokémon to duel with the current Pokémon in the arena based on the j-th attribute. The i-th Pokémon will win if $a_{i,j}$ is **greater than or equal to** the j-th attribute of the current Pokémon in the arena (otherwise, it will lose). After the duel, only the winner will stand in the arena. The cost of this operation is c_i.

Find the minimum cost you need to pay to have the n-th Pokémon stand in the arena.

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 two integers n and m ($2 \le n \le 4 \cdot 10^5$, $1 \le m \le 2 \cdot 10^5$, $2 \le n \cdot m \le 4 \cdot 10^5$).

The second line of each test case contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^9$).

The i-th of the following n lines contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($1 \le a_{i,j} \le 10^9$).

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $4 \cdot 10^5$.

Output

For each test case, output the minimum cost to make the n-th Pokémon stand in the arena.

Standard Input	Standard Output
4 3 3 2 3 1 2 9 9 6 1 7 1 2 1 3 3 2 3 1 9 9 9 6 1 7	2 6 17 1224474550

1 2 1 4 2 2 8 3 5 18 24 17 10 1 10 1 1 6 3 21412674 3212925 172015806 250849370 306960171 333018900 950000001 950000001 950000001 821757276 783362401 760000001 570000001 700246226 600757652 380000001 423513575 474035234 315201473 300580025 287023445 1 1 1	

Note

In the first test case, the attribute array of the 1-st Pokémon (which is standing in the arena initially) is $[2, 9, 9]$.

In the first operation, you can choose $i = 3$, $j = 1$, $k = 1$, and increase $a_{3,1}$ by 1 permanently. Now the attribute array of the 3-rd Pokémon is $[2, 2, 1]$. The cost of this operation is $k = 1$.

In the second operation, you can choose $i = 3$, $j = 1$, and hire the 3-rd Pokémon to duel with the current Pokémon in the arena based on the 1-st attribute. Since $a_{i,j} = a_{3,1} = 2 \geq 2 = a_{1,1}$, the 3-rd Pokémon will win. The cost of this operation is $c_3 = 1$.

Thus, we have made the 3-rd Pokémon stand in the arena within the cost of 2. It can be proven that 2 is minimum possible.

In the second test case, the attribute array of the 1-st Pokémon in the arena is $[9, 9, 9]$.

In the first operation, you can choose $i = 2$, $j = 3$, $k = 2$, and increase $a_{2,3}$ by 2 permanently. Now the attribute array of the 2-nd Pokémon is $[6, 1, 9]$. The cost of this operation is $k = 2$.

In the second operation, you can choose $i = 2$, $j = 3$, and hire the 2-nd Pokémon to duel with the current Pokémon in the arena based on the 3-rd attribute. Since $a_{i,j} = a_{2,3} = 9 \geq 9 = a_{1,3}$, the 2-nd Pokémon will win. The cost of this operation is $c_2 = 3$.

In the third operation, you can choose $i = 3$, $j = 2$, and hire the 3-rd Pokémon to duel with the current Pokémon in the arena based on the 2-nd attribute. Since $a_{i,j} = a_{1,2} = 2 \geq 1 = a_{2,2}$, the 3-rd Pokémon can win. The cost of this operation is $c_3 = 1$.

Thus, we have made the 3-rd Pokémon stand in the arena within the cost of 6. It can be proven that 6 is minimum possible.

,

F. Bitwise Paradox

```
Input file:     standard input
Output file:    standard output
Time limit:     5 seconds
Memory limit:   1024 megabytes
```

You are given two arrays a and b of size n along with a fixed integer v.

An interval $[l, r]$ is called a **good** interval if $(b_l \mid b_{l+1} \mid \ldots \mid b_r) \geq v$, where $\mid$ denotes the bitwise OR operation. The **beauty** of a good interval is defined as $\max(a_l, a_{l+1}, \ldots, a_r)$.

You are given q queries of two types:

- "`1 i x`": assign $b_i := x$;
- "`2 l r`": find the **minimum** beauty among all **good** intervals $[l_0, r_0]$ satisfying $l \leq l_0 \leq r_0 \leq r$. If there is no suitable good interval, output -1 instead.

Please process all queries.

Input

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

The first line of each test case contains two integers n and v ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq v \leq 10^9$).

The second line of each testcase contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$).

The third line of each testcase contains n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq 10^9$).

The fourth line of each testcase contains one integer q ($1 \leq q \leq 2 \cdot 10^5$).

The i-th of the following q lines contains the description of queries. Each line is of one of two types:

- "`1 i x`" ($1 \leq i \leq n$, $1 \leq x \leq 10^9$);
- "`2 l r`" ($1 \leq l \leq r \leq n$).

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

Output

For each test case, output the answers for all queries of the second type.

Standard Input	Standard Output
3 3 7 2 1 3 2 2 3 4 2 1 3 1 2 5 2 2 3 2 1 3 4 5	-1 3 2 5 2 2 1 -1

5 1 2 4 4 2 3 3 6 2 1 4 1 3 15 2 3 4 2 2 4 1 2 13 2 1 4 1 5 6 4 1 2 1 1	

Note

In the first test case, $a = [2, 1, 3]$, $b = [2, 2, 3]$, and $v = 7$.

The first query is of the second type and has $l = 1$ and $r = 3$. The largest interval available is $[1, 3]$, and its bitwise OR is $b_1 \mid b_2 \mid b_3 = 3$ which is less than v. Thus, no good interval exists.

The second query asks to change b_2 to 5, so b becomes $[2, 5, 3]$.

The third query is of the second type and has $l = 2$ and $r = 3$. There are three possible intervals: $[2, 2]$, $[3, 3]$, and $[2, 3]$. However, $b_2 = 5 < v$, $b_3 = 3 < v$. So only the last interval is good: it has $b_2 \mid b_3 = 7$. The answer is thus $\max(a_2, a_3) = 3$.

The fourth query is of the second type and has $l = 1$ and $r = 3$. There are three good intervals: $[1, 2]$, $[2, 3]$, and $[1, 3]$. Their beauty is 2, 3, 3 correspondingly. The answer is thus 2.

In the second test case, $a = [5, 1, 2, 4]$, $b = [4, 2, 3, 3]$, and $v = 5$.

The first query has $l = 1$ and $r = 4$. The only good intervals are: $[1, 2]$, $[1, 3]$, $[1, 4]$. Their beauty is 5, 5, 5 correspondingly. The answer is thus 5.