Problem G. Squares Game

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

This is an interactive problem. In this problem, you have to win against an opponent who makes random moves.

Felix and Sophia play a game on a rectangular board consisting of $m \times n$ square cells. They move in turns. Each cell of the board is either free or painted. Initially all cells are free. A move consists in selecting any four free cells which form a 2×2 square and painting these cells. The player who can not make a move loses the game.

On each of his turns, Felix chooses one of the possible 2×2 squares uniformly at random and paints it. Your task is to play as Sophia. Your solution will be considered correct if it loses no more than 10 games out of 300.

In this problem, each test specifies the size of the board. In each test, you have to play exactly 300 games. In each game, Felix moves first and Sophia moves second.

Input

The first line contains two integers m and n separated by a space: the size of the board ($16 \le m, n \le 25$). Note the lower limit: the board can not be too small. The rows of the board are numbered from 1 to m, and the columns from 1 to n.

The following input is divided into blocks, and each block corresponds to one game. Each block starts with a line "`start` k" where k is the number of the game starting from 1. Each of the following lines has the form "`move` r c" ($1 \le r < m$, $1 \le c < n$). Such a line means that on his turn, Felix painted four cells with coordinates (r, c), $(r, c+1)$, $(r+1, c)$ and $(r+1, c+1)$. If a game finishes because Sophia wins, the line given instead of the next Felix's turn is "`won`", and after that, the block terminates. If a game finishes because Felix wins, the next line after the Felix's turn is "`lost`", and after that, the block also terminates. The blocks follow one after another without any additional separators. After the last block, there is a single line "`end`".

It is guaranteed that Felix makes all moves according to the rules of the game, and each move is chosen pseudo-randomly and uniformly. It is worth noting that, before the first game, the pseudo-random number generator is always initialized by the same value. This means that, if your solution is playing as Sophia deterministically, on each test, all moves and the results of all games will remain the same if the solution is run repetitively.

Output

Each time when it is Sophia's turn in a game and there is at least one possible move, print a line of the form "`move` r c" ($1 \le r < m$, $1 \le c < n$). Such a line will mean that the move you selected for Sophia is to paint four cells with coordinates (r, c), $(r, c+1)$, $(r+1, c)$ and $(r+1, c+1)$.

To prevent output buffering, after printing each line, consider issuing the command which flushes the buffer. For example, this command may be `fflush (stdout)` in C or C++, `System.out.flush ()` in Java, `flush (output)` in Pascal or `sys.stdout.flush ()` in Python.

If your solution makes an invalid move or loses more than 10 games, it gets the «`Wrong Answer`» outcome on the respective test.

Example

standard input	standard output
2 5	*<reading input>*
start 1	*<reading input>*
move 1 4	*<reading input>*
<waiting for output>	move 1 1
won	*<reading input>*
start 2	*<reading input>*
move 1 2	*<reading input>*
<waiting for output>	move 1 4
won	*<reading input>*
end	*<terminating>*

Explanation

When your solution is checked, the first test is the example from the statement. Note that the constraint $m, n \geq 16$ is not satisfied for this test, but still, it is satisfied for all tests starting from the second one. Furthermore, in the example test, there are only two games instead of 300 games as in all other tests. The choice of moves by Felix does not depend on Sophia's moves and is determined in advance. Finally, on a 2×5 board, the second player wins regardless of how the game goes.

Your program passes the example test if it wins both games and successfully terminates after that. Recall that your program passes a regular test if it loses no more than 10 games and successfully terminates after that.