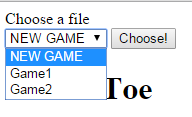
**CSSE 280 Exam 2 Oct 23, 2015 (Session 21)**

You may use your textbook, programs that you have written (not programs written by other students in this course from any term), reference websites such as w3schools.com, php.net, webstepbook.com. You may not search for programs that solve something like the specific problem on this exam. And of course you may not communicate (electronically or otherwise) with anyone other than the instructor. No headphones or earbuds during the in-class exam.

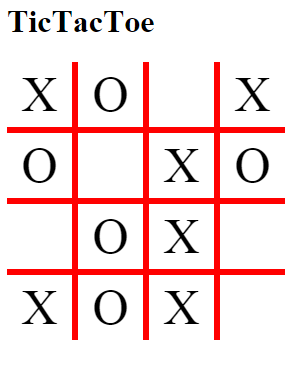
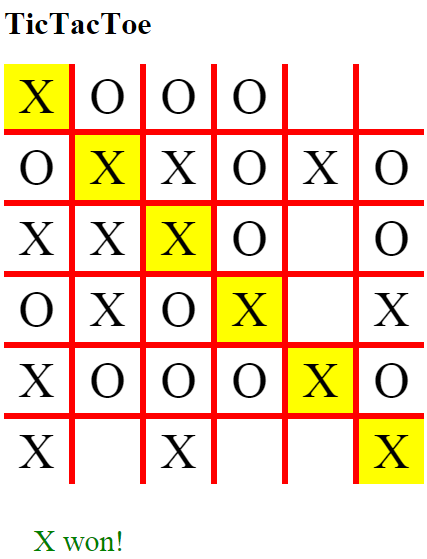
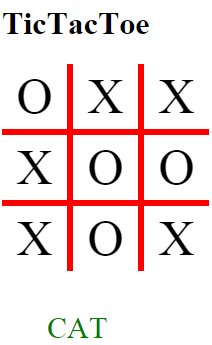
This problem combines the PHP techniques from SearchHighlighter and WordSearch with the JavaScript techniques of 15-puzzle, combined to require you to figure out slightly new ways of representing data and checking for conditions relevant to this problem. You should be able to copy, paste, and tweak a lot of code from these previous problems, and of course you will need to write a significant amount of new code that is specific to TicTacToe.

I will not try to describe in writing the rules of TicTacToe. You can find them here: <https://en.wikipedia.org/wiki/Tic-tac-toe>. I will instead demonstrate my program live in class (at 7:35, 8:10, and 8:30).

I will describe the features that are required for full credit (you can get some of them working and earn partial credit).

1. The PHP program **TicTacToe.php** initially puts up a simple form (same appearance as the form we used for 15 puzzle) that uses radio buttons to allow the user to choose the board size. Unlike the 15-puzzle form (but like the SearchHighlighter form), clicking the Choose button causes a submission back to the server.
2. Upon submission, the server gets the board size from the HTTP request parameters, then looks in the Files folder to find all "saved game" files whose names begin with the character of the selected size. All game files have a .txt extension.
3. A file has only one line, which contains:
   1. First character is the board size, a single character.
   2. Next a vertical bar (acts as a separator.
   3. A character that indicates ("X" or "O" whose turn it is.
   4. SIZE\*SIZE characters, representing the state of each square on board.
   5. The characters are X, O and b; the b represents an empty square.

Example: 4|O|XObXObXObOXbXOXb

1. The erver sends back to the browser a <select> element, whose options are the names of the files. I chopped off the file extension and the number from the beginning of each file name, to make the menu more readable. You are not required to do that. This drop-down list also has NEW GAME as an option. Of course if the user choses NEW GAME, the game starts with an empty board, with X as the first player.
2. The server also includes on the HTML page it sends back the *contents* of all files that have the given board size (but this info is hidden, so it does not display hidden). One thing you have to figure out is how to connect the file names in the menu with the file contents in hidden dives the JavaScript code. This will enable the JavaScript code to grab the appropriate file contents when the user selects a filename without having to go back to the server.
3. Once the PHP program has sent this file selection form and file contents to the browser, all of the rest of the functionality is done in the browser by JavaScript code.
4. The JavaScript code installs an event handler for the Choose! Button, This event handler builds the board and the array of squares (similar to 15-puzzle) based on the info from the chosen file (the hidden contents in the HTML page sent by the server); then it removes the file-selection form from the page.
5. The board has no outside borders (though you can get some partial credit even if you have them). This is accomplished by putting appropriate borders on each individual square.
6. Each square is clickable. When an occupied square is clicked, nothing happens.
7. Clicking an empty square causes an X or O to appear there, depending on whose turn it is. Then a check is made to see if that move wins the game for the current player by having SIZE squares in a row (horizontal, vertical, or diagonal) occupied by that player's symbol. If so, the winning squares are highlighted, and a message indicating who won appears below the board.
8. If the board fills up without having a winner, the tie is called a CAT game, and your program should display a message that illustrates this.
9. I started with my solution and removed a significant amount of the code. You may choose to use any or all of my code (or not!). Of course you will have to write code of your own.
10. Credit will be based almost entirely on how much of the functionality you get working; very little (If any) partial credit will be given for "good ideas in your code".
11. I expect that most students will successfully implement a lot of this functionality in the time allotted for the exam, but I also expect that most students will not get all of it in that time. You must submit what you have before the end of the exam time, but you can also keep working on it. If you submit again (to a different folder) before Saturday at noon, you can earn up to 50% of the points that you do not earn during the exam time. For example, if you earn 64 points during the exam time, you can earn up ot 18 additional points for things you get working later.
12. Further work on the exam after 9:50 today is to be done under exam restrictions. I.e. it is still a solo effort; you may not communicate with anyone else about any technical material from the course until after you have submitted your "after the exam" work.
13. I suggest that you limit yourself to spending no more than three hours on the exam after 9:50 today.