Name:___________________________________ Section:___ Andrew Id: ____________________

15-110 Spring 2019 Quiz4

*** 30 minutes, No calculators, no notes, no books, no computers, no phones**
* Do not assume the size of the canvas. Use data.width and data.height.
* Make reasonable assumptions about anything not explicitly stated here.
* To save time, you may write d.foo instead of data.foo, and e.foo instead of event.foo

1. **Code Tracing [5 pts]** Indicate what the following program prints. Place your answer in the box.

```
def ct1(L):
    rows = len(L)
    cols = len(L[0])
    for row in range(rows):
        for col in range(cols):
            try:
                L[row][col] += L[row+1][col]
            except:
                L[row][col] *= 3

L = [ [1, 'A'],
      [6, 'B']
    ]
ct1(L)
print(L)
```

2. **Code Tracing [5 pts]** Indicate what the following program prints. Place your answer in the box.

```
def ct2(n):
    s = ''
    for i in range(n):
        s += str(i) + ':'
        for j in range(n-i):
            s += str(j)
        s += '\n' # newline
    return s
print(ct2(3))
```

3. **Short Answers [20 pts; 4 pts each value]**

1. Which of the following functions is also known as (aka) viewToModel:

A) getCellBounds B) getCell C) drawAll D) init E) none of these

2. Fill in the blank with the first line from getCellBounds that sets x0 of the cell:

```
def getCellBounds(boardGame, row, col):

    x0 = ________________________________________________________________________
```

3. Fill in the blank (from board-game-basic-example.py):

```
def mousePressed(event, data):

    cell = getCell(data.boardGame, event.x, event.y)

    if (__________________________________________________________________________):

        cell.count += 1
```

4. Fill in the blank (from makeTranspose.py):

```
def makeTranspose(L):

    # Take a rectangular 2d list L and return its transpose, swapping rows and cols

    oldRows = len(L)

    oldCols = len(L[0])

    transpose = make2dList(oldCols, oldRows)

    for oldRow in range(oldRows):

        for oldCol in range(oldCols):

            transpose[oldCol][oldRow] =_____________________________________________

    return transpose
```

5. Fill in the blank (from snake.py):

```
def moveSnake(data):

    newHead = makeNewHead(data)

    if ((newHead.row < 0) or (newHead.row >= data.boardGame.rows) or

        (newHead.col < 0) or (newHead.col >= data.boardGame.cols) or

        (snakeContainsCell(newHead, data) == True)):

        data.gameOver = True

    else:

        if ((newHead.row == data.foodCell.row) and

            (newHead.col == data.foodCell.col)):

            data.snake = [ newHead ] + data.snake # grow snake by 1

            data.foodCell = getRandomFoodCell(data)

        else:

            data.snake = __________________________________________ # remove old tail

        if (len(data.snake) == data.boardGame.rows * data.boardGame.cols):

            # Wow, they won!

            data.gameOver = True
```

4. **Free Response: trialSucceeds for oddsOfTie(tosses) [30 pts]**
Using Monte Carlo methods as we learned in class (even if you know how to solve it some other way), here is the function oddsOfTie(tosses) that takes a positive integer, which is the number of tosses, and returns the odds that after that many tosses of a coin we get exactly as many heads as tails (hence, a tie):

```
def oddsOfTie(tosses):
    successes = 0
    trials = 10000
    for trial in range(trials):
        if (trialSucceeds(tosses) == True):
            successes += 1
    return successes/trials
```

You need to write the helper function that this calls. That is, write trialSucceeds(tosses):

5. **Free Response: red-blue 2d board "game" [40 pts]**
   Starting from board-game-starter-code.py and properly using our 2d Board Game framework (including proper use of getCell and getCellBounds), write init, mousePressed, and drawAll so that:

   a. The app draws a 20x20 board (with a 5-pixel margin around it) of red cells.
      ***** Do not assume the size of the canvas. Use data.width and data.height.
   b. Each time the user clicks the mouse in a cell, if the cell is red it turns blue, and if it is blue it turns red.

6. **Bonus Code Tracing [2 pts]** Indicate what the following program prints. Place your answer in the box.

```
def bonusCt1(L,d):
    result = ''
    for v in L:
        if (type(v) == list):
            v = bonusCt1(v,d+1)
        elif (d%2 == 1): v='.'
        result += str(v)
    return result
print(bonusCt1([ 1, [ [ 2 ], 3, [ 4, [[5],
                 6,7], 8 ] ], 9 ], 0))
```

7. **Bonus Code Tracing [2 pts]** Indicate what the following program prints. Place your answer in the box.

```
def bonusCt2(s):
    for n in range(10**5):
        try:
            s += ',' + str(n) + ']'
            return eval(s)[-1]
        except:
            s += ']'
s = '[ 1, [ [ 2 ], 3, [[[ 4, [[5'
print(bonusCt2(s))
```