Tic-Tac-Toe and Functions
Lab 5

Prepared by Matt Bennett & Eduardo Cortes
for ELEC 3150
Professor Carpenter - Fall 2016

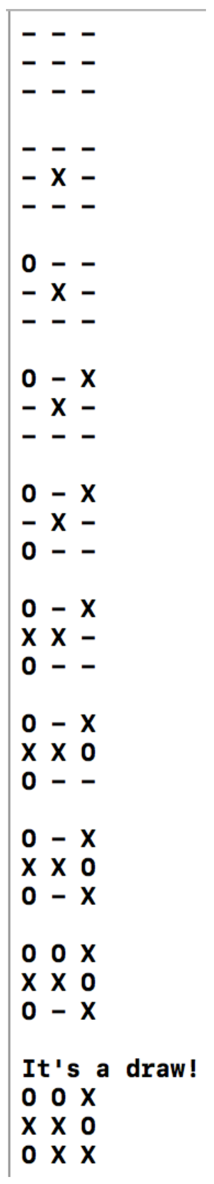
I - Introduction

The goal of this lab was to simulate a tic-tac-toe game using algorithms developed for ‘X’ and ‘O’ players. Both algorithms place the best move in a tic-tac-toe game board given a board in a three by three character array. This lab required both algorithmic thinking to determine the best move of a given board and knowledge of pass by reference C++ functions. The simulated game results in the same draw outcome for each run.

II - Procedure and Assumptions

Both placement algorithms prioritize winning for the given player and attempt to block the opposing player from winning. The general program functions by alternating the X and O player turns checking for end game conditions after each player has updated the board with their move. Neither algorithm use elements of randomness resulting in a singular simulation result.

III - Results



```
- - -
- - -
- - -

- - -
- X -
- - -

O - -
- X -
- - -

O - X
- X -
- - -

O - X
- X -
O - -

O - X
X X -
O - -

O - X
X X O
O - -

O - X
X X O
O - X

O O X
X X O
O - X

It's a draw!
O O X
X X O
O X X
```

Initial board passed to player X

Player X places center when possible.

Player O cannot win; does not need to block Player X; places in first available corner.

Player X cannot win; does not need to block Player O; places in first available corner.

Player O cannot win; Needs to block Player X.

Player X cannot win; Needs to block Player O.

Player O cannot win; Needs to block Player X.

Player X cannot win; does not need to block Player O; places in first available corner.

Player O cannot win; does not need to block Player X; No corners available; places in first open slot.

Player X cannot win; does not need to block Player O; No corners available; places in first open slot.

Figure 1: Algorithm Playthrough

IV - Analysis

The simulated game functions as expected, placing the best logical move for each player given the initial board condition. Both algorithms block the opponent when not able to place a winning move. The result of the two algorithms playing a complete game against each other is shown in Figure 1 in results.

V - Conclusions

Both algorithms function as expected placing the best moves given each input board. The algorithms playing the game results in the same outcome of moves each time. In order to improve upon this and generate more interesting games, algorithms could be altered to generate a list of move of equal priority and picking a position at random from this list.

Appendix A - Team Evaluation

Eduardo and I worked together to generate the basic program flow of the main, isWinner and isDraw functions. Eduardo had a general concept of the how to divide the project into modular functions and the logic that these functions required. Working together and using Eduardo's understanding of the best move placements for O, we worked to turn his concept into C++ code that had been syntactically causing issues based on his various ideas. I attempted to direct him toward a programmatic mindset without providing solutions, however, because of this our algorithms ultimately are similar.