fullName: _____ andrewID: _____ section: ____

15-112 F25 Practice Quiz4
Time: 30 Minutes

This is the Practice Quiz4, part of Prep9. In order to receive credit, you must simulate taking this as if it was an actual quiz, and you must write your name on this paper and hand this back in during lecture on 10/28 or prior to then online using this form. This Practice Quiz is graded based on completion rather than correctness, but if we do not receive it at the prescribed time, or if you have not demonstrated apparent effort, you will receive a zero on the assessment.

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

Do not use sets, dictionaries, recursion, OOP or anything else not yet covered in the course.

Do not open this or look inside (even briefly) before you are ready to begin. Do not spend more than 30 minutes on this assessment.

Code Tracing

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:

```
# getRadiusEndpoint is taken verbatim from the notes:
import math
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(60, 70, 80, 90)
    return [round(v) for v in L]
print(ct1())
```

CT2:

```
# 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, 70, 40, 70)
    return [round(v) for v in L]
```

Fill in the Blank (FitB)

FitB1:

```
# 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')
```

FitB2:

```
# This should increase app.counter by 1 whenever the 'x' key is held down:

def onKeyHold(app, keys):

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

FitB3:

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

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

FitB4:

```
# 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:
        ...
```

Multiple Choice: 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 onMousePress to determine which cell to select.

☐ B) This function is called by onKeyPress 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: Growing Rects Animation

Note: for this animation, you must represent each rectangle as a list of 5 values: [left, top, width, height, counter]. We recommend creating each rect like so:

```
rect = [left, top, width, height, counter]
```

Then you can access parts of a rect like so:

```
rect[1] += 5 # Add 5 to the rect's top
```

With that, write an animation where:

* At first, the canvas is empty.
* The canvas will eventually contain rectangles where:
  * Rectangles have no fill.
  * Rectangles have a border that is black, except that one rectangle may be selected, and if so, that rectangle has a border that is red.
  * Each rectangle contains a counter that is drawn centered in the rectangle.
  * At any time, if a rectangle extends at all beyond any edge of the canvas, the rectangle is immediately deleted.
* Each time the user presses the mouse, if it is not inside a rectangle, then a new 20x20 rectangle is drawn centered on the mouse press. The new rectangle's counter starts at 1, and the new rectangle is selected, so its border is red. Note that if any part of this rectangle would extend beyond any edge of the canvas, then the rectangle is not drawn and the mouse press is ignored.
* Each time the user presses the mouse inside a rectangle, then that rectangle becomes selected (and any rectangle that was selected is de-selected), and its counter increases by 1.
* If the user presses the mouse over overlapping rectangles, then only the most-recently-created rectangle is the one that is selected and its counter is incremented.
* In onKeyHold, if there is a selected rectangle, then:
  * When the user holds down the 'w' key, the selected rectangle's width increases by 2 while the rectangle's center is unchanged.
  * When the user holds down the 'h' key, the selected rectangle's height increases by 2 while the rectangle's center is unchanged.
  * However, when the user holds down BOTH the 'w' and the 'h' key at the same time, then nothing happens (the width and height remain unchanged).
  * All other keys are ignored. For example, if the user holds down both 'w' and 'z', the 'z' is ignored and the selected rectangle's width increases.

Write your solution on the following pages.