**Abir Razzak**

**SYSTEM MANUAL**

**What is tBoard?**

tBoard is the name of the object that is used to represent the tic-tac-toe board. It is essentially a 3x3 matrix that stores the symbol object that is included with the assignment.

**Matrix?**

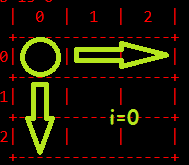
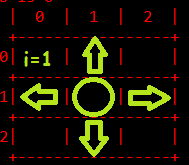
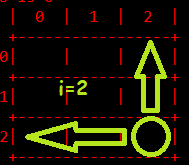
Matrix is a vector of vectors that store symbol objects. Matrix holds 3 vectors representing the 3 rows of the board. The vectors within matrix hold 3 symbol objects, representing the column spaces in that particular row.

**How does it determine that a game is over?**

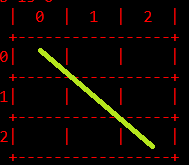
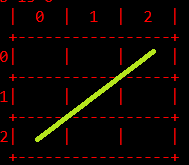
After every turn the program will run the tBoard member function called game\_over(). While this function returns false the game will continue. The game\_over() function works by first checking if there is a winner to the game yet. It does this by running another tBoard member function named winner().

winner() runs as follows:

1. First the function will check each row and column to see if the symbols all match\*

2. If none of the rows or columns match, then the function will check if the diagonals match\*

3. If that does not find a match, then the program will return false

\**A match occurs if all symbols match with the exception of the BLANK symbol*

If winner() returns true, then game\_over() will immediately return true as well. After winner() is called in the game\_over() function, it will then begin to check every space in the matrix to determine if there are any blank spaces left on the board. If there is a blank space, then game\_over() immediately returns false. If both these conditionals do not cause a break in the function, then the game is officially over with no more blank spaces left and no winner, resulting in a tie.

**What does overloading the << operator for tBoard do?**

Basically, all that jumble of code does is draw out the current state of the board to the ostream.

Like so:

