Codeforces Round #737 (Div. 2)

A. Ezzat and Two Subsequences

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

Ezzat has an array of n integers **(maybe negative)**. He wants to split it into two **non-empty** subsequences a and b, such that every element from the array belongs to exactly one subsequence, and the value of $f(a) + f(b)$ is the maximum possible value, where $f(x)$ is the average of the subsequence x.

A sequence x is a subsequence of a sequence y if x can be obtained from y by deletion of several (possibly, zero or all) elements.

The average of a subsequence is the sum of the numbers of this subsequence divided by the size of the subsequence.

For example, the average of $[1, 5, 6]$ is $(1 + 5 + 6)/3 = 12/3 = 4$, so $f([1, 5, 6]) = 4$.

Input

The first line contains a single integer t ($1 \le t \le 10^3$)— the number of test cases. Each test case consists of two lines.

The first line contains a single integer n ($2 \le n \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print a single value — the maximum value that Ezzat can achieve.

Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1,|b|)} \le 10^{-6}$.

Example

input
4 3 3 1 2 3 -7 -6 -6 3 2 2 2 4 17 3 5 -3
output
4.500000000 -12.500000000 4.000000000 18.666666667

Note

In the first test case, the array is $[3, 1, 2]$. These are all the possible ways to split this array:

- $a = [3]$, $b = [1, 2]$, so the value of $f(a) + f(b) = 3 + 1.5 = 4.5$.
- $a = [3, 1]$, $b = [2]$, so the value of $f(a) + f(b) = 2 + 2 = 4$.
- $a = [3, 2]$, $b = [1]$, so the value of $f(a) + f(b) = 2.5 + 1 = 3.5$.

Therefore, the maximum possible value 4.5.
In the second test case, the array is $[-7, -6, -6]$. These are all the possible ways to split this array:

- $a = [-7]$, $b = [-6, -6]$, so the value of $f(a) + f(b) = (-7) + (-6) = -13$.
- $a = [-7, -6]$, $b = [-6]$, so the value of $f(a) + f(b) = (-6.5) + (-6) = -12.5$.

Therefore, the maximum possible value -12.5.

B. Moamen and k-subarrays

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

Moamen has an array of n **distinct** integers. He wants to sort that array in non-decreasing order by doing the following operations in order **exactly once**:

1. Split the array into exactly k non-empty subarrays such that each element belongs to exactly one subarray.
2. Reorder these subarrays arbitrary.
3. Merge the subarrays in their new order.

A sequence a is a subarray of a sequence b if a can be obtained from b by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

Can you tell Moamen if there is a way to sort the array in non-decreasing order using the operations written above?

Input

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

The first line of each test case contains two integers n and k ($1 \le k \le n \le 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le |a_i| \le 10^9$). It is guaranteed that all numbers are **distinct**.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, you should output a single string.

If Moamen can sort the array in non-decreasing order, output "YES" (without quotes). Otherwise, output "NO" (without quotes).

You can print each letter of "YES" and "NO" in any case (upper or lower).

Example

input
3 5 4 6 3 4 2 1 4 2 1 -4 0 -2 5 1 1 2 3 4 5

output
Yes No Yes

Note
In the first test case, $a = [6, 3, 4, 2, 1]$, and $k = 4$, so we can do the operations as follows:

1. Split a into $\{[6], [3, 4], [2], [1]\}$.
2. Reorder them: $\{[1], [2], [3, 4], [6]\}$.
3. Merge them: $[1, 2, 3, 4, 6]$, so now the array is sorted.

In the second test case, there is no way to sort the array by splitting it into only 2 subarrays.

As an example, if we split it into $\{[1, -4], [0, -2]\}$, we can reorder them into $\{[1, -4], [0, -2]\}$ or $\{[0, -2], [1, -4]\}$. However, after merging the subarrays, it is impossible to get a sorted array.

C. Moamen and XOR

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

Moamen and Ezzat are playing a game. They create an array a of n non-negative integers where every element is less than 2^k.

Moamen wins if $a_1 \,\&\, a_2 \,\&\, a_3 \,\&\, \ldots \,\&\, a_n \geq a_1 \oplus a_2 \oplus a_3 \oplus \ldots \oplus a_n$.

Here $\&$ denotes the bitwise AND operation, and $\oplus$ denotes the bitwise XOR operation.

Please calculate the number of winning for Moamen arrays a.

As the result may be very large, print the value modulo $1\,000\,000\,007$ ($10^9 + 7$).

Input
The first line contains a single integer t ($1 \leq t \leq 5$)— the number of test cases.

Each test case consists of one line containing two integers n and k ($1 \leq n \leq 2 \cdot 10^5$, $0 \leq k \leq 2 \cdot 10^5$).

Output
For each test case, print a single value — the number of different arrays that Moamen wins with.

Print the result modulo $1\,000\,000\,007$ ($10^9 + 7$).

Example

input
3 3 1 2 1 4 0

output
5 2 1

Note
In the first example, $n = 3$, $k = 1$. As a result, all the possible arrays are $[0, 0, 0]$, $[0, 0, 1]$, $[0, 1, 0]$, $[1, 0, 0]$, $[1, 1, 0]$, $[0, 1, 1]$, $[1, 0, 1]$, and $[1, 1, 1]$.

Moamen wins in only 5 of them: $[0, 0, 0]$, $[1, 1, 0]$, $[0, 1, 1]$, $[1, 0, 1]$, and $[1, 1, 1]$.

D. Ezzat and Grid

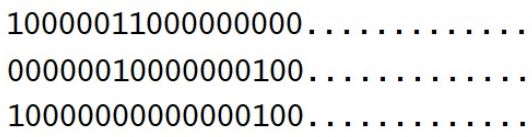
time limit per test: 2.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Moamen was drawing a grid of n rows and 10^9 columns containing only digits 0 and 1. Ezzat noticed what Moamen was drawing and became interested in the minimum number of rows one needs to remove to make the grid beautiful.

A grid is beautiful if and only if for every two consecutive rows there is at least one column containing 1 in these two rows.

Ezzat will give you the number of rows n, and m segments of the grid that contain digits 1. Every segment is represented with three integers i, l, and r, where i represents the row number, and l and r represent the first and the last column of the segment in that row.

For example, if $n = 3$, $m = 6$, and the segments are $(1, 1, 1)$, $(1, 7, 8)$, $(2, 7, 7)$, $(2, 15, 15)$, $(3, 1, 1)$, $(3, 15, 15)$, then the grid is:



Your task is to tell Ezzat the minimum number of rows that should be removed to make the grid beautiful.

Input
The first line contains two integers n and m ($1 \leq n, m \leq 3 \cdot 10^5$).

Each of the next m lines contains three integers i, l, and r ($1 \leq i \leq n$, $1 \leq l \leq r \leq 10^9$). Each of these m lines means that row number i contains digits 1 in columns from l to r, inclusive.

Note that the segments **may overlap**.

Output
In the first line, print a single integer k — the minimum number of rows that should be removed.

In the second line print k distinct integers $r_1, r_2, \ldots, r_k$, representing the rows that should be removed ($1 \leq r_i \leq n$), in any order.

If there are multiple answers, print any.

Examples

input
3 6 1 1 1 1 7 8 2 7 7 2 15 15 3 1 1 3 15 15
output
0

input
5 4 1 2 3 2 4 6 3 3 5 5 1 1
output
3 2 4 5

Note

In the first test case, the grid is the one explained in the problem statement. The grid has the following properties:

1. The 1-st row and the 2-nd row have a common 1 in the column 7.
2. The 2-nd row and the 3-rd row have a common 1 in the column 15.

As a result, this grid is beautiful and we do not need to remove any row.
In the second test case, the given grid is as follows:

$$01100000\ldots\ldots\ldots$$

$$00011100\ldots\ldots\ldots$$

$$00111000\ldots\ldots\ldots$$

$$00000000\ldots\ldots\ldots$$

$$10000000\ldots\ldots\ldots$$

E. Assiut Chess

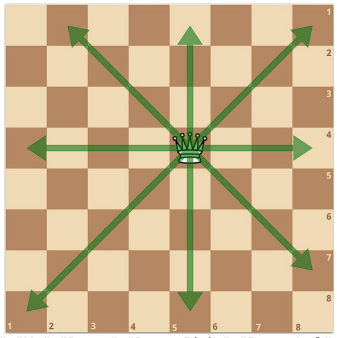
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is an interactive problem.

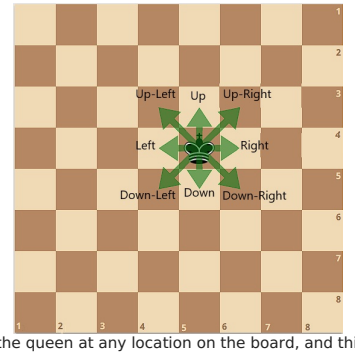
ICPC Assiut Community decided to hold a unique chess contest, and you were chosen to control a queen and hunt down the hidden king, while a member of ICPC Assiut Community controls this king.

You compete on an 8×8 chessboard, the rows are numerated from top to bottom, and the columns are numerated left to right, and the cell in row x and column y is denoted as (x, y).

In one turn you can move the queen to any of the squares on the same horizontal line, vertical line, or any of the diagonals. For example, if the queen was on square $(4, 5)$, you can move to $(q_1, 5)$, $(4, q_1)$, $(q_1, 9 - q_1)$, or $(q_2, q_2 + 1)$ where ($1 \leq q_1 \leq 8$, $q_1 \neq 4$, $1 \leq q_2 \leq 7$, $q_2 \neq 4$). Note that the queen **cannot** stay on its current cell.



In one turn, the king can move "Right", "Left", "Up", "Down", "Down-Right", "Down-Left", "Up-Left", or "Up-Right" such that he doesn't get out of the board. The king **cannot** move into a cell that is on the same row, column or diagonal with the queen (including the position of the queen itself). For example, if the king was on square $(4, 5)$, he can move to $(4 + k_1, 5 + k_2)$ where ($-1 \leq k_1, k_2 \leq 1$, $(k_1, k_2) \neq (0, 0)$).



At the start of the game, you should place the queen at any location on the board, and this is done once per game. After that the king is secretly placed at any cell different from the queen's location. You do not know the position of the king. Then, the king and the queen take turns with the king moving first. The king moves to one of the possible directions ("Right", "Down", "Up-Left", etc.),

and you are only given the direction it moves to. After that, you should move your queen by declaring the square to which your queen will move. The game follows like this until you win the game or run out of moves.

You win if the king has no valid moves. You lose if after 130 moves of the queen the king still has valid moves.

Input
The first line contains a single integer t ($1 \le t \le 60$) — the number of test cases.

Interaction
In each test case, you should print the queen's starting cell immediately. If you placed the queen at the king's cell, you will win immediately.

After that, you may make at most 130 moves. Each move is made in the format x y, where x and y are two integers ($1 \le x, y \le 8$) that denote the new row and column of the queen respectively. Your move should be a valid queen move.

After the initial queen placement and after each move you will receive a string s that represents the direction of the king's move. It will be one of the following: `"Right"`, `"Left"`, `"Up"`, `"Down"`, `"Down-Right"`, `"Down-Left"`, `"Up-Left"`, `"Up-Right"`, or `"Done"` if you win the game. You should consider `"Done"` as the end of each test case.

After printing a query do not forget to output end of 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 documentation for other languages.

If at any point you make an invalid query or try to make more than 130 queries for each test case, the game will terminate immediately and you will receive a Wrong Answer verdict.

Example

input
1 Left Right Done
output
7 5 7 6 7 7

Note
In the example, the hidden king was at $(8, 8)$ at the start. The game follows like this: