fullName: _____ andrewID: _____ section: ____

15-112 F25 Quiz4 Version B
Time: 35 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 quiz 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 sets, dictionaries, or recursion.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than 35 minutes on this assessment.

Code Tracing [2 pts each]

Indicate what the following code prints. Place your answer (and nothing else) in the boxes below. If a line of code crashes, just print "crash" (without quotes) and stop the CT at that point.

CT1:

```
import math
# getRadiusEndpoint is taken verbatim from the notes:
def getRadiusEndpoint(cx, cy, r, theta):
    return (cx + r*math.cos(math.radians(theta)),
            cy - r*math.sin(math.radians(theta)))

def ct1():
    L = getRadiusEndpoint(50, 60, 80, 180)
    return [round(v) for v in L]
print(ct1())
```

CT2:

```
import math
# distance is taken verbatim from the notes:
def distance(x0, y0, x1, y1):
    return ((x1 - x0)**2 + (y1 - y0)**2)**0.5

# getRadiusAndAngleToEndpoint is taken verbatim from the notes:
def getRadiusAndAngleToEndpoint(cx, cy, targetX, targetY):
    radius = distance(cx, cy, targetX, targetY)
    angle = math.degrees(math.atan2(cy-targetY, targetX-cx)) % 360
    return (radius, angle)

def ct2():
    L = getRadiusAndAngleToEndpoint(60, 80, 60, 60)
    return [round(v) for v in L]

print(ct2())
```

Fill in the Blank (FitB) [2.5 pts each]

FitB1:

```
# This should set app.color to 'blue' whenever any key is released:

def ____________________________________________________________:
    app.color = 'blue'
```

FitB2:

```
# Fill in the blank from the snake example:
def takeStep(app):
    # Find the new position of the snake's head:
    headRow, headCol = app.snake[0]
    drow, dcol = app.direction
    newHeadRow = headRow + drow
    newHeadCol = headCol + dcol
    # And move the snake there if we we can:
    if ((newHeadRow < 0) or (newHeadRow >= app.rows) or
        (newHeadCol < 0) or (newHeadCol >= app.cols)):
        # ran off the board
        app.gameOver = True

    elif ____________________________________________________________:
        # ran into itself
        app.gameOver = True
    else:
        ...
```

FitB3:

```
# This should draw a polygon using the points in a 1D list L of
# unknown length. For example, but not necessarily, L might be
# [100, 100, 50, 200, 300, 300, 250, 50]:

drawPolygon(_______________________________, fill='cyan', border='black')
```

FitB4:

```
# This should increase app.counter by 1 whenever either the 'x' key
# or the 'y' key is held down (or both).  Ignore any other keys.

def onKeyHold(app, keys):

    if ____________________________________________________________:
        app.counter += 1
```

Multiple Choice: [2 pts each] Place an X in the box with the correct answer.

MC1:
In the note on drawing a 2d board, why did we need the function drawBoardBorder(app)?

☐ A) Without this function, the border around the board was not drawn at all.

☐ B) Without this function, there were no borders around any cells in the board.

☐ C) Without this function, the border around the board was half as thick as the interior borders.

☐ D) We did not need or use this function.

☐ E) None of the above.

MC2:
In the note on cell selection, how and why did we need the function getCell(app, x, y)?

☐ A) This function is called by onKeyPress to determine which cell to select.

☐ B) This function is called by onMousePress to determine which cell to select.

☐ C) This function is called by onStep to determine which cell to select.

☐ D) This function is called by redrawAll to determine which cell to draw as selected.

☐ E) None of the above.

MC3:
In getCell(app, x, y), why did we use math.floor(dy / cellHeight) instead of (dy // cellHeight)?

☐ A) We need the floor, but (dy // cellHeight) rounds.

☐ B) If dy is a non-number, math.floor(dy / cellHeight) will not crash but (dy // cellHeight) will crash.

☐ C) We did not need or use math.floor here.

☐ D) If dy is a float then (dy // cellHeight) will be a float.

☐ E) None of the above.

FR: Even and Odd Dots Animation [80 pts]

For this animation, you must represent each dot as a list of 3 values: [cx, cy, counter].

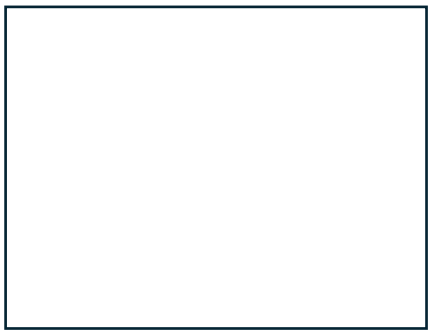
With that, write an animation where:

- At first, the canvas is empty.
- When the mouse is pressed outside of any dots, a new yellow dot is added centered on the mouse press with radius 20. The dot contains a counter that is drawn in black, centered on the dot, and is initially 0.
- When the mouse is pressed inside one or more dots, the pressed dots' counters each increase by 1.
- While the 'e' key is held down, all the dots with even counters move up by 5 pixels.
- While the 'o' key is held down, all the dots with odd counters move up by 5 pixels.
- After any dots are created, or after dots are moved, then do these in this order:
    - First, if any part of a dot would extend beyond the top of the canvas, that dot is deleted.
    - Next, find all the dots that intersect where one's counter is even and the other's counter is odd, and then delete all such dots.

Note: you can use the distance() function without writing it here.

Write your solution on the following pages.

BonusCT [2pts]: This CT is optional, and intended to be very challenging. It is worth very few points. Draw a picture of what this app will draw when it runs. Place your answer (and nothing else) in the box.



```
def redrawAll(app):
    a, b, c, d, e = 100, 145, 16, 200, 116
    for x0 in range(a, b, c):
        x, y, dy = x0, d, 0
        for t in range(d):
            if x == e: dy = 30
            else: drawCircle(x, y, 2)
            if 0 <= x-a <= 32:
                drawCircle(x, 260 - abs(x - e)/2, 2)
            x = x0 + math.cos(t) * 10
            y = d + dy - math.sin(t) * 20

runApp()
```