Name:___________________________________ Section:___ Andrew Id: _____________________

15-110 Spring 2019 Quiz4b

*** 35 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:
                d = L[row+1][col+1]
                L[row][col] = L[row][col] // d
            except:
                L[row][col] = col

L = [ [24, 12, 6],
      [ 4,  3, 2]
    ]
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):
        for j in range(n):
            s += str(i) + str(j)
    return s
print(ct2(2))
```

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: snakeEyes [30 pts]**
   Using Monte Carlo methods as we learned in class (even if you know how to solve it some other way), write all the code necessary to confirm that the probability of rolling "snake eyes" (two 1's) when you roll two six-sided dice is about 2.8%. Again, you must use our standard Monte Carlo methods here. Also, use at least 10,000 trials. Hint: this requires that you write two functions: probabilityOfSnakeEyes (which you can abbreviate simply as p) and trialSucceeds.

5. **Free Response: column-clearer 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, keyPressed, and drawAll so that:

   a. The app draws a board with 10 rows and 10 columns (with a 5-pixel margin around it) of white cells.
   ***** Do not assume the size of the canvas. Use data.width and data.height.
   b. Each cell starts with the value 0 inside it.
   c. Each time the user clicks the mouse in a cell, the value in that cell increases by 1.
   d. When the user presses a digit key, add 10 to each value in the column corresponding to that key. So:
   * When the user presses '0', add 10 to each value in column 0.
   * ...
   * When the user presses '9', add 10 to each value in column 9.
   Hint: confirm that event.key is a digit using isdigit(), then if it is, convert event.key into a number using int().

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

```
def bonusCt1(L):
    try:
        L = [len(L), len(L[0])]
        return L + bonusCt1(L)
    except:
        return L
print(bonusCt1([[1,2,3],[1,2],[1],0]))
```

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

```
def bonusCt2(L):
    try:
        i,x = 0,0
        while True:
            x = L[i%len(L)].pop(x)
            i += x
    except:
        s = set()
        for row in L: s = s.union(set(row))
        return sorted({1,2,3,4,5,6,7,8,9} - s)
print(bonusCt2([ [1,2], [3,2], [4,5,6,7], [8,9,3] ]))
```