Name: ___________________________ Section: _____ Andrew Id : _____________________

15-112 Summer-1 2017 Quiz 3

***Up to 50 minutes. No calculator, no notes, no books, no computers. Show your work!**
No recursion!

1. **Short Answers** [15 pts]

Let a = list(range(N)) and b = list(range(4*N)) The following table gives the runtimes of calling foo(a) and foo(b) for a reasonably large value of N. Fill in the third column, indicating the big-oh function family that most likely describes foo's runtime, where all the answers will be among the major function families we have studied (O(1), O(logN), O(sqrt(N)), O(N), O(NlogN), O(N**2), O(2**N).

Time for foo(a)	Time for foo(b)	Big-Oh function family
2 s	32 s	
4 s	16 s	
4 s	19 s	
3 s	3.5 s	
3 s	6 s	

2. **BigOh** [10 pts]: What is the bigOh of each of the functions below?

```
def bigOh1(L):
    N = len(L)
    a = range(0, 3*N, 3)
    for x in range(1, N, N/100):
        if (x in a):
            print(x)
```

```
def bigOh2(s):
    myS = s * len(s)
    result = ""
    for c in myS:
        if c in result:
            # Hint: think carefully about how this statement affects the BigOh
            break
        else:
            result+=c
    return result
```

3. **Code Tracing** [15 pts]: Indicate what these print. Place your answers (and nothing else) in the boxes below the code.

```
import copy

def ct3(a):
    b, c, d = a, copy.copy(a), copy.deepcopy(a)
    b[0][0] += 1
    c[0][1] += 2
    d[1][0] += 3
    a[1] = [c[0][0]] + [c[0][1]]
    d[0][0] += 5
    a = [[1, 1], [0, 0]]
    print(b)
    print(c)
    print(d)

a = [[1,2],[3,4]]
ct3(a)
print(a)
```

4. **Reasoning over code** [10 pt]: Find an argument for the following function that makes it return True. Place your answers (and nothing else) in the boxes below the code.

```
def g(L):
    (m,n) = (len(L), len(L[0]))
    A = [ ]
    for r in range(m):
        for c in range(n):
            A.append(L[r][c])
    assert(sorted(A) == list(range(m*n)))
    return (L[m-2][n-2] == L[m-1][n-1] + 1)
```

5. **Free Response 1: nQueensChecker(board)** [50 pts]: Background: in Chess, a Queen can move any number of squares in all possible 8 directions (horizontally, vertically, and diagonally)
The "N Queens" problem asks if we can place N queens on an NxN chessboard such that no two queens are attacking each other. For most values of N, there are many ways to solve this problem

Here though, you will write the function nQueensChecker(board) that takes a 2d list of booleans where True indicates a queen is present and False indicates a blank cell, and returns True if this NxN board contains N queens all of which do not attack any others, and False otherwise.

For example, consider the following board:

```
board1 =  [
              [False, False, True , False],
              [True , False, False, False],
              [False, False, False, True],
              [False, True,  False, False]
          ]
```

it's a 4x4 board with 4 queens in it. Notice how none of the queens can attack each other because regardless of the direction any of them moves in (any of the possible 8 directions), it will not collide with any of the other queens.

And now consider the following board:

```
board2 =  [
              [False, False, True , False],
              [False, False, False, True ],
              [True , False, False, False],
              [False, True,  False, False]
          ]
```

it's again a 4x4 with 4 queens in it. Here though, the queen in row 0 col 2, can attach the queen in row 2 col 0 by moving down-left and it can also attack the queen in row 1 col 3 by moving down-right.

So nQueensChecker(board1) would return True but nQueensChecker(board2) would return False.