Goals: Learn to solve problem and design solutions. In particular learn about
1. Array data structure
2. Function definition and calls
3. Parameter passing to functions
4. Arithmetic operations and complex computations
5. Creating your own graphic objects.
6. An important addition to this lab is the input validation. If user clicks on the tile already clicked then you should say it is NOT allowed.

Objectives:

Lab 4: Learn to solve problems using modular solution using functional decomposition. Learn to use arithmetic operations and comparisons and nested control structures.

Lab 3: Learn to solve problems by designing algorithmic solutions. Use array data structures, functions and complex logical expressions and arithmetic operations such as modulus. Use switch statement to implement the state machine-defined logic of a game. Use pushMatrix and popMatrix of Processing to layer your sketch.

(Lab 2) Learn to use graphic primitives translate, rotate; control motion was using loop(), noLoop(), frameRate(); Using control statements if..else, loops such as for and while statements.

(Lab 1: Learn to (i) solve problem using Processing language (ii) use Processing Development Environment (PDE) (iii) use 2D primitives, color settings, image commands, mouse operations and the coordinate system of PDE. )

Problem context:

In this lab we will design and implement a solution for the popular game: TIC-TAC-TOE.

Problem Statement:

You are required to design and implement a solution to the TIC-TAC-TOE problem. You will leverage all the array knowledge and the square board knowledge from the Lab3 to benefit this problem. You will have to make /create images for X and O of the game. (If you do not how it is played, ask somebody in the class, we will play it during lecture.) You should be able to play again and again and should keep track of the cumulative score. Eg. Player O won 10 times, Player X won 6 times, draw 10 times etc.

You are required to provide a general solution using variables and parameterized functions wherever appropriate.
Implementation Details:

- Play the game and understand the game.
- Identify the problems to be solved: Here are some the problems to be solved:
  - Setting up the layout (Eg. 400X 400 for the tiles plus the bottom strip; **title can be at the bottom**)
  - When a mouse is clicked on a square make sure it is not already taken; if yes, then indicate an error and let the user select another tile.
  - Mark 0 and X depending on the turn.
  - Check for three in a line or diagonal entry for X or O after at least three entries in both turns (0 or X) is made.
  - Keep track of the end of the game.
  - Keep track winning or draw status.
  - Reset the board after a game is over.
  - Allow repeated play.
- It is very important you develop the solution **in incremental steps**.
- Your program should be modular with coherent functions and parameters to customize the functions.
- You can use println to debug your program.
- Reuse the images you collected in the lab1 and lab2 and any others if you prefer.
- You can use mouse operations such as mouse pressed or mouse released operations.
- Your code should use variables as appropriate and use comments.
- ATTEND recitation class to learn the approaches to solving this problem and data structures and functions needed to solve the problem.

Due date: 11/14/2014 before 6pm