CECS 174 – LAB ASSIGNMENT 8

OBJECTIVES:

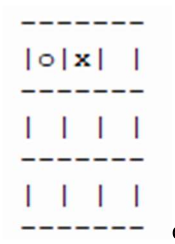
- Able to use a Python 3.x IDE to build Python program(s)
- Implement 2-dimensional list
- Lean about reverse engineering
- Form a sophisticated conditional expression in Python

INSTRUCTIONS:

PART 1 – THIS PART IS WORTH 80%

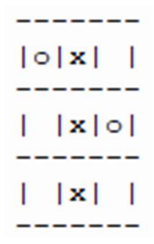
Implement and test 2 functions related to tic-tac-toe game.

1. Write a function that print the board of the tic-tac-toe game, print(board). Board is a 2 dimensional list board[3][3] # 3 rows and 3 columns
2. When this function is called it will print the game board as follows



```
-------
|o|x| |
-------
| | | |
-------
| | | |
-------
```

or



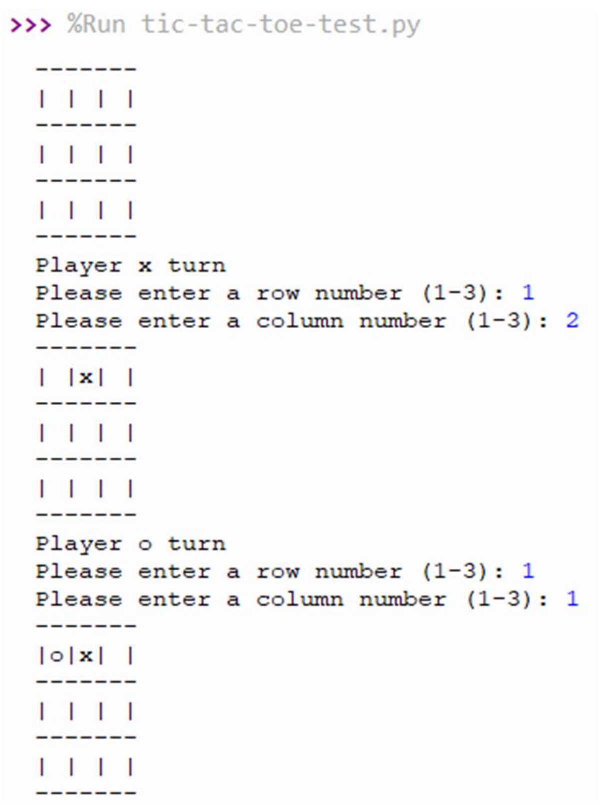
```
-------
|o|x| |
-------
| |x|o|
-------
| |x| |
-------
```

3. Write a function that determines the winning status of a tic-tac-toe game.
   The function is_winner(board, player) will true if the player win. For example is_winner(board, 'x') will return true if player x wins.
   There are 8 scenarios a player can win: 2 diagonal scenarios, 3 cases of wining-by-row and 3 cases of winning-by-column.
4. In the main function, write code to
   a. Set values to the board
   b. Check if a particular player win
   c. Run your app a few times to test at least 4 winning cases and make sure you alternate between player x and o. Also test for not-winning cases as well
5. When you finish testing. Create the pseudocode out from your code. This is called reverse engineering, i.e. write code and test first then abstract the code into pseudocode.

Turn in pseudocode, code, run-time output.

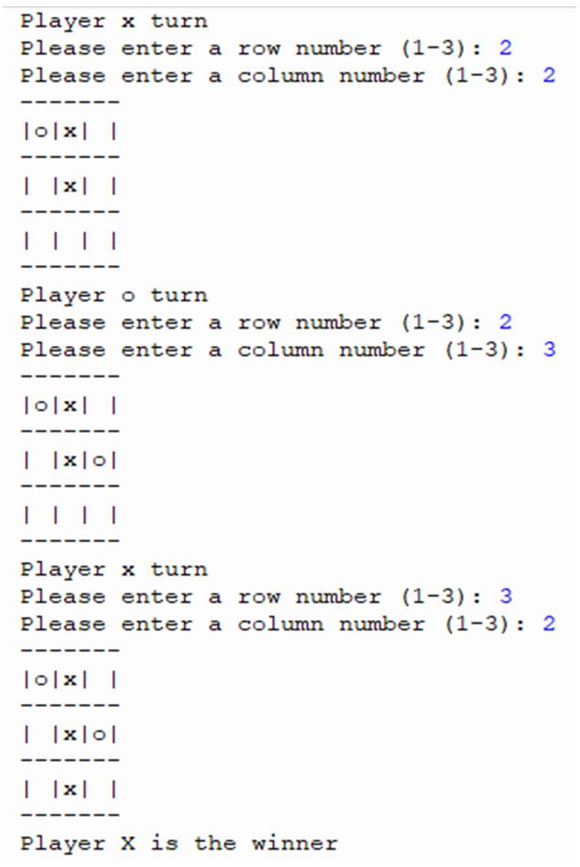
PART 2 – THIS PART IS WORTH 10%

Implement a tictactoe game that allows 2 users to player. Your app should have a main functions and some helper functions. Helper functions are called by main function to do some processing for the main function.
The game can be similar to the following (try to match the display please)



```
>>> %Run tic-tac-toe-test.py

 -------
 | | | |
 -------
 | | | |
 -------
 | | | |
 -------
 Player x turn
 Please enter a row number (1-3): 1
 Please enter a column number (1-3): 2
 -------
 | |x| |
 -------
 | | | |
 -------
 | | | |
 -------
 Player o turn
 Please enter a row number (1-3): 1
 Please enter a column number (1-3): 1
 -------
 |o|x| |
 -------
 | | | |
 -------
 | | | |
 -------
```



```
Player x turn
Please enter a row number (1-3): 2
Please enter a column number (1-3): 2
-------
|o|x| |
-------
| |x| |
-------
| | | |
-------
Player o turn
Please enter a row number (1-3): 2
Please enter a column number (1-3): 3
-------
|o|x| |
-------
| |x|o|
-------
| | | |
-------
Player x turn
Please enter a row number (1-3): 3
Please enter a column number (1-3): 2
-------
|o|x| |
-------
| |x|o|
-------
| |x| |
-------
Player X is the winner
```

HINTS:

- After 9 moves, and no winning, it becomes a tie, (or stale game).
- Player that moves first is the ‘x’ player.
- You are not required to allow users to play multiple games. It is optional though.
- You DON’T have to write pseudocode for this part.

FOR YOUR INFORMATION

Make sure you comment your code. Also turn in an algorithm with pseudocode, and your Python code for part 1. If you write your pseudocode after to write and test-run your program, it is called reverse-engineering. Reverse engineering is a useful engineering process.
You don’t have to turn in pseudocode for part 2. Just code and run-time output.

TURN IN

- Turn in **your pseudocode, Python code and images of your test runs**, (i.e. several runs) in **1 single PDF document**.
- Your turn in document must be in **PDF format.** Upload to the BeachBoard account of this class.