Assignment #5 - Game: Tic-Tac-Toe  
Due: Tuesday, 05/29/12, 11:59pm

(75 pts) You will finish write a program that plays the game Tic-Tac-Toe. In this program, you will be graded on having functions, as well as the ability to play the game correctly. The one requirement for using functions is that your functions, including main(), must not have over 20 lines of code, this doesn’t include comments or blank lines.

Some functions you might want to include are an initialize_board(), which initializes the board to spaces, a determine_player_choice() that allows players to pick their pieces, i.e. ‘X’ or ‘O’, fill_board(), which fills the board with the player’s choice, a print_board() that prints the board to the screen after each user’s turn, is_full() to check if the board is full, a check_for_winner(), which checks to see if there is a winner, and a print_winner_results() that prints the results of the game to the screen.

First, prompt the user to find out if he/she wants to play with one or two players. If the user wants to play with only one player, then the computer must play the one player. You can choose whatever algorithm you want for the computer, i.e. picking random places to put the piece or intelligently selecting your move based on player 1’s selection. However, you mustn’t ever select a position that has already been selected at any time!!!

After you determine how many players are playing the game, then you can determine what character each player wants to choose. You have to make sure that Player 2 or the computer doesn’t choose the same character as Player 1. Print the empty board, and then prompt the player(s) for their position on the board, printing the board after each turn. Make sure that the player(s) chooses valid positions on the board!!!

Example Tic-Tac-Toe:
Do you want 1 or 2 players? 2

Player 1: What character do you want? X
Player 2: What character do you want? O

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Player 1: Where would you like to put your X? 0 0

X |   |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Player 2: Where would you like to put your O? 1 0

X |   |
-----------
O |   |
-----------
    |

Player 1: Where would you like to put your X? 1 1

X |   |
-----------
O | X |
-----------
    |

Player 2: Where would you like to put your O? 2 0

X |   |
-----------
O | X |
-----------
    |

Player 1: Where would you like to put your X? 2 2

X |   |
-----------
O | X |
-----------
    |

Congratulations Player 1, you are a winner!!!

Tic-Tac-Toe Error Handling:
- Player chooses a position that is not on the board, i.e. row 5, column 2.
- Player chooses a position that is already occupied.
- Player 2 chooses Player 1’s character.

Program Input:
- Determine whether the user wants to play 1 or 2 players for Tic-Tac-Toe
- The character/game piece each player wants.
- Player’s choice of position on the board.

Program Output:
- The view of the board after each player’s turn, along with an initial empty board.
- A prompt asking for the player’s selection on the board.
- Appropriate error messages for errors handled above.
(10 pts) In your implementation, make sure that you include a program header in your program, in addition to proper indentation/spacing and other comments! Read the class style guideline for more information: http://classes.engr.oregonstate.edu/eecs/spring2012/cs151-001/151_style_guideline.pdf

You are graded on having a header, proper comments, and readable code with indentation and vertical spacing that is CONSISTENT throughout your program. DO NOT align your entire program on the left side. This will cause you to automatically lose the full 10 points. In addition, do not forget your program header!!!

(15 pts) You are required to turn in a written document (as a pdf) addressing Polya’s steps to solving a problem with step 3 being the C code you write to carry out/implement your plan. With this said, your written document must include these three sections:

**Understanding the Problem**
In your own words, explain what YOU think the problem is asking you to do. In this section, document your uncertainties about the problem and anything else that you feel was unclear or vague. This is to ensure that YOUR understanding matches MY understanding of the problem😊

**Devising a Plan/Design**
At a minimum, provide an algorithm/pseudo code you designed to help solve the problem. In addition, include pictures/flow charts you used to help you devise your plan, as well as any other design decisions you made such as how to manage your time, how to decompose the problem, where to start first, etc. You can scan any handwritten work and attach it to the document as needed.

**Looking Back/Self-Reflection**
Report any checking/self-reflection you did while solving the problem. For instance, how did you make sense of the output from the implementation? This includes things such as using a calculator to make sure the output is correct, testing to make sure your code executes correctly and behaves the way you expect under specific circumstances, using external sources of information such as the internet to make sense of the results, etc. Also, include a statement about what you learned from the assignment.