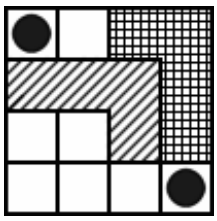
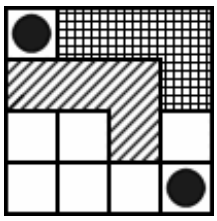
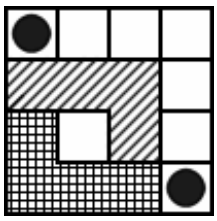
L Game © Edward de Bono

The L Game was designed by Edward de Bono who enjoys playing games and yet hates to concentrate on a large number of pieces. The intention was to produce the simplest possible game that could be played with a high degree of skill. The L game was the result.

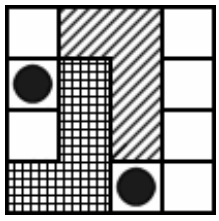
Each player has only one piece, an 'L piece'. There are also two neutral square pieces. The board is four squares by four squares. The object of the game is to manoeuvre the other player into a position on the board where he cannot move his L piece.

Proceeding from the starting position, the first player (and each player on each move thereafter) must move the L piece first. When moving, a player may slide, turn or pick up and flip the L piece into any open position other than the one it occupied prior to the move. When the L piece has been moved, a player may move either one (but only one) of the neutral square pieces to any open square on the board. It is not required that a neutral piece be moved, this is up to the player!

From any of the three positions above, the player with the checkered L can move his L piece to two new locations (the other two of the three positions above). After moving the L piece, he can move one of the neutral pieces to any of the 6 remaining empty squares or decide not to move any of the neutral pieces. All in all, there are 2 * (6 + 6 + 1) possible moves.

A player wins the game when his opponent cannot move his L piece. In the position below, if the player with the checkered L is to move, he loses, as he cannot move his L to a new, unoccupied position on the board.



The game is simple, yet hard as there are so many moves. There are over 18,000 positions for the pieces on the small board and at any moment there may be as many as 195 different moves of which only one is successful.

Write a program that given the position of the pieces and which player is to move, decides if the player has a winning move and output such a move if it exist. If there exist several winning moves, output any one of them. If no winning move exists, you should decide whether the game will end in a draw (assuming perfect play from both players), or if the player to move will in fact lose.

A winning move is defined as a move that, no matter how the opponent plays, he cannot avoid losing the game if the player who is about to move continues the game playing perfectly.

A player is losing the game if, no matter which move he plays, he cannot avoid losing if the opponent continues to play the game perfectly.

If neither of these conditions hold (that is, none of the players can force a win), we say the game position is a draw.

Input data

The input file `lgame.in` consists of four lines describing the position of a game in progress. A dot (‘`.`’) represents an empty square, an ‘`x`’ (lowercase X) represents a neutral square piece, ‘`#`’ represents the L piece of the player A and ‘`*`’ represents the L piece of the player B. The player A is about to move. You may assume that the position is legal and that the player to move has at least one legal move from the current position.

Output data

Output data should be written to the file `lgame.out`. If a winning move exists, output the position after the winning move, using the same format as the input. Otherwise output `No winning move` on a line by itself, and on the next line either `Draw` if the game will end in a draw (assuming perfect play) or `Losing` if the player to move will lose the game.

Examples

lgame.in	**lgame.in**	**lgame.in**

`.***`
`#*.x`
`###.`
`x...`
	`...x`
	`###.`
	`#***`
	`x..*`
		`.###`
		`x#*x`
		`***.`
		`....`

lgame.out.	**lgame.out.**	**lgame.out.**

`.***`
`x*#x`
`###.`
`....`
	`No winning move`
	`Draw`
		`No winning move`
		`Losing`

Grading

In this task, a program receives points for tests with no winning move only if it solves correctly at least one of the tests with a winning move.