fullName:___________________________andrewID:___________________________
recitationLetter:_____

submissionCode:________________

Quiz 5

You **MUST** hand in this **entire** quiz when instructed in lecture. Failure to do so will be considered cheating.

Part 1: Code Tracing

You may not run any code in Part 1 or Part 2 at any time. Note that there are **two** FRs. Save time for them.

CT1: Code Tracing [15pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below. Note that you may not run this code.

```
#prints 2 lists containing lists
import copy
def ct1(L):
    a = L
    b = copy.copy(L)
    c = copy.deepcopy(L)
    b[0] = a[1] * a[1][0]
    a[0][0] += a.pop()[0]
    b[1] = c[0]
    return  b
# Be careful  to  get the brackets  and commas  right!
L = [[1],[2],[3]]
print(ct1(L))
print(L)
```

CT2: Code Tracing [15pts]

Indicate what the following code prints. Place your answers (and nothing else) in the box below. Note that you may not run this code.

```
# Prints 1 list which may contain anything
def ct2():
    a = []
    for i in range(1,4):
        b = [i]
        for j in range(i):
            if j % 2 == 0:
                b.append(j)
            else:
                a.append(j)
        a.append(b)
    return a
print(ct2())
```

Part 2

Note: Part 3 is a (very challenging) bonus code tracing problem worth 2 points. Once you move on, you may not return to Part 1 or Part 2.

Free Response 1: wordSearch [25pts]

We have provided the outer and inner functions from the word search case study. Write the missing middle helper function wordSearchFromCell so that wordSearch(board, row) works properly.

Remember that wordSearch(board, word) takes board (a 2d list of lowercase letters) and word (a non-empty string of lowercase letters) and returns a tuple that contains the word, (startRow, startCol) as a tuple, and a string describing the direction. We may test your code using cases not shown here. You must not modify the helper functions we have provided.

This problem is worth relatively few points, so you should not spend all your time on it. Note that as long as your code is functional without modifying wordSearch or wordSearchFromCellInDirection, it is fine if your variable names and code do not perfectly match ours. Do not hardcode.

```
#Do not modify this function!
def wordSearch(board, word):
    (rows, cols) = (len(board), len(board[0]))
    for row in range(rows):
        for col in range(cols):
            result = wordSearchFromCell(board, word, row, col)
            if (result != None):
                return result
    return None

#Write the missing helper function below the supplied test cases
def wordSearchFromCell(                              ):
    return 42

#Do not modify this function!
def wordSearchFromCellInDirection(board, word, startRow, startCol, drow, d
    (rows, cols) = (len(board), len(board[0]))
    dirNames = [ ["up-left"  ,   "up", "up-right"],
                 ["left"     ,   ""  , "right"   ],
                 ["down-left", "down", "down-right" ] ]
    for i in range(len(word)):
        row = startRow + i*drow
        col = startCol + i*dcol
        if ((row < 0) or (row >= rows) or
            (col < 0) or (col >= cols) or
            (board[row][col] != word[i])):
            return None
    return (word, (startRow, startCol), dirNames[drow+1][dcol+1])

def testWordSearch():
    print('Testing wordSearch(board, row)...', end='')
    board = [ [ 'd', 'o', 'g' ],
              [ 't', 'a', 'c' ],
              [ 'o', 'a', 't' ],
              [ 'u', 'r', 'k' ],
            ]
    assert(wordSearch(board, "dog") == ('dog', (0, 0), 'right'))
    assert(wordSearch(board, "cat") == ('cat', (1, 2), 'left'))
    assert(wordSearch(board, "tad") == ('tad', (2, 2), 'up-left'))
    assert(wordSearch(board, "cow") == None)
    print('Passed!')
testWordSearch()
```

Free Response 2: zeroRectCount(L) [45pts]

Background: given a 2d list of integers L, we will say that a rectangular region of L is a "zeroRect" (a coined term) if the sum of the values in that region equals 0. For example, consider this list:

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

Here are the rectangular regions of L that sum to 0:

```
R1 = [ [ 1,  2, -3  ] ]   # 1x3 in  top-left  of  L

R2 = [ [ 1,  2 ],              # 2x2 in  top-left  of  L
       [ 3, -6  ] ]

R3 = [ [ 0 ] ]                   # 1x1 near  bottom-right  of  L
```

With this in mind, write the function zeroRectCount(L) that takes a rectangular 2d list of integers L, and returns the total number of zeroRects in L. For example, with L as above, zeroRectCount(L) returns 3.

Hint: while you may solve this any way you wish, our sample solution used a large number of nested 'for' loops to try all possible rectangles (so don't be discouraged if your solution does so as well).

You may not import or use any module other than copy. You may not use any method, function, or concept that we have not covered this semester. We may use additional test cases not shown here. Do not hardcode.

```
# Note: almostEqual(x, y) and roundHalfUp(n) are both supplied for you.
# You must write all other helper functions you wish to use.
def almostEqual(d1, d2, epsilon=10**-7): #helper-fn
    return (abs(d2 - d1) < epsilon)

import decimal
def roundHalfUp(d): #helper-fn
    # Round to nearest with ties going away from zero.
    rounding = decimal.ROUND_HALF_UP
    return int(decimal.Decimal(d).to_integral_value(rounding=rounding))

import copy
#---------------
```

```
#Write your function below the supplied test cases
def zeroRectCount(L):
    return 42

def testZeroRectCount():
    print('Testing zeroRectCount(L)...', end='')
    L = [[42]]
    assert(zeroRectCount(L) == 0)
    L = [[0]]
    assert(zeroRectCount(L) == 1)
    L = [[-3, -1, 2, 3]]
    assert(zeroRectCount(L) == 0)
    L = [[-3, -1, 1, 3]]
    assert(zeroRectCount(L) == 2)

    L = [ [ 1,  2, -3,  5,  1 ],
          [ 3, -6,  4,  0,  1 ] ]
    assert(zeroRectCount(L) == 3)

    L = [ [ 1,  2, -3,  5],
          [ 3, -6,  4,  0],
          [-4,  6,  1, -9] ]
    assert(zeroRectCount(L) == 7)
    print('Passed!')

testZeroRectCount()
```

Bonus Part 3

This part is optional. You may not return to Parts 1 or 2.

bonusCT1: Code Tracing [2pts]

This question is optional. Indicate what the following code prints. Place your answers (and nothing else) in the box below. Note that you may not run this code.

```
def bonusCt1(n, b):
    while (b[-1]**0.5 < 1+2+3):
        n += 1;
        b = [sum([list(range(k))
                  for k in range(n)][i][:i-1])
                  for i in range(n)]
    return b[-3]
print(bonusCt1(10,  [1]))
```