fullName:____________________________andrewID:____________________________ recitationLetter:______

15-112 S24

WS8+Quiz8 version B

You **MUST** stop writing and hand in this **entire** quiz when instructed in lecture.

- You may not unstaple any pages.
- Failure to hand in an intact quiz will be considered cheating. Discussing the quiz with anyone in any way, even briefly, is cheating. (You may discuss it only once the quiz has been posted to the course website.)
- You may not use your own scrap paper. If you must use additional scrap paper, raise your hand and we will provide some. You must hand any scrap paper in with your paper quiz, and we will not grade it.
- You may not ask questions during the quiz, except for English-language clarifications. If you are unsure how to interpret a problem, take your best guess.
- You may not use any concepts (including builtin functions) which we have not covered in the notes in weeks 1-8.
- You may not use dictionaries, sets, or recursion.
- We may test your code using additional test cases.
- Assume almostEqual(x, y) and rounded(n) are both supplied for you. You must write all other helper functions you wish to use, unless we specify otherwise.

Writing Session 8 (10% of HW8)

WS1. mutatingInsertRowAndCol

Here is the start to the hw8 writeup for mutatingInsertRowAndCol:

Write the function `mutatingInsertRowAndCol(L, row, col, val)` which takes a rectangular 2d list L, and inserts one new row and one new column at the locations specified by `row` and `col`. The cells in the new row and column are all set to `val`.

With that in mind, fill in the blanks with the missing code to make this function work properly. You may not insert multiple lines in one blank:

```
def mutatingInsertRowAndCol(L, row, col, val):
    cols = len(L[0])


    L.insert(row, _______________________________________ )  # <-- fill in this blank
    for rowList in L:


        ________________________________________________      # <-- fill in this blank
```

WS2. rotate2dListClockwise (from Tetris Step 4)

Fill in the blanks with the missing code to make this function work properly:

```
def rotate2dListClockwise(L):
    oldRows, oldCols = len(L), len(L[0])
    newRows, newCols = oldCols, oldRows
    M = [([None] * newCols) for row in range(newRows)]
    for oldRow in range(oldRows):
        for oldCol in range(oldCols):


            newRow = __________________________________________ # <-- fill in this blank


            newCol = __________________________________________ # <-- fill in this blank
            M[newRow][newCol] = L[oldRow][oldCol]
    return M
```

Quiz8

CT1: Code Tracing [16pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```
def ct1(L):
    M = copy.copy(L)
    N = copy.deepcopy(L)
    L += [1]
    M[0] += [2]
    L = L[::-1]
    N += [[3]]
    print(M)
    print(N)
    return L

L = [[4]]
print(ct1(L))
print(L)
```

CT2: Code Tracing [12pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below.

```python
# Hint for ct2: before you trace each line, take a moment to think
# about what this code is doing in general.

def ct2(L, M):
    for x,y in L:
        for i in range(len(M)):
            for j in range(len(M[i])):
                if M[i][j] == x:
                    ct2Helper(M, i, j, y)

def ct2Helper(L, r, c, v):
    rows, cols = len(L), len(L[0])
    for drow, dcol in [[-1, 0], [+1, 0], [0, -1], [0, +1]]:
        row = r + drow
        col = c + dcol
        if ((row >= 0) and (row < rows) and (col >= 0) and (col < cols)):
            L[row][col] += v

M = [[21, 22, 23],
     [24, 25, 26]]
L = [[22, 10], [25, 20], [24, 30]]
ct2(L, M)
for v in M: print(v)
```

Note: You may use this empty page for any scrap work, but nothing on this page will be graded for any problem.

Free Response 1: rowColMax(L) [30pts]

Write the mutating function `rowColMax(L)` that takes a non-empty rectangular 2d list L of integers, and mutates L so that each value is set to the max of any value in the same row or same column as that value. For example:

```
L = [[2, 4, 7],
     [1, 0, 5],
     [6, 3, 8]]
```

When we call rowColMax(L), it returns None, because it is mutating:

```
assert(rowColMax(L) == None)
```

But then L was modified as noted, so:

```
assert(L == [[7, 7, 8],
             [6, 5, 8],
             [8, 8, 8]])
```

For example, consider the middle value (at row 1, col 1). This was 0 to start, but it is set to the largest value in row 1 ([1, 0, 5]) and column 1 ([4, 0, 3]). The largest of these values is 5, so the middle value of L becomes 5.

```
def testRowColMax():
    print('Testing rowColMax()...', end='')
    L = [[2, 4, 7],
         [1, 0, 5],
         [6, 3, 8]]
    assert(rowColMax(L) == None)
    assert(L == [[7, 7, 8],
                 [6, 5, 8],
                 [8, 8, 8]])
    # Remember, we will consider additional test
    #   cases when grading your code
```

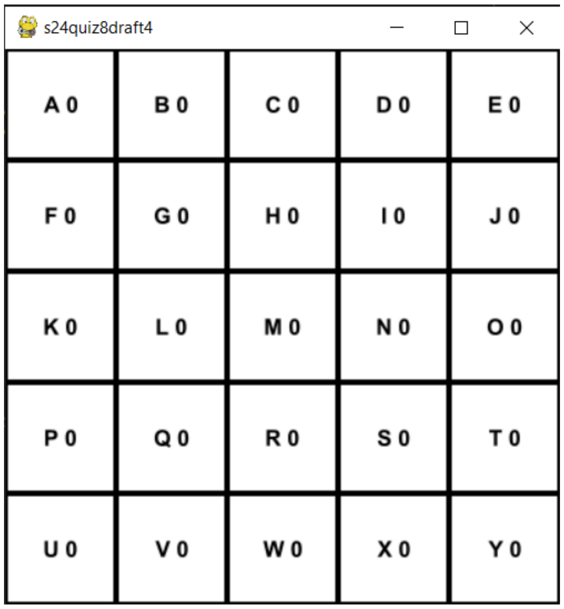
Begin your FR1 answer on the following page:

Write your FR1 answer here:

Free Response 2: Animation [42pts]

Write an animation such that:

- When started, the app draws a 400x400 canvas that looks like this:



A 0	B 0	C 0	D 0	E 0
F 0	G 0	H 0	I 0	J 0
K 0	L 0	M 0	N 0	O 0
P 0	Q 0	R 0	S 0	T 0
U 0	V 0	W 0	X 0	Y 0

  - This is a 5x5 grid of squares
  - Each square contains a letter followed by a count (a number)
  - The letters are A to Y, in order, where the first row in A,B,C,D,E, and the second row is F,G,H,I,J, and so on.
  - The counts are all 0 to start
- When the user presses a letter key (using the keyboard not the mouse), that letter's count increases by 1.
  This is case-insensitive, so both 'a' or 'A' increase the count in the top-left cell by 1.
- When the user presses 'z' or 'Z', that zeroes out the board, so every count is set to 0 (as when the app started).
- When the user presses the mouse in a cell, that cell's count increases by 1.
- Also: be sure not to have any MVC Violations!

Note that we have provided function headers for the necessary cmu_graphics functions, but you must write any additional helper functions you wish to call (except almostEqual and rounded, which are always provided, but may or may not be relevant to this problem).

Begin your FR2 answer on the following page.

Begin your FR2 answer here:

```
from cmu_graphics import *

def onAppStart(app):









def redrawAll(app):
```

Continue your FR2 answer here:

```
def onMousePress(app, mouseX, mouseY):
```

```
def onKeyPress(app, key):
```

Continue your FR2 answer here:

```
def main():
    runApp()

main()
```

The problems below are not required. Indicate what the following code prints. Place your answers (and nothing else) in the boxes below.

bonusCt1 [optional, 1pt]

```
def bonusCt1(M):
    z = 0
    while M:
        z += max(M.pop()) ** min(M.pop(0))
    N = [ ]
    while z:
        N.insert(0, str(z%2))
        z//=2
    return ''.join(N)
M = [[3, 8, 1, 3, 4],
     [6, 5, 2],
     [9, 7],
     [-1, -2]]
print(bonusCt1(M))
```

bonusCt2 [optional, 1pt]

```
def bonusCt2(M):
    def s(L): return sum(sum(z) for z in L)
    L = [ ]
    while M:
        N = [Q[1:-1] for Q in M[1:-1]]
        L.append(s(M) - s(N))
        M = N
    return L
print(bonusCt2([list(range(i, i+5)) for i in range(5)]))
```