

Problem A. Area

In mathematics, the *Kronecker product*, denoted by $\otimes$, is an operation on two matrices of arbitrary size resulting in a block matrix. Given matrices A and B of sizes $k \times l$ and $m \times n$, respectively, their Kronecker product $A \otimes B$ is a $km \times ln$ matrix

$$\begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1l}B \\ a_{21}B & a_{22}B & \dots & a_{2l}B \\ \dots & \dots & \dots & \dots \\ a_{k1}B & a_{k2}B & \dots & a_{kl}B \end{pmatrix}$$

where each block $a_{ij}B$ expands to an $m \times n$ submatrix

$$\begin{pmatrix} a_{ij}b_{11} & a_{ij}b_{12} & \dots & a_{ij}b_{1n} \\ a_{ij}b_{21} & a_{ij}b_{22} & \dots & a_{ij}b_{2n} \\ \dots & \dots & \dots & \dots \\ a_{ij}b_{m1} & a_{ij}b_{m2} & \dots & a_{ij}b_{mn} \end{pmatrix}$$

The Kronecker product is also called *direct product* of matrices.

Consider matrix $A = \begin{pmatrix} +1 & +1 \\ +1 & -1 \end{pmatrix}$. Let us denote $A^{(2)} = A \otimes A$, $A^{(3)} = A^{(2)} \otimes A$, $\dots$, $A^{(p)} = A^{(p-1)} \otimes A$. Finally, we define $A^{(\infty)}$ as an infinite two-dimensional table that can be obtained if we continue the direct multiplication process ad infinitum. Clearly, it contains only plus and minus ones.

Pick a cell (x, y) in $A^{(\infty)}$; the upper left cell has coordinates $(0, 0)$. Let the *territory associated with* (x, y) be the set of cells of the table which are reachable from (x, y) by traversing a path in which every two adjacent cells contain equal numbers and share a common side.

Given the cell coordinates, find the area of the territory associated with it, i.e., the number of cells in that territory. Note that this area is not necessarily finite.

Input

The first line of the input file contains two integer numbers x and y ($0 \leq x, y \leq 10^9$) separated by a single space.

Output

If the area of the territory associated with (x, y) is finite, print it at the first line of the output file. Otherwise, print the word "infinity" instead.

Examples

standard input	standard output
2 3	5
1 4	infinity

Problem B. Fill the Triangle

You are given an equilateral triangle drawn on an infinite hexagonal grid. The vertices of the triangle are in the centers of the grid cells and side of the triangle has length of n units (where one unit is equal to length of any side of any cell).

Some of cells are white and some of cells are black. As one can see, the triangle intersects exactly $n \cdot (n + 1) / 2$ hexagonal cells. You are allowed to simultaneously repaint to opposite color any three cells which have a common vertex (white cells become black and vice versa). It is allowed to do such operations only with vertices that lie inside triangle.

Your task is to determine whether it's possible to fill the entire triangle by black color.

Input

The input consists of one or more test cases. The first line of the case contains an integer n ($1 \leq n \leq 1000$). The i -th of the next n lines contains $n - i + 1$ characters '0' or '1', where zeroes correspond to black color and ones to white color.

The input is terminated by a case with $n = 0$. This case should not be processed.

Output

Adhere to the sample output below.

Example

standard input	standard output
3	Triangle 1 can be
100	filled.
01	Triangle 2 cannot be
0	filled.
2	
10	
1	
0	

Problem C. Gary

Gary is a Beer Robot at the Rocket Engine Bar orbiting Alpha Herculis II. His job is to accept bank notes and coins from the customers, and offer beer in return. Gary has one slot for bank notes and another slot for coins. He has no holes to give back change.

Gary is an old and experienced bar-robot. Pirates from nearby planetary systems like to listen to his stories about the good old times when a simple railgun was the heaviest spacecraft armament in existence, and boarding was a fast and effective way to capture a tradeship.

However, being an old worn-out robot has certain drawbacks. Perhaps the most nasty one is that he can accept only one type of coins and one type of bank notes. And Gary is pretty much aware that, for example, one cannot pay exactly 5 credits using only 4-credit bank notes and 3-credit coins.

Fortunately, Gary's self-repair equipment allows him to perform some fine tuning and re-pick the denominations of the coins and bank notes his slots would accept. Now he wants to cleverly choose the numbers a and b which would be the denominations of these bank notes and coins. More specifically, for certain a and b he wants to estimate how good or bad this choice would be. And for that, he needs to know how many positive integer amounts of credits q could not be paid using only a -credit bank notes and b -credit coins, and the greatest integer amount of credits l that could not be paid.

Input

On the first line of the input file there are two integer numbers a and b separated by a single space ($1 \leq a, b \leq 10^9$).

Output

You should print two numbers, q and l , on the first line of the output file. Numbers should be separated by a single space. If q or l is infinite, print "infinity" instead of the respective number.

Examples

standard input	standard output
3 4	3 5
2 2	infinity infinity

Problem D. K-th sequence

Consider the set S of all non-negative integer sequences of length n . Sort all sequences in S by sum of their components (sequences with the same sums are ordered lexicographically), thus obtaining some sorted list of sequences T . Your task is to find k -th sequence in T (numbering is 0-based).

Input

Input consists of one or more test cases. Each case is given on a separate line and consists of two numbers, n and k ($1 \leq n \leq 1000$, $0 \leq k \leq 10^{18}$). Input is terminated by a line containing two zeroes; this line should not be processed.

Output

For each test case, output the corresponding sequence. Adhere to the output format shown below as close as possible.

Example

standard input	standard output
2 2	Case 1: 1 0
3 6	Case 2: 0 2 0
0 0	

Problem E. Longest Increasing Subsequences

Consider a permutation $a_1, a_2, \dots, a_n$ of integers 1 through n . An *increasing subsequence* of a of length p is a sequence $a_{k_1}, a_{k_2}, \dots, a_{k_p}$ where $1 \leq k_1 < k_2 < \dots < k_p \leq n$ and $a_{k_1} < a_{k_2} < \dots < a_{k_p}$.

Let l be the length of the longest increasing subsequence of a . Now, we want to know how many different increasing subsequences of length l one can find in a . Subsequences $a_{k_1}, a_{k_2}, \dots, a_{k_l}$ and $a_{j_1}, a_{j_2}, \dots, a_{j_l}$ are considered different if $k_i \neq j_i$ for some index i . Since the number of different longest increasing subsequences can be quite large, you should calculate it modulo m .

Input

On the first line of the input file, there are two integers n and m separated by a single space ($1 \leq n \leq 100\,000$, $1 \leq m \leq 10^9$). On the second line, there are n integer numbers $a_1, a_2, \dots, a_n$ separated by single spaces. These n numbers are a permutation of $1, 2, \dots, n$.

Output

On the first line of the output file, print a single integer — the number of longest increasing subsequences in a modulo m .

Examples

standard input	standard output
3 8 1 3 2	2
4 1000000000 2 4 1 3	3

Problem F. Polyomino Packing

Two polyominoes lie on an infinite grid, with sides of their cells

going along the grid lines. The polyominoes can be moved in such a way that their sides remain parallel to the grid lines, and at any moment there is no area where they overlap; however, it is possible for the polyominoes to touch each other. After the movement is finished, we calculate the area of the smallest rectangle with sides parallel to grid lines which contains both polyominoes.

In this problem, our goal is to find the minimal area of such rectangle which can be achieved after moving the polyominoes.

Input

On the first line of the input file, there are two integers n and m separated by a single space — the size of the initial rectangular area ($1 \leq n, m \leq 200$). Next n lines contain m characters each. Here, '.' stands for empty space, 'A' for the cells of the first polyomino and 'B' for the cells of the second. It is guaranteed that each of the two polyominoes is connected by side. There is at least one 'A' and at least one 'B' in the input file.

Output

On the first line of the output file, print one number — the minimal area of a rectangular bounding box which can contain both polyominoes.

Examples

standard input	standard output
3 5 AA... A..BB A...B	9
4 4 AA.. .ABB AAB. ..BB	12

Problem G. Non-Attacking Queens

Find the number of ways to place three chess queens on an $n \times n$ chessboard such that they do not attack each other.

Input

The first line of the input file contains a single integer n ($1 \leq n \leq 100\,000$).

Output

Print the number of ways to place three non-attacking queens on the first line of the output file.

Examples

standard input	standard output
4	24
5	204

Problem H. Vuglusr and Mouse

There are Vuglusr and Mouse somewhere inside the convex polygonal arena. Vuglusr's goal is to catch Mouse and do something untoward to it, and Mouse's goal is to escape the arena before such a calamity happens.

Vuglusr can move with any speed not greater than v_1 , and Mouse can flee with any speed not greater than v_2 . It is not necessary to move along a straight line.

Your task is to determine the outcome given that both sides act optimally. Both sides are allowed to change their strategies at any time depending on opponent's moves.

If Vuglusr and Mouse will reach the arena boundary simultaneously in same point, it is considered a draw.

Input

Input consists of one or more test cases. The first line of each case contains a single integer n — the number of vertices in the polygon ($1 \leq n \leq 100$). n lines follow, each containing coordinates of a vertex x_i, y_i , given in counter-clockwise order. The case is terminated by two lines, first specifying Vuglusr's initial position and maximal speed, second specifying Mouse's. The positions are different and lie strictly inside the given polygon. All numbers in the input file are integers which do not exceed 10000 by absolute value. Input is terminated by the line containing a single zero; this line should not be processed.

Output

Adhere as close as possible to the output format shown below. If Mouse can reach the boundary, output any point where it can happen. Print real numbers as precisely as possible.

Example

standard input
4
10 -10
10 10
-10 10
-10 -10
0 0 2
5 0 1
4
10 -10
10 10
-10 10
-10 -10
0 -1 13
0 1 12
4
10 -10
10 10
-10 10
-10 -10
0 1 10
0 2 8
0
standard output
Arena 1: There can be draw at (10.000000000, 0.000000000).
Arena 2: Mouse can escape at (10.000000000, 10.000000000).
Arena 3: Something bad will happen!

Problem I. Fleet Vulnerability

A large imperial fleet has entered the pseudospace around Alpha Herculis III which currently serves as Alpha Herculis pirates' meeting point and base for operations. Since a ship must enter pseudospace to make a hyperjump, this basically means that every pirate vessel trying to escape Alpha Herculis planetary system will be intercepted and captured. And no doubt that the Imperial Marine Corps is already on its way from Alpha Herculis II. As the

captain of a vessel in orbit of the pirate planet, you should find a way to escape. Luckily, your aide have come up with a brilliant idea.

The pseudospace around a planet can be represented as a 2-dimensional plane, spacecrafts being points on that plane; different spacecrafts occupy different points on the plane. When a fleet of 4 or more spacecrafts enters the same pseudospace, it may become vulnerable. Consider the set S of spacecrafts, and the set $T(S)$ of triangles with vertices in S ; here, a triangle is just a set of three spacecrafts from S . Then, the *vulnerability* of S is just the number of pairs of equal triangles from $T(S)$. Two triangles are considered equal if, given the sides $a_1 \leq b_1 \leq c_1$ of the first triangle and $a_2 \leq b_2 \leq c_2$ of the second, we have $a_1 = a_2, b_1 = b_2$ and $c_1 = c_2$.

Fortunately, your chief tech has recently installed a distortion generator on board. The generator is a weapon which can be effectively used against large fleets if their captains don't pay attention to vulnerability concerns — it distorts the pseudospace around itself, making it harder to reach the user. Fleets with high vulnerability can even take damage from the distortion.

So, one way to escape would be to pin out to pseudospace, use the distortion generator and quickly make a hyperjump. But first, you should determine the vulnerability of the fleet to make sure your trick will work.

Input

The first line of the input file contains a single integer n — the number of spacecrafts in the imperial fleet ($1 \leq n \leq 300$). Next n lines contain two integers x_i and y_i each — the coordinates of i -th spaceship in pseudospace. The coordinates do not exceed 30000 by absolute value.

Output

Print the vulnerability of the fleet on the first line of the output file.

Examples

standard input	standard output
4 0 0 0 1 1 0 1 1	6
3 1 2 2 3 3 1	0

Problem J. Exclusive OR

Given a sequence of n numbers $a_1, a_2, \dots, a_n$, find its subsequence $a_{k_1}, a_{k_2}, \dots, a_{k_l}$ such that the number $s = a_{k_1} \oplus a_{k_2} \oplus \dots \oplus a_{k_l}$ will be as large as possible, and q — the number of subsequences giving the maximal s . Here, $\oplus$ is the bitwise exclusive-or operation.

Input

On the first line of the input file, there is an integer number n ($1 \leq n \leq 10000$). On the second line, there is the sequence of integer numbers $a_1, a_2, \dots, a_n$ separated by single spaces ($0 \leq a_i \leq 10^9$).

Output

On the first line of the output file, print the numbers s, q and

l separated by single spaces — the xor-sum of the subsequence, the number of subsequences producing the same result, and the number of elements in one such subsequence, respectively. On the second line, print l numbers $a_{k_1}, a_{k_2}, \dots, a_{k_l}$ separated by spaces — the subsequence itself ($1 \leq k_1 < k_2 < \dots < k_l \leq n$). The number s should be maximal possible. You may print any of the subsequences which give s as the xor-sum.

Examples

standard input	standard output
3 1 2 3	3 2 1 3
6 1 10 2 9 4 8	15 4 4 10 9 4 8



It's not that hard to figure out that ONLY segment B is broken and the sequence of states the display is trying to produce is simply "9 -> 8 -> 7 -> 6 -> 5". Then the next number should be 4, but considering of the brokenness of segment B , the next state should be:



Problem K. Seven-segment Display

Tom is a boy whose dream is to become a scientist, he invented a lot in his spare time. He came up with a great idea several days ago: to make a stopwatch by himself! So he bought a seven-segment display immediately.

The seven elements of the display are all light-emitting diodes (LEDs) and can be lit in different combinations to represent the arabic numerals like:

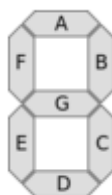


However, just when he finished the programs and tried to test the stopwatch, some of the LEDs turned out to be broken! Some of the segments can never be lit while others worked fine. So the display kept on producing some ambiguous states all the time...

Tom has recorded a continuous sequence of states which were produced by the display and is curious about whether it is possible to understand what this display was doing. He thinks the first step is to determine the state which the display will show **next**, could you help him?

Please note that the display works well despite those broken segments, which means that the display will keep on counting down **cyclically** starting from a certain number (can be any one of 0-9 since we don't know where this record starts from). 'Cyclically' here means that each time when the display reaches 0, it will keep on counting down starting from 9 again.

For convenience, we refer the seven segments of the display by the letters A to G as the picture below:



For example, if the record of states is like:

Input

The first line of the input gives the number of test cases, t ($1 \leq t \leq 2000$). Each test case is a line containing an integer n ($1 \leq n \leq 100$) which is the number of states Tom recorded and a list of the n states separated by spaces. Each state is encoded into a 7-character string represent the display of segment A - G , from the left to the right. Characters in the string can either be '1' or '0', denoting the corresponding segment is on or off, respectively.

Output

For each test case, if the input unambiguously determines the next state of the display print the next state (in the same format as the input). Otherwise, you should print "ERROR!" (without quotes).

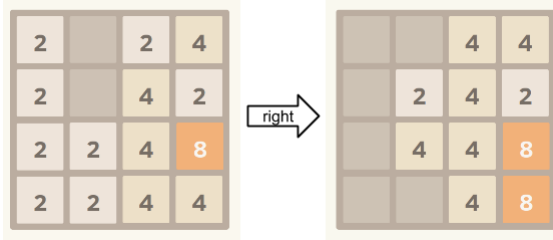
Examples

standard input	standard output
4 1 111111 2 000000 0001010 3 0100000 0000111 0000011 5 1011011 1011111 1010000 1011111 1011011	1110000 ERROR! 0100011 0010011

Problem L. Super 2048

2048 is a famous single-player game in which the objective is to slide tiles on a grid to combine them and create a tile with the number 2048.

2048 is played on a simple 4×4 grid with tiles that slide smoothly when a player moves them. For each movement, the player can choose to move all tiles in 4 directions, left, right, up, and down, as far as possible at the same time. If two tiles of the same number collide while moving, they will merge into a tile with the total value of the two tiles that collided. **In one movement, one newly created tile can not be merged again and always is merged with the tile next to it along the moving direction first.** E.g. if the three "2" are in a row "2 2 2" and the player choose to move left, it will become "4 2 0", the two most left "2" are merged.



The above figure shows how 4×4 grid varies when player moves all tiles 'right'.

Alice and Bob accidentally find this game and love the feel when two tiles are merged. After a few round, they start to be bored about the size of the board and decide to extend the size of board to $n \times n$, which they called the game "Super 2048".

The big board then makes them dazzled. They ask you to write a program to help them figure out what the board will be looked like after all tiles move to one specific direction on a given board.

Input

The first line of the input gives the number of test cases, t ($1 \leq t \leq 100$). t test cases follow. The first line of each test case gives the side length of the board, n ($1 \leq n \leq 20$), and the direction the tiles will move to, dir . n and dir are separated by a single space. dir will be one of four strings: "left", "right", "up", or "down".

The next n lines each contain n space-separated integers describing the original state of the board. Each line represents a row of the board (from top to bottom); each integer represents the value of a tile (or 0 if there is no number at that position). Each number in the grid is either 0 or a power of two between 2 and 1024, inclusive.

Output

For each test case, n lines, each containing n space-separated integers which describe the board after the move in the same format as the input.

Example

standard input	standard output
3	0 0 4 4
4 right	0 2 4 2
2 0 2 4	0 4 4 8
2 0 4 2	0 0 4 8
2 2 4 8	4 0 0 0 0 0 0 0 0 0
2 2 4 4	4 0 0 0 0 0 0 0 0 0
10 up	4 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0	4 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0	4 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0
2 0 0 0 0 0 0 0 0 0	0 2 4
2 0 0 0 0 0 0 0 0 0	0 4 8
2 0 0 0 0 0 0 0 0 0	0 8 16
3 right	
2 2 2	
4 4 4	
8 8 8	

Problem M. Addition

Six years ago, a robot, Bob, with infant's intelligence has been invented by an evil scientist, Alice.

Now the robot is six years old and studies in primary school. Addition is the first operation he learned in math. Due to his strong reasoning ability, he could now conclude $a + b = 12$ from $a = 2$ and $b = 10$.

Alice wanted to test Bob's addition skills. Some equations were given to Bob in form of $a = 2$, $b = 10$, $c = 4$, and Bob has to find out the answers of questions like $a + b$, $a + c$, etc.

Alice checked Bob's answers one by one in the test papers, and no mistake has been found so far, but Alice lost the given equations after a cup of coffee poured on them. However she has some of Bob's correct answers, e.g. $a + b = 12$, $a + c = 6$, $c + d = 5$. She wants to continue with the checkable equations, e.g. $b + d = 11$ could be concluded by $a + b = 12$, $a + c = 6$, $c + d = 5$, and thus the question $b + d$ is checkable.

To prevent the artificial intelligence technology from being under the control of Alice, you disguised yourself as her assistant. Now Alice wants you to figure out which of the rest of questions are checkable and their answers.

Input

The first line contains a single integer n ($1 \leq n \leq 5000$): the number of correctly answered questions. Each of the next n lines contain one correctly answered question in the form " $x+y=z$ ", where x and y are names of variables and z ($-200000 \leq z \leq 200000$) is a decimal integer.

The next line contains a single integer q ($1 \leq q \leq 5000$): the number of remaining questions. Each of the next q lines contain one question in the form " $x+y$ ", where x and y are names of variables.

Names of variables are strings of lowercase English letters. Each name contains at most 10 characters.

There is no contradiction in the answered questions and if the answer is checkable, the result is an integer.

Output

For each question in the input that was checkable, output a single line with the answer in the form " $x+y=z$ ", where x and y are names of variables and z is a decimal integer. Questions should be listed in the same order as they were given in the input. Please do **NOT** ignore duplicated questions, since Alice would fire you if you pointed any mistake of hers.

Examples

standard input	standard output
2 apple+banana=10 coconut+coconut=12 5 apple+banana apple+banana apple+apple banana+apple peach+apple	apple+banana=10 apple+banana=10 banana+apple=10
3 a+b=3 b+c=3 c+d=3 4 a+c a+d b+c b+d	a+d=3 b+c=3

Problem N. Cut Tiles

Enzo is doing renovation for his new house. The most difficult part is to buy exactly the right number of tiles. He wants n tiles of different sizes. Of course they have to be cut from the tiles he bought. All the required tiles are square. The lengths of side of the tiles are $2^{s_1}, 2^{s_2}, \dots, 2^{s_n}$. He can only buy a lot of tiles sized $m \times m$, and he decides to only cut tiles parallel to their sides for convenience. How many tiles does he need to buy?

Input

The first line of the input gives the number of test cases: t ($1 \leq t \leq 1000$). t lines follow. Each line start with the number n ($1 \leq n \leq 500$) and m ($1 \leq m \leq 2^{31} - 1$), indicating the number of required tiles and the size of the big tiles Enzo can buy. n numbers follow: $s_1, s_2, \dots, s_n$ ($1 \leq 2^{s_k} \leq m$), showing the sizes of the required tiles.

Output

For each test case, output one line containing the number of the big tiles Enzo need to buy.

Example

standard input	standard output
4 1 6 2 2 6 2 2 3 6 2 1 1 7 277 3 8 2 6 1 3 6	1 2 1 2