fullName: _____ andrewID: _____ section: ____ seat#: _____

15-112 S26 Quiz7 Version B
Time: 30 Minutes

You must write your name on this paper and hand this back in immediately after the assessment. If we do not receive it immediately, you will receive a zero on the assessment. Do not unstaple any pages. All pages must be handed in intact.

Do not use your own scrap paper. You should not need it, but if you must absolutely have scrap paper, raise your hand and we will provide some. Write your andrewID clearly on it and hand it in with your quiz. We will not grade anything on scrap paper.

You may not ask questions during the quiz, except for English-language clarification questions. If you are unsure about a problem, take your best guess.

Before and during the quiz, you may not view any other notes, prior work, websites or resources, including any form of AI. You may not use calculators, phones, laptops, or any other devices. You may not communicate with anyone else except for current 112 TAs or faculty during the assessment. All syllabus policies apply.

You may not discuss this test with anyone else, even briefly, in any form, until we have released grades. Failure to abide by these rules may result in an academic integrity violation.

Do not use material we have not yet covered. Thus, do not use sets, dictionaries, OOP, or recursion.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than the specified time noted above on this assessment.

Multiple Choice [5 pts, 1 pt each]

For each of the following variables in the model of the Snake case study, specify whether they should be set in onAppStart or resetApp. Note that while technically every variable *could* be set in resetApp, you should only select resetApp if the variable *must* be set in that function. Indicate your answer by filling in the dot.

A) `app.cols = 10`

◯ onAppStart

◯ resetApp

B) `app.boardLeft = 25`

◯ onAppStart

◯ resetApp

C) `app.direction = (0, -1)`

◯ onAppStart

◯ resetApp

D) `app.paused = True`

◯ onAppStart

◯ resetApp

E) app.stepsPerSecond = 4

◯ onAppStart

◯ resetApp

Fill in the Blank [10 pts, 5 pts each]

1. Fill in the blanks to implement the `drawCellDot` function from the Snake example:

```
# This function draws a dot in the given (row, col) of the board
def drawCellDot(app, row, col, color):
    cellLeft, cellTop = getCellLeftTop(app, row, col)
    cellWidth, cellHeight = getCellSize(app)

    cx = _____________________________

    cy = _____________________________
    r = cellWidth/3
    drawCircle(cx, cy, r, fill=color)
```

2. Fill in the blanks to implement the `pieceIsLegal` function from Tetris. Assume the model contains the variables `app.piece`, `app.pieceTopRow`, and `app.pieceLeftCol`.

```
def pieceIsLegal(app):
    pieceRows, pieceCols = len(app.piece), len(app.piece[0])
    for pieceRow in range(pieceRows):
        for pieceCol in range(pieceCols):
            if app.piece[pieceRow][pieceCol]:

                boardRow = _________________________

                boardCol = _________________________
                if ((boardRow < 0) or (boardRow >= app.rows) or
                    (boardCol < 0) or (boardCol >= app.cols)):
                    # Part of the piece is off the board
                    return False
                if ____________________________:
                    # Part of the piece is over a non-empty cell
                    return False
    return True
```

FR1: Moving Triangle [20 pts]

Write an app such that:

- The app starts by drawing a black triangle with vertices at `(150, 250)`, `(250, 250)`, and `(200, 100)`.
- The entire triangle should continuously move upwards by 1 pixel at a time whenever the up key is held (and the down key is not held). The triangle should not move horizontally.
    - If the entire triangle is offscreen, then the app resets to its initial state.

Write your solution here:

FR2: Counting Dots [30 pts]

Write an app such that:

- The app starts with an empty canvas.
- Each time the user presses the mouse **outside** of a dot:
    - A new cyan dot of radius 20 is added, centered on the mouse press location.
    - Centered on the dot is a counter, at first set to 0.
- Each time the user presses the mouse **inside** a dot:
    - The counter in that dot increases by 1. When that counter would get to 3, the dot (and its counter) is instead deleted.
    - Note that a mouse press may be inside more than one dot. In that case, do this for **all** the dots that the mouse press intersects.

Note: if it helps, you can call distance(x0, y0, x1, y1) without writing it.

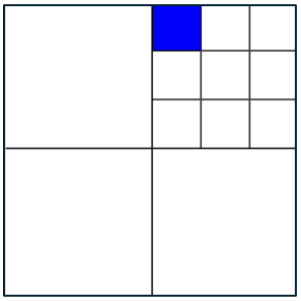
Write your solution on the next page:

FR3: Quadrant Grid [35 pts]

Assume the canvas is square and at least 100x100, but **not of any particular size**

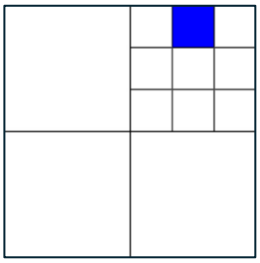
You do **not** need to worry about minor visual details such as border widths for this exercise

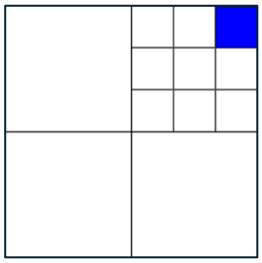
With that, write an app such that:

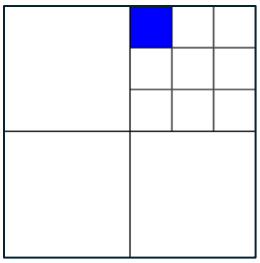


- The app starts by drawing a vertical line (centered horizontally) and a horizontal line (centered vertically) to divide the canvas into 4 quadrants.
- Also, a 3x3 grid is drawn so it fills the entire top-right quadrant of the canvas, as shown to the right.
    - The top-left cell of that 3x3 grid is selected and drawn in blue.
    - The other cells are empty (only their outlines are drawn).

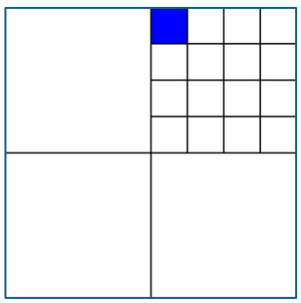
- Each time the user presses the right arrow:
    - The selection moves one cell to the right.
    - If the selection goes beyond the rightmost cell, it wraps around to the leftmost cell in the first row of the 3x3 grid. This wraparound only occurs in the grid within the top-right quadrant.
    - The three images below demonstrate what the canvas should look like after each of the first three right arrow presses:







- Each time the user presses the up arrow:
    - The grid size grows by 1, so 3x3 becomes 4x4, or 4x4 becomes 5x5, etc. There is no upper bound to the grid size. The 4x4 case is shown to the right.
    - Also, the selection resets to the top-left cell of the grid whenever the grid size increases. Note that the grid will only ever be drawn in the top-right quadrant of the canvas.



Write your solution below: