TIC
TAC
TOE

By Andrew Yaros
CS172-062 - Spring 2017
Homework 3

SYSTEM MANUAL

This system manual covers the tic class and header files. The tic class utilizes the vector, iostream, and string classes, as well as "symbol.h" and "tic.h".

Private attributes

The tic class has the following attributes:

- Row vectors; each of the following vectors contains one row of the board.
  - vector<symbol> row0_
  - vector<symbol> row1_
  - vector<symbol> row2_
  - Each item in the vectors is a spot on the board. (Symbol objects can be "X", "O", or "BLANK")
- symbol winner_
  - Used by the game_over function to record which of the players won. If there is a tie, this variable is not changed.

Constructors

The default (and only) constructor creates a new board by adding three blanks to each row. The resulting data structure is a 3 x 3 grid of rows and columns:

	col0	col1	col2
row0	**0, 0**	**0, 1**	**0, 2**
row1	**1, 0**	**1, 1**	**1, 2**
row2	**2, 0**	**2, 1**	**2, 2**

Inspectors

- symbol getRowItem(int x, int y) const
  - This function returns the value of the symbol on the board at the Xth row and the Yth column. It is used when drawing the board.
- winner()
  - Returns the value (x, o, or blank) of the attribute winner_, which can be set by the setWinner method (which is used by the game_over facilitator).

Methods

- Move(symbol m, int x, int y)
  - This function is used to actually make a move on the board.
  - First, it checks to see if the symbol at row X and column Y is blank. If so, it is changed to the value of m (which is either x or o depending on which player's turn it is). Since this move was legal, the function returns true.
  - If the x and y coordinates point to a spot on the board that is not blank, then someone has already put an x or o into this spot on the board. Therefore, the move is illegal and the function returns false.
- setWinner(symbol m)
  - This is used in the game_over function (see below).

Facilitators

- game_over()
  - After a move is made, this function checks if the game is over. If so, it returns true, otherwise it returns false.
  - Next, it checks the rows, columns, and diagonals to see if a player has three of their symbols in a row. For each row/column/diagonal:
    - A player has won if the three non-blank symbols in the row/column/diagonal are identical.

    - If a player has indeed won, then setWinner sets winner_ to the value of an item in the row/column/diagonal
  - If, after checking all rows, columns and diagonals, the game has not been won, it means there is either a tie, or the game is still in progress.
    - The facilitator now checks the board for blank items. As soon as a blank spot is found, the function returns false (the game is not over unless all blank spots have been filled).
    - If there were no blank spots on the board, then all areas have been filled in and a tie has been reached, which means the game is over. The function returns true.

Operators

- ostream & operator<<(ostream& os, const tBoard& myTable)
  - The << operator has been overloaded to allow the board to be drawn. This is done by outputting the contents of the board myTable to the ostream os; the ostream is then returned.