Overview:

The goal of this assignment is to gain experience with arrays and functions. For this assignment, you will write a program that plays the popular game of tic-tac-toe. The game will have one human player, and one computer player. The human player will be the X's, and the computer the O's.

Operation of the game:

The human player (player X) will always go first. In the beginning of the game, and after each player makes a move, the game board will be printed. Each of the possible locations in the game board are assigned a number (1-9), as follows:

```
 1 | 2 | 3
---|---|---
 4 | 5 | 6
---|---|---
 7 | 8 | 9
```

When the game begins the above game board is printed, and player X is prompted for a move, like:

```
 1 | 2 | 3
---|---|---
 4 | 5 | 6
---|---|---
 7 | 8 | 9

PLAYER X: Enter Desired Location >
```

The player will enter in a number (an integer) between 1 and 9. The number entered indicates the location in which player X wishes to place an X. Once the player enters the location, the game board is updated, and reprinted. For example, if the location entered is 6, the following is printed:

```
 1 | 2 | 3
---|---|---
 4 | 5 | X
---|---|---
 7 | 8 | 9
```

The computer will now make its move automatically (see the following sections for details on how to choose the computer's move). The choice the computer makes, along with a new game board are then printed. For example, if the computer chooses location 5, the following would print:

```
 1 | 2 | 3
---|---|---
 4 | 5 | X
---|---|---
 7 | 8 | 9
```
COMPUTER O: I Choose 5.

```
  1 | 2 | 3  
  -------
  4 | O | X  
  -------
  7 | 8 | 9  
```

PLAYER X: Enter Desired Location >

This process repeats and the game proceeds in this manner, printing the updated game board after each move, until there is a winner or a tie.

**Error Checking:**

When the player enters a location, its validity must be checked. If the location entered is not between 1 and 9 (inclusively), the following error message should be reported.

Invalid location, please try again.

If the location entered, say k, is between 1 and 9 (inclusively), but that particular location has already been taken (i.e. there is an X or O already there), then the following error message should be reported:

Location k is not available, please try again>

In either case, player X is prompted again to re-enter the choice of location.

**Detecting the end of the game:**

After each move (either from the computer or the human player), the program should check for the end of the game. A winner exists if all three locations in any row, column, or diagonal have the same token (O or X). A tie happens if there is not a winner, but all nine squares have been taken. Either a tie or a win indicates the end of the game. If a winner is found, the winner should be reported as follows:

X has won!  
Or  
O has won!

depending on which player won the game. If a tie is found, the following message should be reported:

The game is a tie.

Once the end of a game is reached, the program should prompt the user for another game, like

Would you like to play again? (1 for yes, 0 for no)
If the user enters a '1' then a new game is started, otherwise the program quits.

**Algorithm for the computer's move:**

There are many different ways in which the computer can choose its next move, however we are asking you to implement the following algorithm for deciding what the computer selects as its move:

1. If it is possible for the computer to win (i.e. there is an empty location in which an O can be placed and the resulting game board will have at least one row, column or diagonal with all Os) then the computer should take this winning move. If you find more than one possible winning move, pick one at random.
2. If there is a threat of the human player winning in his next move, block this move. That is, if there is a location on a row, column, or diagonal that is empty, which already has two X's, then move into this location to block this victory. If you find more than one such location, pick one at random.
3. Otherwise, if neither 1 nor 2 applies, move into the center if it is not already taken by X.
4. Finally, if none of choices 1, 2 or 3 apply, move into the location with the smallest integer number that is available.

We know that there are algorithms which if implemented computer would never lose. However for the assignment to be turned in here we are asking you to implement the above.

**Implementation Requirements:**

It is up to you how to represent the game board and implement the needed algorithms. However, one way to represent the game board is with a linear (1D) array where each index corresponds to a location on the board. An alternative method would be to use a two-dimensional (3 x 3) array. Remember that array indices in C start with 0.

You are required to implement the program using "functions". Your program must contain the following functions:

1. Function to print the game board on the screen. – void printGame();
2. Function to calculate the position for computer's next move. – int calculateMove();
3. Function to validate the entry from the user.– int isValid();
4. Function to check the current status of the game for a win.– int winStatus();

You may have additional functions, but the above functions are a minimum expected. The arguments to the above functions would depend on your implementation choices.

**Miscellaneous:**

Your program should be structured in a logical manner and be well documented. Do not use any global variables.
You must submit your programs electronically using the turnin command. You will ONLY submit your source code. Please name your source code file “assign3.c”. You can then turn in your program with the following command from your ece account:

`turnin engr170 assign3 assign3.c`

You can turn in your file as often as you like before the due date. When you issue the above command, you will overwrite all previously submitted files. No late assignments will be accepted.