Name:_____ Section:___ Andrew Id: _____

15-112 Fall 2015 Quiz 6x (a)

*** Up to 30 minutes. No calculators, no notes, no books, no computers.**

*** No recursion! * To receive credit (in Code Tracing), show your work.**

1. **Code Tracing** [20 pts]:Indicate what this prints. Place your answer (and nothing else) in the box below the code.

```
def ct1(L):
    L.append(list())
    while (len(L) > 2): L[-1].append(L.pop(1) * 2)
    return L[-1] * 2
L = [ 5, 6 ]
print(ct1(L))
print(L)
```

```
import copy
def f(a):
    return 10*a[0][0]+a[1][0]

def ct2(a):
    b = copy.copy(a)
    c = copy.deepcopy(a)
    d = a
    e = a[0:len(a)]
    c[0][0] = 1
    d[0] = [2]
    e[1] = [3]
    b[0][0] = 4
    print(f(b), f(c), f(d), f(e)) # f is defined above
a = [[5], [6]]
ct2(a)
print(f(a)) # don't miss this
```

2. **Reasoning Over Code [20 pts]:** For each of the following functions, find values of the arguments that will make the function return True. You only need one set of arguments for each function, even if there are multiple correct answers. Show your work. Place your answer (and nothing else) in the box below the code.

```
def rc1(L):
    assert(sorted(L) == sorted(range(len(L))))
    result = [ ]
    for i in range(len(L)):
        m = L[i]
        for j in range(i, len(L)):
            if (L[j] > m):
                m = L[j]
        result.append(m)
    return (result == [3, 3, 2, 1])
```

L =

```
def rc2(L):
    result = 0
    if (len(L[0]) == 3):
        for i in range(len(L)):
            for j in range(len(L[0])):
                k = L[i][len(L[0])-1-j]
                if (0 <= k <= 9):
                    result = 10*result + k
    return (result == 234567)
```

L =

3. **Fill in the blank:** [30 pts; 10 pts each]
You are provided with near-complete solutions to some of the functions from the course notes. Fill in the blanks with the missing code so these functions work correctly.

3A: from wordSearch

```
def wordSearchFromCellInDirection(board, word, startRow, startCol, drow, dcol):
    (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

            ____________________________________________________________________):
            return None
    return (word, (startRow, startCol), dirNames[drow+1][dcol+1])
```

3B: from Connect4

```
def checkForWin(board, player):
    winningWord = player * 4

    return ____________________________________________________________________
```

3C: from Othello

```
def makeMoveInDirection(board, player, startRow, startCol, dir):
    (rows, cols) = (len(board), len(board[0]))
    dirs = [ (-1, -1), (-1, 0), (-1, +1),
             ( 0, -1),          ( 0, +1),
             (+1, -1), (+1, 0), (+1, +1) ]
    (drow,dcol) = dirs[dir]
    i = 1
    while True:
        row = startRow + i*drow
        col = startCol + i*dcol
        if (board[row][col] == player):
            # we found the other side of the 'sandwich'
            break
        else:
            # we found more 'meat' in the sandwich, so flip it!

            ____________________________________________________________________
            i += 1
```

4. **Free Response**: **medians(L)** [30 pts]
Background: the median of a list is the middle value (if the list was sorted), or if the list has even length, the average of the two middle values. For example, the median of [3,1,2] is 2 and the median of [6,1] is 3.5. With this in mind, write the function medians(L) that takes a value L, and if it is a possibly-ragged 2d list of numbers, returns a list of the medians of each list in L, sorted from largest to smallest. For example, medians([[3,1,2], [6,1]]) would return [3.5, 2]. If L is not a 2d list or contains non-numbers or empty lists, your function must return None (and not crash).

5. **Bonus/Optional**: **Code Tracing** [2.5 pts]:Indicate what this prints. Circle your answer. Show your work!

```
def bonus1(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[-1]
print(bonus1(10, [0]))
```