Name: **Session:**

Programming I
Lab Exercise 11.3.2023
Making a PyGame Tic-Tac-Toe Board

In today's activity, you will make a Tic-Tac-Toe board using Pygame.

Your main program should look like this:

```
#initialize pygame
pygame.init()

#set counter to keep track of whose turn
turn = 0

#keep track of state of board - used to check who won
state = [[0,0,0], [0,0,0], [0,0,0]]

#set the drawing screen to 600 x 600
screen = pygame.display.set_mode([600,600])
pygame.display.set_caption("Tic-Tac_Toe")

#fill the screen with white
screen.fill([255, 255, 255])

#draw the lines for the tic-tac-toe board
drawLines()

#Create flag variable for ending game loop
quit = False

#Game loop
while not quit:
   #Check all pygame events
   for event in pygame.event.get():
      #quit event
      if event.type == pygame.QUIT:
         pygame.display.quit()
         sys.exit()
      #KEYDOWN events
      elif event.type == pygame.KEYDOWN
         if event.key == pygame.K_c:        #clear board event
            resetState()
            screen.fill((255, 255, 255))
            drawLines()
            turn = 0
```

```
    elif event.key == pygame.K_s:       #display the state of board
        showState()
#MOUSEBUTTON event
elif event.type == pygame.MOUSEBUTTONDOWN:
    if pygame.mouse.get_pressed() == (1, 0, 0):
        #get position of mouse click and conver to list
        pos = list(pygame.mouse.get_pos())

        #Determine row and column selected
        row = pos[1]//200
        col = pos[0]//200

        #Adjust x position to properly position X or O
        if pos[0] < 200:
            pos[0] = 20
        elif pos[0] < 400:
            pos[0] = 220
        else:
            pos[0] = 420

        #Adjust y position to properly position X or O
        if pos[1] < 200:
            pos[1] = 20
        elif pos[1] < 400:
            pos[1] = 220
        else:
            pos[1] = 420

        #update turn for next player
        turn += 1

        #Player turn
        #draw X or O
        #Update state array
        #Check for win
        if turn%2:
            drawX(pos[0], pos[1])
            state[row][col] = 1
            if checkWin():
                quit = True
        else:
            drawO(pos[0], pos[1])
            state[row][col] = 2
            if checkWin():
                quit = True

        #update display
        pygame.display.flip()
```

```
time.sleep(4)
pygame.display.quit()
```

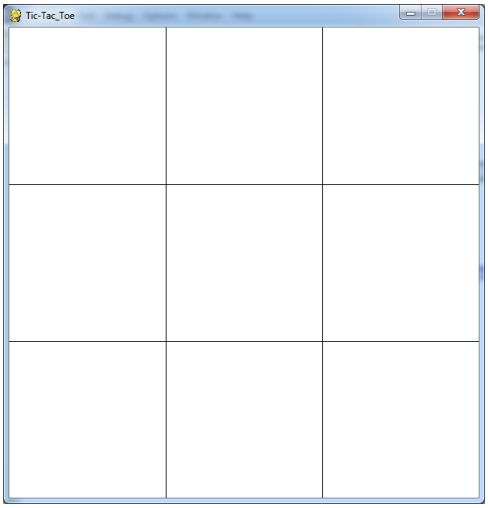
If you examine the starter code you will see that I have given you a function drawX which will draw an X at a specified location. You will need to create a drawO function that will draw an O at a specified location. The code for the drawO function is the same with the exception of the map of 1's and 0's which you will have to modify.

The window is 600 x 600. You will need to draw 4 lines that divide the window into equal thirds. You will then draw X's and O's at alternating locations as shown in the drawing above by calling the drawX(x,y) and drawO(x, y) functions.

In order to draw the lines for the game board, you should write a function drawLines as such:

```
def drawLines():
    #Draw lines for Tic-Tac-Toe board
    pygame.draw.line(screen, [0,0,0],[0,200], [600, 200])
    pygame.draw.line(screen, [0,0,0],[0,400], [600, 400])
    pygame.draw.line(screen, [0,0,0],[200,0], [200, 600])
    pygame.draw.line(screen, [0,0,0],[400,0], [400, 600])
    pygame.display.flip()
```

When you have completed your program, your tic-tac-toe board should look like this.



Once you have tested your program, and you get the above display, you will use the mouse to capture an (x, y) coordinate and draw and X or an O at that location.

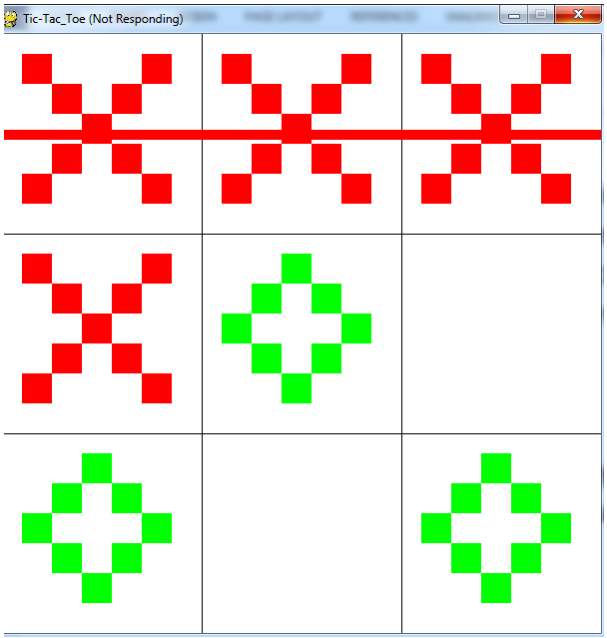
Next we need to determine if someone has won the game. Your program has a list of lists (2D array) to keep track of the state of the board. In order to control things we will write 3 functions showState(), resetState(), and resetGame(). The showState() function is used during the development process for testing purposes. The resetState() function is used to reset the state array to all 0. It is called by resetGame() function. The resetGame() function resets the game to the starting state.

```
def showState():
    #this function is used for testing
    for i in range(3):
        for j in range(3):
            print(state[i][j], end = ' ')
        print()

def resetState():
    for i in range(3):
        for j in range(3):
            state[i][j] = 0

def resetGame():
    resetState()
    screen.fill((255, 255, 255))
    drawLines()
    turn = 0
    quit = False
```

Next we need to determine who has won so we have a checkWin() function. In Tic-Tac-Toe, there are 8 ways to win. We have to check each row, each column and the 2 diagonals. The strategy we will use is that if any 3 cells in a row, column, or diagonal adds up to 3, then player 1 wins. If the sum is 6, then player 2 wins. I have given you the code to test column 1 (actually column 0 in CS terms). Your final working game should look like this:



When you have completed your program, print your source code and a screenshot of your board, attach it to this sheet and turn in.