ITP 165: Introduction to C++ Programming

Homework 5: Tic-Tac-Toe Due: 11:59pm, October 2nd, 2015

Goal

For this assignment, you will be making a game! You will be implementing the two-player game Tic-Tac-Toe. Part of the logic has been taken care of for you and provided to you in the form of a header file full of functions. This assignment will give you practice with calling functions that have input and return values.

Setup

- Create a new project in Visual Studio or Xcode called **homework05** and a new C++ file named **tictactoe.cpp**.
- Your **tictactoe.cpp** file must begin with the following (replace the name and email with your actual information):

```
// Name
// ITP 165, Fall 2015
// Homework 05
// USC email
#include "tictactoe.h"
```

- Notice that the first line of code is the `#include "tictactoe.h"`. This must come before any other code in your file (including the other `#include` lines).

Part 1: Variable setup and board display

- To start, you will need to set up a few variables to complete the logic of Tic Tac Toe:
  - You will need a `char` variable to track and display the winner of the game.
  - You will need two `bool` variables: the first to track whose turn it is (Player X or Player O), the second to track the validity of user input.
  - You will need two `int` variables: the first for user input, the second will be a counter for the number of total turns taken.
  - You will also need an array of type `char` and size 9 to display the game board and store the user's moves. It will be used to display a 3x3 grid, and will be initialized with the character representations of the numbers 1 through 9 (remember this is a `char` array).
- For now, just declare but do not initialize the other variables, other than the `char` array. They will have to be initialized according to our logic, which we will implement later.
- Next, call the provided *PrintBoard* function with the `char` array as an input parameter to display the game board.
  - For more information on how the function *PrintBoard* works, please refer to tictactoe.h
- Your output in Part 1 should look like:

```
 1 | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Press any key to continue...
```

Part 2: User input for player moves

- Notice our array displays the numbers 1 through 9, to signify to the user where there is a free place to move for their turn. Ask the user for a number ranging from 1 to 9 and store it in the first `int` variable that we created for user input in Part 1.
- You will also need to check the validity of the user's move – Tic-Tac-Toe rules state that you may not make a move where a move has already been played. To do so, you will be using another provided function named *IsValidMove*, which returns a Boolean true or false depending on if the move is allowed, and takes in as parameters the `char` array that stores the game board as well as the index of the move the player is trying to make.
    - Please refer to tictactoe.h to review the *IsValidMove* function and comments for more details on how the function works.
- Store the return value of the *IsValidMove* function into the second `bool` variable you created in Part 1. This variable will be used in a data validation loop.
- Setup a loop to validate the user's response. This loop will be working on a more complex conditional than normal. You will need to check that the user's response is in the correct range (1-9) and ALSO does not attempt to overwrite a previous move (in other words, it is a valid move).
    - Remember to now initialize the `bool` variable we created in Part 1 to test answer validity. You will probably want to set this `bool` to something that sets your loop condition to `true`, so you can enter the loop that allows for user input.
    - Be sure to let the user know if they have entered an invalid response, and ask for the input again.
- Your output after completing Part 2 should look similar to this: (User input in **red**)

```
 1 | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 0
INVALID MOVE. PICK AGAIN.
Where do you want to go? Pick an available number: 1
Press any key to continue...
```

Part 3: Output user's move

- Now that the user has entered valid input on where they would like to place their move, you have to store this choice and make that move for them.
- First initialize the `bool` variable we created before to track whose turn it is. In typical Tic-Tac-Toe rules, the player who plays `'X'` goes first.
- Now depending on whose turn it is, you will either store an `'X'` or an `'O'` in the `char` array, at the space the user input in Part 2.
    - Remember our game board is numbered 1 to 9, but that arrays start at index 0.
- After you have stored the player's move, call the *PrintBoard* function again, with the updated array as a parameter, to show the player the move they made.

- Output after Part 3 should look similar to this: (user input in **red**)

```
 1 | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 0
INVALID MOVE. PICK AGAIN.
Where do you want to go? Pick an available number: 1
 X | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Press any key to continue...
```

Part 4: Check if there is a winner

- You have another function provided to you that handles the logic of checking which player has won. This function is called *WinnerIfAny*, has a parameter of the game board array, and returns a `char` that signifies who is the current winner. Either `'X'` or `'O'` or `'N'` is returned, depending if X has won, O has won, or No one has won.
    - Please refer to tictactoe.h for the function and comments for further understanding on how this function works.
- Initialize the `char` variable from Part 1 that we created to store the winner, to be the return value from the *WinnerIfAny* function.
- Now, we also want to check if there has been a tie. Check to see if the `int` turn counter variable from Part 1 has reached at least 9 turns AND that the winner is still No one after calling our *WinnerIfAny* function.
    - If yes, this means the game has reached a tie. Set the `char` variable that stores the winner to an `'S'` to signify a Stalemate.
- Output after Part 4 should still look like the output from Part 3. To test Part 4, I recommend making a temporary std::cout line to test your variables that you can comment out later.

Part 5: Loop to allow for multiple turns

- At this point, the first player makes a move and then our game ends. We now want to make sure that we loop over Parts 2 to 4 until the game has ended.
- We will be using the `char` variable that tracks the winner of the game for the condition of our loop. Please implement the logic that our loop will continue as long as the winner is No one.
    - Remember, in Part 4, we determined that there are only 4 possible values for our `char` variable: X, O, N, or S.
- IMPORTANT NOTE: If you want your logic to work correctly for each turn, be sure to RESET/UPDATE your variables at the end of your loop. There are four variables we need to worry about:

- Update the `bool` variable that tracks the player's turns. You will need to switch the value to the other player.
- Reset the `int` user input variable for the player's space choice.
- Reset the `bool` variable that tracks the validity of a player move.
- Update the `int` move counter variable to increment by one move.

- Output after Part 5 should look similar to this: (user input in **red**)

```
 1 | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 1
 X | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 4
 X | 2 | 3
-----------
 O | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 2
 X | X | 3
-----------
 O | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 5
 X | X | 3
-----------
 O | O | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 3
 X | X | X
-----------
 O | O | 6
-----------
 7 | 8 | 9
Press any key to continue...
```

Part 6: Present the winner!

- Now that we have an entire run through of a game, present who wins!
- After our game loop has ended, there are 3 possible values that our `char` variable that tracks the winner could be: X, O, or S. Print out a message depending on what value that variable is.
- Output after Part 6 should look similar to this: (user input in **red**)

```
 1 | 2 | 3
------------
 4 | 5 | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 1
 X | 2 | 3
------------
 4 | 5 | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 4
 X | 2 | 3
------------
 O | 5 | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 2
 X | X | 3
------------
 O | 5 | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 5
 X | X | 3
------------
 O | O | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 3
 X | X | X
------------
 O | O | 6
------------
 7 | 8 | 9
CONGRATULATIONS PLAYER “X”!!! YOU WON!!!
Press any key to continue...
```

Part 7: Loop as long as user wants to play the game

- Lastly, you will implement one more loop to loop ALL of Parts 1 through 6 as long as the user wants to. At the end of the game, ask the user if they would like to play again, and loop again depending on their answer.
- Output after Part 7 should look similar to this: (user input in **red**)

```
 1 | 2 | 3
------------
 4 | 5 | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 1
 X | 2 | 3
------------
 4 | 5 | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 4
 X | 2 | 3
------------
 O | 5 | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 2
 X | X | 3
------------
 O | 5 | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 5
 X | X | 3
------------
 O | O | 6
------------
 7 | 8 | 9
Where do you want to go? Pick an available number: 3
 X | X | X
------------
 O | O | 6
------------
 7 | 8 | 9
CONGRATULATIONS PLAYER “X”!!! YOU WON!!!
```

```
Would you like to player another game? (y/n): y
 1 | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number:

...
```

A Note on Style

Be sure to comment your code.

As we discussed in lecture, it is extremely important that your code is properly indented as it greatly adds to readability. Because of this, if you submit a code file that is not reasonably indented, you will have points deducted.

Likewise, you will lose points if your variable names are not meaningful. Make sure you use variable names that correspond to what you are actually storing in the variables.

Full Sample Output

Below is sample output for a full run-through of the program. User input is in **red**.

```
 1 | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 1
 X | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 4
 X | 2 | 3
-----------
 O | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 2
 X | X | 3
-----------
 O | 5 | 6
-----------
 7 | 8 | 9
```

```
Where do you want to go? Pick an available number: 5
 X | X | 3
-----------
 O | O | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 3
 X | X | X
-----------
 O | O | 6
-----------
 7 | 8 | 9
CONGRATULATIONS PLAYER “X”!!! YOU WON!!!
Would you like to play another game? (y/n): y
 1 | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 1
 X | 2 | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 2
 X | O | 3
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 3
 X | O | X
-----------
 4 | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 4
 X | O | X
-----------
 O | 5 | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 5
 X | O | X
-----------
```

```
 O | X | 6
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 6
 X | O | X
-----------
 O | X | O
-----------
 7 | 8 | 9
Where do you want to go? Pick an available number: 7
 X | O | X
-----------
 O | X | O
-----------
 X | 8 | 9
Where do you want to go? Pick an available number: 8
 X | O | X
-----------
 O | X | O
-----------
 X | O | 9
Where do you want to go? Pick an available number: 9
 X | O | X
-----------
 O | X | O
-----------
 X | O | X
STALEMATE!!!
Would you like to play another game? (y/n): n
Press any key to continue...
```

Deliverables

1. A compressed **Hw05** folder containing both the files named **tictactoe.cpp** and **tictactoe.h**. It must be submitted through Blackboard and should be named **Hw05.zip**.

Grading

Item	Points
Part 1: Variable setup and board display	5
Part 2: User input for player moves	10
Part 3: Display player's move	5
Part 4: Check if there is a winner	5
Part 5: Loop to allow for multiple turns	5
Part 6: Present the winner!	3
Part 7: Loop as long as the players wants to play	2
Total*	**35**

* Points will be deducted for poor code style, or improper submission.