Assignment #9
2-D Arrays: TicTacToe
Design Due: Sunday, 11/20/16, 11:59pm
Implementation Due: Sunday, 11/27/16, 11:59pm

You will write a program that plays the game **Tic-Tac-Toe**. TicTacToe is a game played on a 3x3 board, where players' alternate turns selecting an empty position on the board to place their game piece (traditionally Xs and Os for game pieces). The first player to get 3 of their own pieces in a row (diagonally, horizontally, or vertically) wins. If the board fills and there are no winners, then the game is tied.

First, prompt the user for the character each player wants to choose. Print the empty board, and then prompt the player(s) for their position on the board, printing the board after each turn. If the player chooses a position on the board that is already occupied, you must prompt them for another position on the board!!! Ask the user if he/she wants to play again, and keep track of the number of wins for each player, as well as the number of ties. Print this information at the end of all the games played, after the user no longer wants to play.

Tic-Tac-Toe Error Handling/Recovery:

- Player's board choice is not appropriate, such as a non-positive int for row/column
- Player chooses a position that is not on the board, i.e. row 5, column 2.
- Player chooses a position on the board that is already occupied.
- Player 2 chooses Player 1's character/game piece.

Requirements (Each is an automatic 10 point deduction!!!):

- You must have a main function and a print board function.
- Your code cannot contain any global variables or statements, except a global call to main.
- Each function with comments including description, parameters, preconditions, postconditions, and returns.

Example Tic-Tac-Toe:

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

    |   |
-------------
    |   |
-------------
    |   |

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

  X |   |
-------------
    |   |
-------------
    |   |
```

```
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 |
-------------
  O |   |


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

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

Congratulations Player 1, you are a winner!!!

Do you want to play again (y-yes, n-no): n

Total Wins:
Player 1: 1
Plater 2: 0
Ties: 0
```

Program Input:
- The character/game piece each player wants.
- Player's choice of position on the board.
- Whether the user wants to play again.

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.
- A prompt asking whether the user wants to continue.
- The total number of wins for both players and the number of ties.

Program Design/Implementation
Begin by designing you program using these steps, and write steps 1, 2, and 4 on paper or in a text editor. Then, implement the program using Python.

- **Step 1: Problem Analysis**. (10 pts)
  a. Comments about the problem to aid in understanding it.
  b. Description of the knowledge base (this list would include what you would be expected to know to follow the solution).
- **Step 2: Program Design**. (30 pts) List the specific steps needed to play the game of TicTacToe. Remember, you have to be very explicit here to make sure the computer can accomplish the task using your directions.
  1. Define all the functions needed.
  2. What are the preconditions, post conditions, and return values for each function?
  3. Describe the specific steps needed in each function, including the main function.
  ….
- **Step 3: Program Implementation** (50 pts). This is the Python code that plays TicTacToe.
- **Step 4: Program Testing**. (10 pts)
  Create a Test Plan with several test cases including the good and bad cases.

(10 pts) Extra Credit 1:

All your functions, including main(), **must not have over 15 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.

(10 pts) Extra Credit 1:

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!!!

Electronically submit your **implementation (.py file)** and **program design (.pdf)** by the assignment due date, using TEACH:
https://secure.engr.oregonstate.edu:8000/teach.php?type=want_auth