Problem D. Lines

Input file:	`lines.in`
Output file:	*`standard output`*
Time limit:	1 second
Memory limit:	256 mebibytes

You are given n lines on a plane. Your task is to select the maximum possible number of lines so that among the selected ones, no two lines are the same, no two lines are parallel and no two lines have an intersection at a point with $x = 0$.

Input

The first line of input contains one positive integer T, the number of test cases. The test cases follow.

Each test case starts with a line containing an integer n, the number of lines ($1 \le n \le 3000$). Each of the next n lines of input contain three integers A, B and C describing a line as a set of points (x, y) for which the equation $Ax + By + C = 0$ holds ($-10^9 \le A, B, C \le 10^9$, $A^2 + B^2 > 0$).

The sum of n in the input does not exceed 3000.

Output

For each test case, first, on a separate line, print the number k: the maximum possible number of lines that can be selected. On the next line, print k integers: the numbers of the chosen lines in any order. The lines are numbered starting from 1 in the order they are given in the input.

If there are several optimal answers, print any one of them.

Example

lines.in	standard output

2
2
1 1 0
1 1 1
2
-1 1 1
-2 1 1
	1
	1
	1
	1

Problem E. Yet Another Problem About Permutations

Input file:	`permutation.in`
Output file:	*`standard output`*
Time limit:	3 seconds
Memory limit:	256 mebibytes

This is a problem about permutations. If you are not familiar with some of the terms used below, please see the note following the example.

A permutation p is said to be *simple* if the length of each of its cycles does not exceed two. For example, permutation $2, 1, 4, 3$ is *simple*, but permutation $3, 1, 2$ is not.

You are given a permutation p. Your task is to represent it as a product of minimal number of simple permutations.

Input

The first line of input contains one integer T, the number of test cases ($1 \le T \le 10^5$). The test cases follow.

Each of next T lines describes a single test case. Each test case description consists of an integer n, the length of the permutation p ($1 \le n \le 10^5$), followed by n distinct integers $p_1, p_2, \ldots, p_n$, the permutation p itself ($1 \le p_i \le n$, each number from 1 to n appears in the permutation exactly once).

The total length of all permutations in the input is not greater than 10^6.

Output

For each test case, start by printing a line containing an integer k, the minimal number of simple permutations in the product. The next k lines must describe simple permutations $q^{(1)}$, $q^{(2)}$, …, $q^{(k)}$, one per line. On i-th of these lines, print n distinct integers from 1 to n describing permutation $q^{(i)}$. The product $q^{(1)} \circ q^{(2)} \circ \ldots \circ q^{(k)}$ must be equal to p.

If there are several optimal answers, print any one of them.

Example

`permutation.in`	`standard output`

`2`
`4 2 1 4 3`
`3 3 1 2`
	`1`
	`2 1 4 3`
	`2`
	`3 2 1`
	`1 3 2`

Note

A *permutation* of length n is a sequence of n integers where each integer from 1 to n appears exactly once.

A *cycle* in a permutation p is a sequence $i_1, i_2, \ldots, i_t$ of distinct integers from 1 to n such that $p_{i_1} = i_2$, $p_{i_2} = i_3$, …, $p_{i_{t-1}} = i_t$ and $p_{i_t} = i_1$. The number $t \ge 1$ is called the *length* of the cycle.

The *product* $a \circ b$ of two permutations a and b is a permutation c such that for each i, $c_i = a_{b_i}$. For example, if $a = 3\,2\,1$ and $b = 1\,3\,2$, their product is $a \circ b = 3\,1\,2$.

The *product* of three or more permutations can be evaluated in any order, for example, $a \circ b \circ c = (a \circ b) \circ c = a \circ (b \circ c)$.

Problem G. Shortest Accepted Word

Input file:	`shortest-accepted.in`
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

In this problem, we consider strings consisting of lowercase Latin letters "`a`", "`b`" and "`c`".

For this problem, let us define a *regular expression* recursively as follows:

1. A single character "`$`" is a regular expression accepting the empty string.
2. Single characters "`a`", "`b`" and "`c`" are regular expressions accepting strings "`a`", "`b`" and "`c`", respectively.
3. If P is a regular expression, "`(`P`)`" is also a regular expression accepting all strings accepted by P.
4. If P is a regular expression which is a **direct** result of applying any of the rules 1–4, its *iteration*, denoted as "P`*`", is also a regular expression accepting strings of the form $s = u_1u_2 \ldots u_k$ (a concatenation of zero or more strings) where k is any non-negative integer and each string u_i is accepted by P.
5. If P and Q are regular expressions which are **direct** results of applying any of the rules 1–5, their *concatenation*, denoted simply as "PQ", is also a regular expression accepting strings of the form $s = uv$ (a concatenation of u and v, each of which may be empty) where the prefix u is accepted by P and the suffix v is accepted by Q.
6. If P and Q are regular expressions which are **direct** results of applying any of the rules 1–6, their *union*, denoted as "P`|`Q", is also a regular expression accepting both strings accepted by P and strings accepted by Q.

The restrictions in rules 4–6 are imposed to prevent ambiguities and prioritize operations: when reading a regular expression, evaluate iteration, then concatenation, then union. Parentheses play the usual role of prioritizing operations enclosed in them. For example, the regular expression "`a(bac|ac*)`" is read as "accept `a` followed by either (`b` followed by `a` followed by `c`) or (`a` followed by zero or more copies of `c`)".

Given a regular expression r, find the shortest string s which is accepted by this regular expression r. If there are several such strings, find the lexicographicaly smallest one.

Input

The first line of input contains one integer T, the number of test cases ($1 \le T \le 300$).

Each of the next T lines describes a single test case. Each test case description consists of a regular expression r which is a string constructed by the above rules. Its length is from 1 to 300 characters.

The sum of all lengths of regular expressions is not greater than 300.

Output

For each test case print a single line containing the shortest string accepted by the given regular expression r. If there is more than one such string, print the **lexicographically smallest** one.

If the answer is an empty string, print a single character "`$`" instead.

Example

shortest-accepted.in	standard output
3 a ab|ac*(ca|cb) ((ab|ac)a)*	a ab $

Problem H. Work

Input file:	`work.in`
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

There are N men and M different working assignments for them. You are given a matrix A in which $A_{i,j} = 1$ if i-th worker is qualified to complete j-th job, and $A_{i,j} = 0$ otherwise. A worker can be assigned to a job only if he is qualified to complete that job.

Your goal is to assign workers to jobs in such a way that the distribution of the amounts of jobs done by workers is as close as possible to uniform (in Euclidean metric). This means that the N-dimensional vector in which i-th element is the amount of jobs completed by i-th worker must be as close as possible to the N-dimensional vector in which each element is equal to the real number M/N.

An additional requirement is that each job which can be completed at all must be assigned to exactly one worker.

Input

The first line of input contains two integers N and M ($1 \leq N, M \leq 300$). It is followed by N lines each containing M characters. Each of these characters is either '`0`' or '`1`'. These lines represent the matrix A.

Output

Print N lines. On i-th line, first, print k_i, the amount of jobs assigned to i-th worker. After that, print the numbers of those jobs. If there are several different ways to assign jobs and get an optimal distribution, print any one of them.

Examples

work.in	standard output

3 3
111
111
111
	1 1
	1 2
	1 3

2 4
0100
1100
	1 2
	1 1

Note

The Euclidean distance between two vectors $(u_1, u_2, \ldots, u_N)$ and $(v_1, v_2, \ldots, v_N)$ is the real number

$$\sqrt{(v_1 - u_1)^2 + (v_2 - u_2)^2 + \ldots + (v_N - u_N)^2}.$$

Problem I. Jam

Input file:	`jam.in`
Output file:	*`standard output`*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Two programmers sit in a car following the bus in a traffic jam. At the stop two people enter the bus. Then, three people are observed leaving the bus. Then one of the programmers said: If one person is to enter now, the bus will be empty!

Those programmers did several observations and wrote down numbers of people who entered and left the bus at the consecutive stops.

Your goal is to write a program that will calculate the minumum number of passenger in the bus before their observations started.

Input

The first line of the input consists of a single integer T ($0 < T \leq 50$) — the number of test cases.

Each of the test cases begins with a line containing a single integer M ($0 < T \leq 50$) — number of observations. Then sequential observations follow each on the new line. Each observation is described by two integers P_1 and P_2 separated by a space — number of people, entering the bus, and then number of people, leaving the bus, respectively ($0 \leq P_1, P_2 \leq 1000$). Note that one observation consists of two events: first, P_1 people enter the bus, then P_2 people leave the bus.

Output

For each test case, print the minimum number of people that would have to have been inside the bus at the beginning.

Example

jam.in	standard output

1
3
4 6
5 6
2 0
	3

Problem J. King of Guess

Input file:	`kingofguess.in`
Output file:	*`standard output`*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Well known guessing game ¡¡King of Guess¿¿ played in the next way: given a goal number N (hidden) and a range (X, Y) containing it (not including X and Y), everyone takes turns to guess what N is. After every wrong guess, the range replaces to a new range with either X or Y replaced by the guessed number. The game ends when a guess hits the number N.

Supposing that everyone always guesses the median of the range (if there are two, choose the smaller one), how many guesses need to be played for the game to end?

Input

The first line of the input file input contains three integers N, X and Y ($0 \le N, X, Y \le 10^4$, $X < Y$).

Output

Print number of guesses that will be played.

Examples

`kingofguess.in`	`standard output`
`42 20 80`	`3`

Problem K. Lesson

Input file:	`lesson.in`
Output file:	*`standard output`*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Alice and Bob are sitting on the lesson in the high school and playing Sea Battle. This game is played with two players, and follows these rules:

Each player has an $N \times N$ grid where they have placed four non-overlapping ships, each of the ships parallel to one of the axes. They place exactly one of each of the following ships:

Name	**Length**
Sail	1
Frigate	2
Cruiser	3
Dreadnought	4

Starting with the first player, they call out cells in the grid. If the cell called out by Player A was covered by a ship on Player B's grid, Player B has to tell Player A that it was a hit. If all cells covered by that ship have been hit by Player A, Player B has to announce that their ship sunk, and player A gets another turn. If no ship was sunk by Player A's move, it's Player B's turn. This continues until one player has sunk all the other player's ships. The player who managed to sink all the other player's ships is the winner.

To hide the game process from the teacher, Alice and Bob decided to automate the process a bit. Now they instead write down where they placed the ships and in what order they wanted to call out grid cells. They never call out the same grid cell twice.

Given the moves by each player, output what the players would have announced if they actually played the game. In other words, you have to find out which ships sunk (and in what order) and who the winner is, if Alice takes a first shoot.

Input

The first line of the input consists of a single integer, T, the number of test cases ($1 \leq T \leq 20$).

Each of the following T cases begin with a line consisting of a single integer, N ($4 \leq N \leq 10$), the size of the grids.
The following N lines represents Alice's grid, and each line consists of N characters. These characters are either '`.`', '`1`', '`2`', '`3`' or '`4`'. A '`.`' means that the cell is empty. '1', '2', '3' or '4' means that that cell is covered by one of Alice's ships. All cells with the same number are covered by the same ship (each ship will be represented as a contiguous line of the same character and be parallel to one of the axes; the number does not necessarily correspond to the length of the ship).

The next N lines represent Bob's grid, and has the same format as Alice's grid.
The next $N \cdot N$ lines represent Alice's planned moves. Each line has two integers, R_i and C_i, the row and column she calls out for that move (if the game gets that far).
The next $N \cdot N$ lines represent Bob's planned moves, and has the same format as Alice's moves ($1 \leq R_i, C_i \leq N$).

Output

For each test case output one line for each ship that was sunk, on the format "`PlayerX sank PlayerY's ShipName`", in the order they were sunk. The last line of the output for each test case should be the name of the winner.

Examples

lesson.in	standard output
1	Alice sank Bob’s Frigate
4	Alice sank Bob’s Sail
1..4	Bob sank Alice’s Sail
22.4	Alice sank Bob’s Dreadnought
3334	Alice sank Bob’s Cruiser
...4	Alice
112.	
..2.	
.32.	
4444	
1 1	
1 2	
1 4	
2 3	
3 2	
4 3	
2 1	
3 3	
4 1	
4 2	
3 4	
4 4	
1 3	
2 2	
2 4	
3 1	
1 3	
4 4	
3 4	
1 1	
4 3	
4 1	
2 4	
2 2	
1 2	
4 2	
3 1	
3 2	
2 3	
1 4	
3 3	
2 1	

Problem L. Maze

Input file:	`maze.in`
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Maze on the some asteroid have exactly one entrance/exit, contains no cycles and has no empty space that is completely enclosed by walls. A robot is sent in to explore the entire maze.

The robot always faces the direction it travels in. At every step, the robot will try to turn right. If there is a wall there, it will attempt to go forward instead. If that is not possible, it will try to turn left. If all three directions are unavailable, it will turn back.

Robot writes a log that describes his path, starting from the entrance/exit square until it finally comes back to it. The movements are described by a single letter: '`F`' means for- ward, '`L`' is left, '`R`' is right and '`B`' stands for backward.

Each of '`L`', '`R`' and '`B`' does not only describe the change in orientation of the robot, but also the advancement of one square in that direction. The robots initial direction is East. In addi- tion, the path of the robot always ends at the entrance/exit square.

Can you reconstruct the maze from the given log?

Input

First line of the input file contains one positive integer: the number of test cases T $(1 \le T \le 100)$. Each test case consists of one line with a single string: the log of movements of the robot through the maze.

Output

For each test case print one line with two space-separated integers h and w $(3 \le h, w \le 100)$: the height and width of the maze, respectively. Then print h lines, each with w characters, describing the maze: a '`X`' indicates a wall and a '`.`' represents an empty square.

The entire contour of the maze consists of walls, with the exception of one square on the left: this is the entrance. The maze contains no cycles (i.e. paths that would lead the robot back to a square it had left in another direction) and no empty squares that cannot be reached from the entrance. Every row or column — with the exception of the top row, bottom row and right column — contains at least one empty square.

Example

maze.in	standard output
3	4 4
FFRBLF	XXXX
FFRFRBRFBFRBRFLF	...X
FRLFFFLBRFFFRFFFRFRFBRFLBRFRLFLFFR	XX.X
	XXXX
	7 5
	XXXXX
	...XX
	XX.XX
	X...X
	XX.XX
	XX.XX
	XXXXX
	7 7
	XXXXXXX
	X...X.X
	X.X...X
	X.X.XXX
	..XXX.X
	X.....X
	XXXXXXX