2016 Cross-Strait Research forum

Coding Contest

Date: 20 July 2016 (WED)

Coding Contest (Round 1)

Time limit: 10.000 seconds

Problem description

4×4 tic-tac-toe is played on a board with four rows (numbered 0 to 3 from top to bottom) and four columns (numbered 0 to 3 from left to right). There are two players, *x* and *o*, who move alternately with *x* always going first. The game is won by the first player to get four of his or her pieces on the same row, column, or diagonal. If the board is full and neither player has won then the game is a draw. Assuming that it is *x*'s turn to move, *x* is said to have a ***forced win*** if *x* can make a move such that no matter what moves *o* makes for the rest of the game, *x* can win. This does not necessarily mean that *x* will win on the very next move, although that is a possibility. It means that *x* has a winning strategy that will guarantee an eventual victory regardless of what *o* does.

Write a program that, given a partially-completed game with *x* to move next, will determine whether *x* has a forced win. You can assume that the game has not already been won by either player, and that the board is not full.

Input

The input contains the total number of test cases, followed by one or more test cases. The characters used in a board description are the period (representing an empty space), lowercase *x*, and lowercase *o*. There should be a blank line between each test case.

Output

For each test case, output the test case number, followed by the (*row*, *column*) positions of the forced win for *x*, or '#####' if there is no forced win. Your program should search for a forced win by examining positions (0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), ..., (3, 2), (3, 3), in that order, and output all the forced win positions for x. There should be a blank line between the result of each test case.

Sample Input	**Sample Output**
`2`	`Test Case #1:`
`....`	`#####`
`.xo.`	
`.ox.`	`Test Case #2:`
`....`	`(1, 0)`
	`(2, 3)`
`xooo`	`(3, 1)`
`.xxx`	`(3, 2)`
`oox.`	`(3, 3)`
`o...`	

Remarks:

- In the second test case above, *x* can win immediately by playing at (1, 0) or (3, 3), but playing at (2,3), (3, 1) and (3,2) will still also ensure victory (although it unnecessarily *delays* it).
- Format the output exactly as shown in the example above.

Submission

Submit the source code of your program (WinningMoves.c, WinningMoves.cpp or WinningMoves.java) at https://submit.comp.polyu.edu.hk.

Judging Criteria

The performance of the teams will be ranked by the program's correctness, and ties are broken by the total execution time of all the successful test cases.

Coding Contest (Round 2)

Presentation

Extend your program to solve the 5×5 version of the state based problem. Discuss the algorithm design, the optimization applied, and the performance of your program.

Judging Criteria

The performance of the teams will be assessed by a judging panel consisting of professors from different universities.

Criteria:

- Algorithms and optimization techniques
- Organization and clarify of the presentation.