Lab #2

Tic Tac Toe – Part 1

For this lab, we will be writing a program that runs a Tic Tac Toe game. In Tic Tac Toe, players take turns filling up a 3x3 grid with X’s and O’s until one player successfully manages to place three X’s or O’s in a row. Full rules can be found here: <https://en.wikipedia.org/wiki/Tic-tac-toe>

In order to implement tic tac toe we will need to design methods which accept input from players, as well as methods that check whether or not the game has been won.

1. To begin, open VSCode to a folder called “tictactoe”. In that location, create a new console application from a template by opening the terminal and using the command dotnet new console.
2. Define a class *TicTacToeGrid* in a namespace *TicTacToe.*
3. Inside this class, create a property *grid*. This grid property should be a 3x3 array of chars.
4. You should define a TicTacToe constructor which initializes the grid to be the correct size, as well as to ensure that each location in the grid contains a blank space to begin (“ “).
5. Create a method *placeCharacter* which takes as input a character to place (X or O), as well as the row and column to place that character in. The *placeCharacter* method should place the character at that location in the grid. In the event that the character is invalid (not X or O), or the row or column does not fit within the grid, the method should repeat the request until the user gives valid input.

Hint: The Convert.ToInt32() and Console.ReadLine() methods will be useful here. See MSDN C# documentation for details.

1. Create a method *printGrid* which prints out the layout of the grid. (Use whatever method of display that works best for you, don’t spend too much time here. Just printing all the characters in the correct location would be sufficient, though if you want to go above and beyond you can actually draw out the grid itself.)

Hint: The Console.WriteLine() method will be useful here. See MSDN C# documentation for details.

1. Create a method *checkRows*, which should check all the horizontal rows (and only the rows) in the grid to see if either player has won. If X has won, the method should return X. If O has won, the method should return O. If neither player has won, the method should return an empty string “”.

(Note: As you implement this method, be mindful about modularity. Try not to rely on the fact that you are dealing with a 3x3 grid and implement this method in a way that it would find the answer for a grid of any size.)

1. Test the above method to make sure it works correctly!
2. Create a similar method *checkColumns* to check the columns. Test it.
3. Create a similar method *checkDiagonals* to check the diagonals. Test it.
4. Create a method *checkGrid*, which calls the above three methods, and as above, returns “O” if O has won, “X” if X has won, or “” if neither have won. If the entire gird has been filled, the String “Tie” should be returned.
5. Now create a second class *TicTacToeGame,* also in the *TicTacToe* namespace*.* Inside this class, define a Main method which creates a *TicTacToeGrid*, and then repeatedly asks players to input the location at which they would like to place X’s and O’s. After each character is placed, you should call the *checkGrid* to see whether the game has ended. If it has ended, you should print the winner, or “Tie” if the game resulted in a Tie.
6. Bonus: If you have time remaining, try to modify your existing TicTacToeGrid so it now plays on a 4x4 grid. If your *checkRows*, *checkColumns,* and *checkDiagonals* methods were designed with modularity in mind, this should be relatively easy!

If you find you are needing to modify these methods, try to change them so that if we go back to a 3x3 grid, you will not need to modify them again!