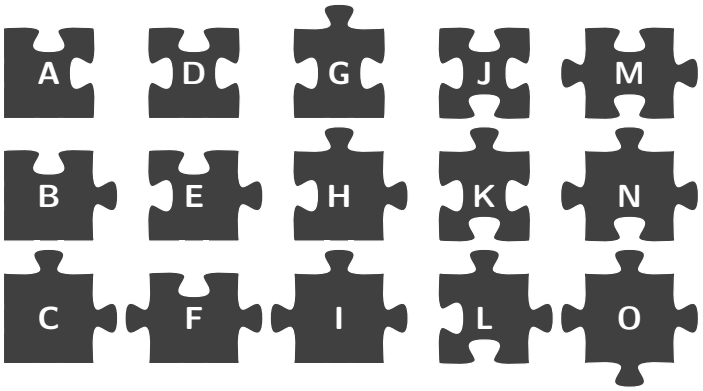
Problem A. Missing Piece

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

You are solving a blank 3×3 wooden jigsaw puzzle. But after solving the outside it turns out that the center piece is missing! As the puzzle is blank and its connections are all the same you can freely rotate and flip the pieces, so there are only 15 possible different pieces, labeled by the letters from A to O:



You want to make a new piece to finish the puzzle. Given your solution of the outside of the puzzle, find which piece is missing.

Input

The first row of input consists of a single integer t ($1 \leq t \leq 4096$), the number of testcases. Then for each case the input is three lines of three symbols each. The center symbol will be '`?`', denoting the missing piece. The other eight symbols will be uppercase English letters from '`A`' to '`O`', denoting a solution to the outside of the puzzle. It is guaranteed that they form a valid solution for the outside of a 3×3 puzzle.

Output

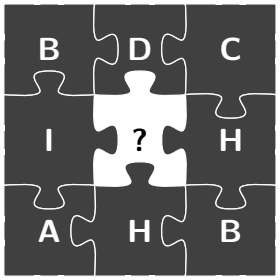
For each testcase output a single line with a single uppercase English alphabet letter from '`A`' to '`O`', the label of the piece missing from your puzzle.

Example

standard input	standard output
1 BDC I?H AHB	K

Note

The sample corresponds to this puzzle:



Problem B. Midpoint Chain

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

You are given $n \geq 3$ points $M_1, \ldots, M_n$ with integer coordinates in the 2-dimensional plane. Find any n points $A_1, \ldots, A_n$ with integer coordinates such that M_i is the midpoint of the segment (A_i, A_{i+1}), with M_n being the midpoint of (A_n, A_1), or determine that this is impossible. Note that $M_1, \ldots, M_n$ are not necessarily distinct, and $A_1, \ldots, A_n$ are also not required to be.

Input

The first line contains a single integer n ($3 \leq n \leq 10^5$), the number of midpoints. The i-th of the next n lines describes M_i by its integer coordinates x_i and y_i ($-10^9 \leq x_i, y_i \leq 10^9$).

Output

If the required is impossible, simply output "`No`". Otherwise, in the first line output "`Yes`", followed by n lines, where the i-th line describes A_i in the same format as M_i. The absolute value of each coordinate must not exceed 10^{18}. If there are multiple solutions, output any of those.

Examples

standard input	standard output
4 0 0 0 1 1 1 1 0	Yes 1 1 -1 -1 1 3 1 -1
6 -10 5 6 3 9 -7 9 -3 -10 5 9 4	No

Problem C. Reverse Scrabble

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

You've gotten bored with the classic Scrabble, so you've bought yourself a new single-player board game, "Reverse Scrabble". In this game, instead of placing the letter tiles on the board, you have to place the bonus tiles on your word to maximize the score.

More specifically, the game has n letter tiles having scores $a_1, \ldots, a_n$. Unlike the original Scrabble, these can be any positive integers. In the beginning of the game, all the letters are placed in a line to form some word (the letters and their order are not important for this problem, only their scores). Without any bonus tiles, the score of the word is simply the sum of the tile scores.

As in regular Scrabble, there are also 4 types of bonus tiles. However, unlike regular Scrabble, you have to place these bonus tiles, at most one on each letter. The bonuses have the following effects. "Double Letter" and "Triple Letter" multiply the score of the corresponding letter by 2 and 3, respectively. "Double Word" and "Triple Word" multiply the total score of the word by 2 and 3, respectively. The "Letter" bonuses are applied before the "Word" bonuses. If there are multiple "Word" bonuses, all of them are applied.

For example, if the word consists of 6 tiles with the scores 2, 6, 1, 3, 5 and 1, a "Triple Letter" is placed on the second letter, a "Double Letter" on the fifth letter and two "Double Word" tiles are placed on two of the other letters, then the final total score is equal to $(2 + 3 \cdot 6 + 1 + 3 + 2 \cdot 5 + 1) \cdot 2 \cdot 2 = 140$. This is also an optimal placement of these bonuses that maximizes the score.

In a single round of the game, you draw a set of bonus tiles. Determine an optimal placement of the bonuses so that the final total score is maximized!

Input

The first line contains a single integer n ($1 \le n \le 10^5$), the number of letter tiles. The next line contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^9$), the scores of the letter tiles. The last line contains four integers dl, tl, dw and tw ($0 \le dl, tl, dw, tw \le n$), the numbers of the "Double Letter", "Triple Letter", "Double Word" and "Triple Word" bonus tiles, respectively.

Output

Output n strings of length 2, where each describes the type of bonus to be placed on each letter. The strings "`DL`", "`TL`", "`DW`", "`TW`" denote the "Double Letter", "Triple Letter", "Double Word" and "Triple Word" bonuses, respectively. The string "`--`" means that no bonus should be placed on the corresponding letter. You can apply any number of bonuses, but not more than are given in the input. If there are multiple solutions, output any of those.

Examples

standard input	standard output
6 2 6 1 3 5 1 1 1 2 0	-- TL -- DW DL DW
5 347458 819237 982137 129873 459068 4 3 1 3	DW TW TL TW TW

Problem D. Motocross

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

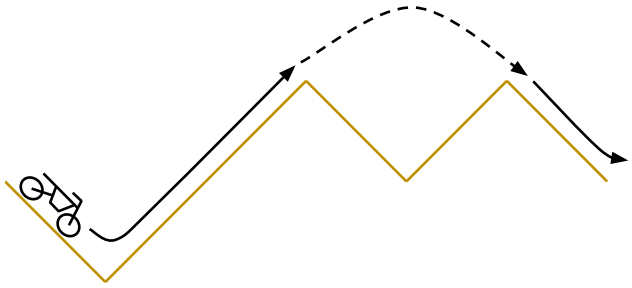
You are planning your route through the last obstacle segment of a new motocross track. As the segment lies just before the starting straight, you want to maximize your speed when exiting the obstacle.

The obstacle consists of n sections, where each section can be either an upslope or a downslope. You start in the first section, having entered it with a speed of v. The obstacle always ends with a downslope.

If you are currently on a downslope, or an upslope with the next section also being an upslope, you drive to the next section (or the straight if you are at the last section). During this you may accelerate, adding 1 to your speed.

However, if you are currently on an upslope and the next section is a downslope, you jump. Before jumping you may brake, setting your speed to any integer between 1 and your current speed. Then you jump forward as many segments as your remaining speed. If you land on a downslope you maintain this speed. If you land on an upslope you slam into it and nearly stop, setting your speed to 1.

Driving optimally, what is the maximum speed you can enter the straight with? You may enter the straight while airborne.



Input

The first line of input contains two integers n and v ($1 \le n, v \le 3 \cdot 10^5$), the number of sections and your starting speed, respectively.

The second line consists of an n character string, where the i-th character is '\' if the i-th section is a downslope and '/' if it is an upslope. It is guaranteed that the last character is '\'.

Output

Output a single integer, the maximum speed with which you can exit the obstacle.

Examples

standard input	standard output
6 1 \//\/\	4
2 100 /\	100

Problem E. Sensor Array

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

You are calibrating the cutting edge detector for a new physical field. The detector consists of an array of sensors arranged in a regular $n \times m$ grid. Each sensor reports a single positive integer value. It is known that the field fluctuates so that orthogonally adjacent sensors always report values which differ exactly by one. Further, the detector only measures differences from a baseline reading, so at least one sensor always reports a value of 1.

You would like to know how many possible outputs the detector could report under the known constraints.

Input

The input consists of two positive integers n, m ($1 \leq n \leq 10^9$, $1 \leq m < 10$), the dimensions of the grid.

Output

Output a single integer, the number of possible outputs modulo $10^9 + 7$.

Examples

standard input	standard output
2 6	486
2 2	6

Problem F. Pills

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.25 seconds
Memory limit:	256 megabytes

You are old and need to take k different pills every day. You forgot k, but luckily you have written down the sequence of pills you have taken for the last n days. Each pill is denoted by a letter.

You know that each day you took the correct k pills, but the order could be different each day. Your task is to find out the value of k!

Input

The first line of input contains a single integer $n \cdot k$ ($1 \leq n \cdot k \leq 10^5$) – the length of the pill sequence you have taken over the last n days. The second line contains $n \cdot k$ lower case English alphabet letters describing the pill sequence, where each letter represents a different kind of pill.

Output

Output a single integer, the value of k.

Example

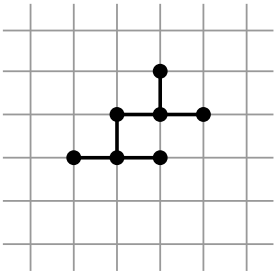
standard input	standard output
12 ipsuusipspiu	4

Problem G. Binary Tree

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

Given a positive integer n, you need to draw a perfect binary tree of depth n on a regular square grid or determine that it is impossible. The tree must be drawn in such a way that:

- each vertex is drawn as a dot at a line intersection of the grid;
- each edge is drawn on a single side of a cell of the grid;
- no two vertices are located in the same line intersection of the grid.



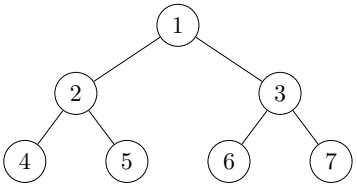
Input

The input contains a single integer n ($1 \le n \le 16$), the depth of the tree.

Output

If the required is impossible, simply output "`No`". Otherwise, output "`Yes`" in the first line, and the description of the tree in the next $2^{n+1} - 1$ lines as follows.

First, enumerate the $2^{n+1} - 1$ vertices of the tree with integers from 1 to $2^{n+1} - 1$. The number 1 is assigned to the root vertex, and for each non-leaf vertex, if it is assigned the number v, then its children are numbered by $2v$ and $2v + 1$, as shown below.



Then, the i-th line should contain the coordinates (x_i, y_i) of the i-th vertex of the tree ($-10^9 \le x_i, y_i \le 10^9$). If there are multiple solutions, output any of those.

Example

standard input	standard output
2	Yes 0 0 1 0 0 -1 1 1 2 0 -1 -1 1 -1

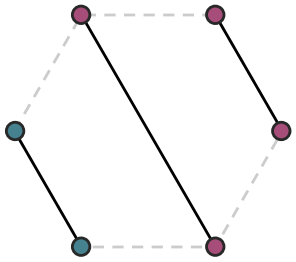
Note

A *tree* is a connected graph that has no cycles. A rooted tree is a tree with one vertex being the *root vertex*. In a rooted tree, a vertex u is a *child* of v if and only if there is an edge between v and u, and u does not belong to the path that connects the root vertex with v. A vertex of a rooted tree is called a *leaf* if and only if it has no children. A *binary* tree is a tree where each vertex has either 0 or 2 children. The *perfect* binary tree of *depth* n is a binary tree which has exactly 2^n leaves that all are at distance n from the root.

Problem H. Colored Polygon

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

There are n points on a plane forming a regular n-gon, where n is even. The points are numbered clockwise from 1 to n. Each point is colored in one of k colors numbered by the integers from 1 to k. Two points can be connected with a straight segment if and only if they have the same color. You need to determine whether it is possible to connect $n/2$ pairs of points by segments so that no two segments have any common points. If it is possible, you need to construct any such configuration.



Input

The first line of input contains two integers n and k ($4 \le n \le 5 \cdot 10^5$, $n \equiv 0 \pmod 2$, $1 \le k \le n$), the number of points and the number of different colors, respectively. The next line contains n integers $c_1, \ldots, c_n$ ($1 \le c_i \le k$), where c_i denotes the color of the i-th point. It is guaranteed that for each color between 1 and k, there exists at least one point with such color.

Output

If the required is impossible, simply output "`No`". Otherwise, in the first line output "`Yes`", followed by $n/2$ lines. Each of these lines should contain the indices of a pair of points connected by a segment. You can output the segments in any order; if there are multiple solutions, output any of those.

Examples

standard input	standard output
6 2 1 1 1 2 2 1	Yes 1 2 4 5 6 3
4 2 2 1 2 1	No

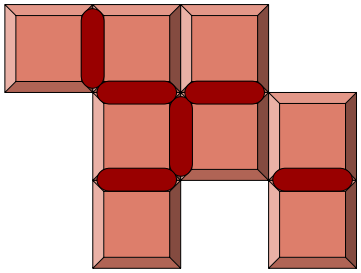
Problem I. Backrests

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

After moving to a bigger apartment, you want to design a custom couch for it. You've settled on the layout of the couch; all that's left is to place the backrests.

The couch consists of n identical square cushions connected orthogonally in a grid. Each cushion has at least one side not connected to any other cushion. Each cushion needs a backrest so that people can kick back and relax – a backrest is a segment of length 1 along the side of a cushion. For proper legroom, a functional backrest has to be facing an open side of the cushion.

If two cushions share a side, a single backrest can be used for both cushions. As the couch is already quite expensive, you want to minimize your spending on backrests. Find the minimal number of backrests so that every cushion has a functional one.



Input

The first line of input has a single integer n ($1 \le n \le 10^5$), the number of cushions forming the couch. The next n lines contain the positions of the cushions. The i-th line contains two integers x_i and y_i ($0 \le x_i, y_i \le 10^5$), the coordinates of the i-th cushion. The cushions may be given in any order. No two cushions share the same coordinates. It is guaranteed that the cushions form a single orthogonally connected couch and that each cushion has at least one open side.

Output

Output a single integer, the minimal number of backrests needed so that each cushion has a functional backrest.

Examples

standard input	standard output
3 0 0 1 0 2 0	3
8 0 2 1 2 1 1 1 0 2 1 3 1 3 0 2 2	6

Problem J. Island Counting

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3.5 seconds
Memory limit:	256 megabytes

You are surveying an uncharted chain of islands using a mapping satellite. The satellite scans the terrain in an unpredictable pattern, producing a series of n axis-aligned rectangles on a grid fully covering all land in the region. Any territory not covered by any of the rectangles is water.

You want to analyze this data and count the number of islands in the chain. Any overlapping or touching rectangles form a single island, even a touching corner means a connecting isthmus of land. Additionally, you are interested in finding large freshwater reservoirs, and so want to count the number of islands with lakes – an island has a lake if it fully encloses some water.

Input

The first line of input contains a single integer n ($1 \le n \le 2 \cdot 10^5$), the number of rectangles of land. The following n lines contain a description of the rectangles. The i-th line contains four integers $x_{s,i}, y_{s,i}, x_{e,i}, y_{e,i}$ ($-10^9 \le x_{s,i} \le x_{e,i} \le 10^9$, $-10^9 \le y_{s,i} \le y_{e,i} \le 10^9$), the description of the i-th rectangle, which corresponds to the smallest by area axis-aligned rectangle that fully contains the grid cells at positions $(x_{s,i}, y_{s,i})$ and $(x_{e,i}, y_{e,i})$.

Output

Output two integers: the total number of islands and the number of islands containing at least one lake.

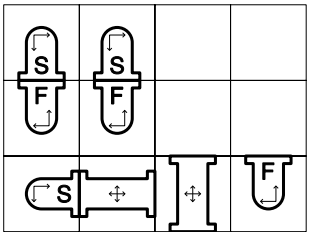
Examples

standard input	standard output
6 4 0 4 3 0 0 4 0 6 10 37 41 0 4 3 4 11 15 23 22 0 0 0 4	2 1
10 -10 -10 10 -10 -10 -10 -10 10 -10 10 10 10 10 -10 10 10 -6 -6 6 -6 -6 -6 -6 6 -6 6 6 6 6 -6 6 6 0 -6 0 6 3 3 3 3	3 2

Problem K. Pipe Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.25 seconds
Memory limit:	256 megabytes

In the pipe game you are given a grid where each cell contains a piece of pipe which can be rotated in 90° increments, the goal is to rotate all the pieces so that there are no unconnected ends of pipe.



In this task you have to solve a simplified version of the game. Each piece can only be in one of two states and there are only 3 types of pieces:

- straight pipe – can be either horizontal or vertical; in the input these are denoted by '`|`';
- start piece – can only point down or to the right; in the input denoted by '`S`';
- end piece – can only point up or to the left; in the input denoted by '`F`'.

Input

The first line contains two integers w and h ($1 \leq w, h \leq 500$), the width and height of the game field. The following h lines contain a description of each row, consisting of the characters '`S`', '`F`', '`|`' and '`.`'.

Output

If there is no solution, simply output "`No`". Otherwise, output "`Yes`" in the first line, and a grid containing a solution in the following h lines, using these symbols:

- '`.`' – empty cell;
- '`R`' – pipe pointing right;
- '`D`' – pipe pointing down;
- '`L`' – pipe pointing left;
- '`U`' – pipe pointing up;
- '`|`' – vertical pipe;
- '`-`' – horizontal pipe.

If there are multiple solutions, output any of those.

Example

standard input	standard output
4 3 SS.. FF.. S||F	Yes DD.. UU.. R--L

Problem L. Mech Control

Input file:	`standard input`
Output file:	`standard output`
Time limit:	0.35 seconds
Memory limit:	256 megabytes

You are learning to pilot your new giant combat mech, the last hope of Neo Riga – an essentially infinite square grid of compass-aligned city blocks. For reasons you slept through the explanation of, the mech has to be controlled by walking around its control room.

The control room is an $n \times m$ grid of panels with the entrance at the top left corner, where you are currently located. Each panel depicts an arrow pointing up, right, down or left, respectively corresponding to the compass directions north, east, south and west.

In a single move, you may step from your current location to any orthogonally adjacent panel. The mech then moves one city block in the direction corresponding to the entered panel. However, the panel then changes to the opposite symbol (left to right or up to down and vice versa).

For this training exercise, you want to move the mech to the point exactly $n - 1$ blocks west and $m - 1$ blocks north of its current location before the mech runs out of power after 10^5 moves. Find a series of moves to accomplish this!

Input

The first line of input contains two integers n and m ($1 \le n, m \le 500$), the dimensions of the control room. It is guaranteed that $n \cdot m > 1$. The next m lines each consists of n characters '`<`', '`>`', '`^`' and '`v`', describing the initial state of the control room.

Output

If the required is impossible, simply output "`No`". Otherwise output "`Yes`" in the first line and a string with the list of moves on the second line not exceeding 10^5 in length. Use the characters '`L`', '`R`', '`U`' and '`D`' to denote a move one panel to the left, right, up or down, respectively.

Example

standard input	standard output
3 2 >^< v<>	Yes RRDLR