A. Wonderful Sticks

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

You are the proud owner of n sticks. Each stick has an integer length from 1 to n. The lengths of the sticks are **distinct**.

You want to arrange the sticks in a row. There is a string s of length $n-1$ that describes the requirements of the arrangement.

Specifically, for each i from 1 to $n-1$:

- If $s_i =$ <, then the length of the stick at position $i+1$ must be **smaller** than all sticks before it;
- If $s_i =$ >, then the length of the stick at position $i+1$ must be **larger** than all sticks before it.

Find any valid arrangement of sticks. We can show that an answer always exists.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 500$). The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 100$) — the number of sticks.

The second line of each test case contains a single string s of length $n-1$ consisting of characters < and > — describing the requirements of the arrangement.

Output

For each test case, output n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$, the a_i are distinct) — the lengths of the sticks in order. If there are multiple solutions, print any of them.

Standard Input	Standard Output

5
2
<
5
<<><
2
>
3
<>
7
><>>><
	2 1
	4 3 2 5 1
	1 2
	2 1 3
	3 4 2 5 6 7 1

Note

For the first test case, the requirements of the arrangement are as follows:

- $s_1 =$ <, which means a_2 is smaller than a_1.

Thus, one possible arrangement is $[2, 1]$.

For the second test case, the requirements of the arrangement are as follows:

- $s_1 =$ <, which means a_2 is smaller than a_1;
- $s_2 =$ <, which means a_3 is smaller than a_1 and a_2;
- $s_3 =$ >, which means a_4 is larger than a_1, a_2, and a_3;
- $s_4 =$ <, which means a_5 is smaller than a_1, a_2, a_3, and a_4.

Thus, one possible arrangement is $[4, 3, 2, 5, 1]$.

,

B. Wonderful Gloves

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

You are the proud owner of many colorful gloves, and you keep them in a drawer. Each glove is in one of n colors numbered 1 to n. Specifically, for each i from 1 to n, you have l_i left gloves and r_i right gloves with color i.

Unfortunately, it's late at night, so **you can't see any of your gloves**. In other words, you will only know the color and the type (left or right) of a glove **after** you take it out of the drawer.

A matching pair of gloves with color i consists of exactly one left glove and one right glove with color i. Find the minimum number of gloves you need to take out of the drawer to **guarantee** that you have **at least** k matching pairs of gloves with **different** colors.

Formally, find the smallest positive integer x such that:

- For any set of x gloves you take out of the drawer, there will always be at least k matching pairs of gloves with different colors.

Input

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

The first line of each test case contains two integers n, k ($1 \leq k \leq n \leq 2 \cdot 10^5$) — the number of different colors, and the minimum number of required matching pairs of gloves.

The second line of each test case contains n integers $l_1, l_2, \ldots, l_n$ ($1 \leq l_i \leq 10^9$) — the number of left gloves with color i for each i from 1 to n.

The third line of each test case contains n integers $r_1, r_2, \ldots, r_n$ ($1 \leq r_i \leq 10^9$) — the number of right gloves with color i for each i from 1 to n.

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 a single integer — the minimum number of gloves you need to take out of the drawer.

Standard Input	Standard Output

5
3 3
1 1 1
1 1 1
1 1
100
1
3 2
100 1 1
	6
	101
	303
	481
	1010

```
200 1 1
5 2
97 59 50 87 36
95 77 33 13 74
10 6
97 59 50 87 36 95 77 33 13 74
91 14 84 33 54 89 68 34 14 15
```

Note

In the first test case, you must take out all of the gloves, so the answer is 6.

In the second test case, the answer is 101. If you take out 100 gloves or fewer, then it is possible that all of them are left gloves, which means you won't have a matching pair of gloves.

In the third test case, the answer is 303. If you only take out 302 gloves, then one possible scenario is as follows:

- Color 1: 100 left gloves, 200 right gloves
- Color 2: 1 left glove, 0 right gloves
- Color 3: 0 left gloves, 1 right glove

You only have multiple matching pairs of gloves with color 1. So you won't have at least 2 matching pairs of gloves with different colors.

,

C. Wonderful City

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

You are the proud leader of a city in Ancient Berland. There are n^2 buildings arranged in a grid of n rows and n columns. The height of the building in row i and column j is $h_{i,j}$.

The city is *beautiful* if no two adjacent by side buildings have the same height. In other words, it must satisfy the following:

- There **does not** exist a position (i, j) ($1 \le i \le n$, $1 \le j \le n-1$) such that $h_{i,j} = h_{i,j+1}$.
- There **does not** exist a position (i, j) ($1 \le i \le n-1$, $1 \le j \le n$) such that $h_{i,j} = h_{i+1,j}$.

There are n workers at company A, and n workers at company B. Each worker can be hired **at most once**.

It costs a_i coins to hire worker i at company A. After hiring, worker i will:

- Increase the heights of all buildings in row i by 1. In other words, increase $h_{i,1}, h_{i,2}, \ldots, h_{i,n}$ by 1.

It costs b_j coins to hire worker j at company B. After hiring, worker j will:

- Increase the heights of all buildings in column j by 1. In other words, increase $h_{1,j}, h_{2,j}, \ldots, h_{n,j}$ by 1.

Find the minimum number of coins needed to make the city beautiful, or report that it is impossible.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 1000$) — the size of the grid.

The i-th of the next n lines of each test case contains n integers $h_{i,1}, h_{i,2}, \ldots, h_{i,n}$ ($1 \le h_{i,j} \le 10^9$) — the heights of the buildings in row i.

The next line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the costs of hiring the workers at company A.

The next line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_j \le 10^9$) — the costs of hiring the workers at company B.

It is guaranteed that the sum of n over all test cases does not exceed 1000.

Output

For each test case, output a single integer — the minimum number of coins needed, or -1 if it is impossible.

Standard Input	Standard Output

4
2
1 2
2 1
100 100
	0
	14
	-1
	183

```
100 100
4
1 2 1 2
3 2 1 2
1 2 1 1
1 3 1 2
1 2 3 4
5 6 7 8
3
1 2 2
2 2 1
2 1 1
100 100 100
100 100 100
6
8 7 2 8 4 8
7 7 9 7 1 1
8 3 1 1 8 5
6 8 3 1 1 4
1 4 5 1 9 6
7 1 1 6 8 2
11 23 20 79 30 15
15 83 73 57 34 63
```

Note

For the first test case, we can see that the city is already beautiful. Thus, the answer is 0.

For the second test case, we can hire worker 2 from company A, worker 4 from company A, and worker 4 from company B:

$$\begin{array}{|c|c|c|c|}\hline 1 & 2 & 1 & 2 \\ \hline 3 & 2 & 1 & 2 \\ \hline 1 & 2 & 1 & 1 \\ \hline 1 & 3 & 1 & 2 \\ \hline \end{array} \Longrightarrow \begin{array}{|c|c|c|c|}\hline 1 & 2 & 1 & 3 \\ \hline 4 & 3 & 2 & 4 \\ \hline 1 & 2 & 1 & 2 \\ \hline 2 & 4 & 2 & 4 \\ \hline \end{array}$$

The cost of hiring the workers is $2 + 4 + 8 = 14$. This is the minimum possible cost.

For the third test case, no matter what we do, it is impossible to make the city beautiful. Thus, the answer is -1.

D. Wonderful Lightbulbs

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

You are the proud owner of an infinitely large grid of lightbulbs, represented by a Cartesian coordinate system. Initially, all of the lightbulbs are turned off, except for one lightbulb, where you buried your proudest treasure.

In order to hide your treasure's position, you perform the following operation an arbitrary number of times (possibly zero):

- Choose two **integer** numbers x and y, and switch the state of the 4 lightbulbs at (x, y), $(x, y + 1)$, $(x + 1, y - 1)$, and $(x + 1, y)$. In other words, for each lightbulb, turn it on if it was off, and turn it off if it was on. Note that there are **no constraints** on x and y.

In the end, there are n lightbulbs turned on at coordinates $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Unfortunately, you have already forgotten where you buried your treasure, so now you have to figure out one possible position of the treasure. Good luck!

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 ($1 \le n \le 2 \cdot 10^5$) — the number of lightbulbs that are on.

The i-th of the next n lines contains two integers x_i and y_i ($-10^8 \le x_i, y_i \le 10^8$) — the coordinates of the i-th lightbulb. It is guaranteed that all coordinates are distinct.

Additional constraint: There exists at least one position (s, t) ($-10^9 \le s, t \le 10^9$), such that if the lightbulb at position (s, t) is initially turned on, then after performing an arbitrary number of operations (possibly zero), we will get the given configuration of lightbulbs.

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 integers s and t ($-10^9 \le s, t \le 10^9$) — one possible position of the buried treasure. If there are multiple solutions, print any of them.

For this problem, hacks are disabled.

Standard Input	Standard Output

4
1
2 3
3
-2 -1
-1 -2
-1 -3
7
	2 3
	-2 -2
	7 27
	72 7

```
7 26
6 27
6 28
7 27
8 26
8 27
7 28
11
70 9
69 8
69 0
73 5
70 -1
70 5
71 7
70 4
73 4
71 3
72 3
```

Note

For the first test case, one possible scenario is that you hid your treasure at position $(2, 3)$. Then, you did not perform any operations.

In the end, only the lightbulb at $(2, 3)$ is turned on.

For the second test case, one possible scenario is that you hid your treasure at position $(-2, -2)$. Then, you performed 1 operation with $x = -2$, $y = -2$.

The operation switches the state of the 4 lightbulbs at $(-2, -2)$, $(-2, -1)$, $(-1, -3)$, and $(-1, -2)$.

In the end, the lightbulbs at $(-2, -1)$, $(-1, -2)$, and $(-1, -3)$ are turned on.

E. Wonderful Teddy Bears

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

You are the proud owner of n teddy bears, which are arranged in a row on a shelf. Each teddy bear is colored either black or pink.

An arrangement of teddy bears is *beautiful* if all the black teddy bears are to the left of all the pink teddy bears. In other words, there **does not** exist a pair of indices (i, j) $(1 \le i < j \le n)$ such that the i-th teddy bear is pink, and the j-th teddy bear is black.

You want to reorder the teddy bears into a beautiful arrangement. You are too short to reach the shelf, but luckily, you can send instructions to a robot to move the teddy bears around. In a single instruction, the robot can:

- Choose an index i $(1 \le i \le n-2)$ and reorder the teddy bears at positions i, $i+1$ and $i+2$ so that all the black teddy bears are to the left of all the pink teddy bears.

What is the minimum number of instructions needed to reorder the teddy bears?

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 $(3 \le n \le 2 \cdot 10^5)$ — the number of teddy bears.

The second line of each test case contains a single string s of length n consisting of characters B and P — the colors of the teddy bears. For each i from 1 to n, the i-th teddy bear is colored black if $s_i = \mathtt{B}$ and pink if $s_i = \mathtt{P}$.

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 a single integer — the minimum number of instructions needed to reorder the teddy bears.

Standard Input	Standard Output
5 3 PPP 3 BPP 3 PPB 7 PPBPPBB 15 BPBPBBBBBPBBBBB	0 0 1 5 14

Note

For the first test case, all the teddy bears are pink. Thus, the arrangement is already beautiful, so the answer is 0.

For the second test case, all the black teddy bears are to the left of all the pink teddy bears. Thus, the answer is 0.

For the third test case, we can perform 1 instruction with $i = 1$.

After the instruction, the sequence of colors changes from $\texttt{PPB}$ to $\texttt{BPP}$, and we are done.

For the fourth test case, we can perform 5 instructions as follows:

- $i = 1$: $\texttt{PPBPPBB} \to \texttt{BPPPPBB}$
- $i = 5$: $\texttt{BPPPPBB} \to \texttt{BPPPBBP}$
- $i = 4$: $\texttt{BPPPBBP} \to \texttt{BPPBBPP}$
- $i = 3$: $\texttt{BPPBBPP} \to \texttt{BPBBPPP}$
- $i = 2$: $\texttt{BPBBPPP} \to \texttt{BBBPPPP}$

,

F. Wonderful Impostors

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

You are a proud live streamer known as Gigi Murin. Today, you will play a game with n viewers numbered 1 to n.

In the game, each player is either a crewmate or an impostor. You don't know the role of each viewer.

There are m statements numbered 1 to m, which are either **true or false**. For each i from 1 to m, statement i is one of two types:

- $0\ a_i\ b_i$ $(1 \le a_i \le b_i \le n)$ — there are no impostors among viewers $a_i, a_i + 1, \ldots, b_i$;
- $1\ a_i\ b_i$ $(1 \le a_i \le b_i \le n)$ — there is **at least** one impostor among viewers $a_i, a_i + 1, \ldots, b_i$.

Answer q questions of the following form:

- $l\ r$ $(1 \le l \le r \le m)$ — is it possible that statements $l, l+1, \ldots, r$ are **all true**?

Note that it is **not guaranteed** that there is at least one impostor among all viewers, and it is **not guaranteed** that there is at least one crewmate among all viewers.

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 two integers n, m $(1 \le n, m \le 2 \cdot 10^5)$ — the number of viewers, and the number of statements.

The i-th of the next m lines contains three integers x_i, a_i, b_i $(x_i \in \{0, 1\}, 1 \le a_i \le b_i \le n)$ — describing statement i.

The next line contains a single integer q $(1 \le q \le 2 \cdot 10^5)$ — the number of questions.

Each of the next q lines contains two integers l and r $(1 \le l \le r \le m)$ — describing a question.

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

Output

For each question, output "YES" if it is possible that the requested statements are all true. 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.

Standard Input	Standard Output

4
4 3
1 1 3
	YES
	YES
	YES

```
1 2 4
0 2 3
1
1 3
5 2
0 1 5
1 1 5
3
1 1
2 2
1 2
1 2
0 1 1
1 1 1
2
1 1
2 2
7 9
1 2 2
1 4 5
0 5 6
1 2 2
1 1 1
0 4 7
0 3 7
0 2 7
0 6 6
5
1 5
2 6
3 7
4 8
5 9
```

```
NO
YES
YES
YES
NO
YES
NO
YES
```

Note

In the first test case, there are 4 viewers and 3 statements. The statements are as follows:

- Statement 1: There is at least one impostor among viewers 1, 2, and 3;
- Statement 2: There is at least one impostor among viewers 2, 3, and 4;
- Statement 3: There are no impostors among viewers 2 and 3.

We can see that it is possible that statements 1, 2, and 3 are all true. For example, this is one possible scenario:

- Viewer 1 is an impostor;
- Viewer 2 is a crewmate;
- Viewer 3 is a crewmate;
- Viewer 4 is an impostor.

In the second test case, there are 5 viewers and 2 statements. The statements are as follows:

- Statement 1: There is at least one impostor among viewers 1, 2, 3, 4, and 5;
- Statement 2: There are no impostors among viewers 1, 2, 3, 4, and 5.

We can see that it is possible that statement 1 is true, and it is possible that statement 2 is true. However, it is not possible that both statements 1 and 2 are true.

,

G. Wonderful Guessing Game

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

This is an interactive problem.

You are a proud teacher at the Millennium Science School. Today, a student named Alice challenges you to a guessing game.

Alice is thinking of an integer from 1 to n, and you must guess it by asking her some queries.

To make things harder, she says you must **ask all the queries first**, and she will **ignore** exactly 1 query.

For each query, you choose an array of k **distinct** integers from 1 to n, where k is even. Then, Alice will respond with one of the following:

- `L`: the number is one of the first $\frac{k}{2}$ elements of the array;
- `R`: the number is one of the last $\frac{k}{2}$ elements of the array;
- `N`: the number is not in the array;
- `?`: this query is ignored.

Alice is impatient, so you must find a strategy that **minimizes** the number of queries. Can you do it?

Formally, let $f(n)$ be the minimum number of queries required to determine Alice's number. Then you must find a strategy that uses **exactly** $f(n)$ queries.

Note that the interactor is **adaptive**, which means Alice's number is not fixed at the beginning and may depend on your queries. However, it is guaranteed that there exists at least one number that is consistent with Alice's responses.

We can show that $f(n) \leq 20$ for all n such that $2 \leq n \leq 2 \cdot 10^5$.

Input

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

The only line of each test case contains a single integer n ($2 \leq n \leq 2 \cdot 10^5$) — the maximum possible value of Alice's number.

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

Interaction

The interaction begins by reading the integer n.

Then, output a single integer q ($1 \leq q \leq 20$) — the number of queries.

To ask a query, output a line in the following format:

- $k\, a_1\, a_2 \ldots a_k$ ($2 \leq k \leq n$, k is even, $1 \leq a_i \leq n$, the a_i are distinct) — the length of the array, and the array itself.

Once you've asked all q queries, read a string s ($|s| = q$) — the responses to the queries as described above.

When you know Alice's number, output a single integer x ($1 \le x \le n$) — the value of the number.

Then, move on to the next test case, or terminate the program if there are no more test cases.

After outputting all q queries, do not forget to output the end of the line and flush the output. Otherwise, you will get `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.

Note that even if you correctly determine Alice's number but use more than $f(n)$ queries, you will get `Wrong answer`.

For this problem, hacks are disabled.

Standard Input	Standard Output
2	
3	2
	2 1 2
	2 1 2
?N	3
5	3
	4 3 2 4 1
	4 5 4 3 1
	4 1 5 3 4
R?L	1

Note

In the first test case, $n = 3$. We ask 2 queries: $[1, 2]$, and $[1, 2]$ again.

- For the first query, Alice's response is `?`, which means this query is ignored.
- For the second query, Alice's response is `N`, which means her number is not in the array $[1, 2]$.

From the information above, we can determine that Alice's number is 3.

It can be shown that all valid strategies for $n = 3$ require at least 2 queries.

In the second test case, $n = 5$. We ask 3 queries: $[3, 2, 4, 1]$, $[5, 4, 3, 1]$, and $[1, 5, 3, 4]$.

- For the first query, Alice's response is `R`, which means her number is in the array $[4, 1]$.
- For the second query, Alice's response is `?`, which means this query is ignored.
- For the third query, Alice's response is `L`, which means her number is in the array $[1, 5]$.

From the information above, we can determine that Alice's number is 1.

It can be shown that all valid strategies for $n = 5$ require at least 3 queries.

,

H. Wonderful XOR Problem

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

You are the proud... never mind, just solve this problem.

There are n intervals $[l_1, r_1], [l_2, r_2], \ldots [l_n, r_n]$. For each x from 0 to $2^m - 1$, find the number, modulo $998\,244\,353$, of sequences $a_1, a_2, \ldots a_n$ such that:

- $l_i \le a_i \le r_i$ for all i from 1 to n;
- $a_1 \oplus a_2 \oplus \ldots \oplus a_n = x$, where $\oplus$ denotes the bitwise XOR operator.

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 contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 18$).

The i-th of the next n lines contains two integers l_i and r_i ($0 \le l_i \le r_i < 2^m$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$, and the sum of 2^m over all test cases does not exceed 2^{18}.

Output

For each x from 0 to $2^m - 1$, let:

- f_x be the number of valid sequences, modulo $998\,244\,353$;
- $g_x = f_x \cdot 2^x \mod 998\,244\,353$.

Here, f_x and g_x are both integers in the interval $[0, 998\,244\,352]$.

Let $h = g_0 \oplus g_1 \oplus \ldots \oplus g_{2^m-1}$.

Output a single integer — the value of h itself. **Do not** perform a modulo operation.

Standard Input	Standard Output

4
2 2
0 2
1 3
5 3
3 7
1 3
0 2
1 5
3 6
10 14
314 1592
653 5897
	22
	9812
	75032210
	1073741823

932 3846
264 3383
279 5028
841 9716
939 9375
105 8209
749 4459
230 7816
1 5
0 29	

Note

For the first test case, the values of f_x are as follows:

- $f_0 = 2$, because there are 2 valid sequences: $[1, 1]$ and $[2, 2]$;
- $f_1 = 2$, because there are 2 valid sequences: $[0, 1]$ and $[2, 3]$;
- $f_2 = 2$, because there are 2 valid sequences: $[0, 2]$ and $[1, 3]$;
- $f_3 = 3$, because there are 3 valid sequences: $[0, 3]$, $[1, 2]$, and $[2, 1]$.

The values of g_x are as follows:

- $g_0 = f_0 \cdot 2^0 = 2 \cdot 2^0 = 2$;
- $g_1 = f_1 \cdot 2^1 = 2 \cdot 2^1 = 4$;
- $g_2 = f_2 \cdot 2^2 = 2 \cdot 2^2 = 8$;
- $g_3 = f_3 \cdot 2^3 = 3 \cdot 2^3 = 24$.

Thus, the value to output is $2 \oplus 4 \oplus 8 \oplus 24 = 22$.

For the second test case, the values of f_x are as follows:

- $f_0 = 120$;
- $f_1 = 120$;
- $f_2 = 119$;
- $f_3 = 118$;
- $f_4 = 105$;
- $f_5 = 105$;
- $f_6 = 106$;
- $f_7 = 107$.