Name:___________________________________ Section:___ Andrew Id: ____________________

15-112 Fall 2015 Quiz 10x (a)
*** Up to 40 minutes. No calculators, no notes, no books, no computers.**
*** 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(y):
    def f(x,y=3):
        g = lambda x: x*y
        return g(x) if (x%2 == 0) else f(x+1,x%2)
    if (y < 1): return [42]
    else: return ct1(y-1) + [f(y)]
print(ct1(4)) # hint: prints a list with 5 values
```

```
def ct2(L, depth=0):
    print("  "*depth, L)
    if (len(L) < 2):
        result = sum([x**2 for x in L])
    else:
        mid = len(L)//2
        (L1, L2) = (L[:mid], L[mid:])
        # hint: don't miss the *2 in the next line!
        result = ct2(L1, depth+1)*2 + ct2(L2, depth+1)
    print("  "*depth, "-->", result)
    return result
print(ct2([1,3,2])) # prints 11 total lines
```

2. **Reasoning Over Code [10 pts]:** For the following function, find the value of y that will make the function return True. You only need one set of arguments, even if there are multiple correct answers. Show your work. Place your answer (and nothing else) in the box below the code.

```
def rc1(y):
    def f(x, r=0):
        if (x == 0): return y+r
        else: return f(x-1,x+r)
    return (f(5) == 25)
```

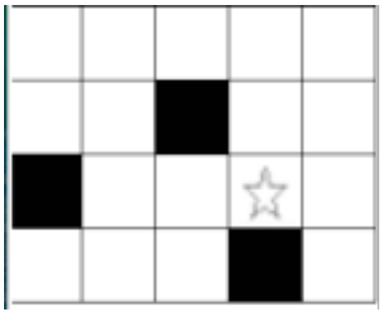
```
y =
```

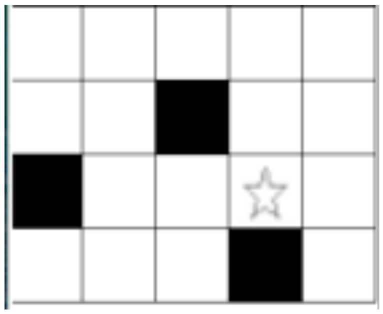
3. **Short Answer [15 pts]:** Answer in as few words as possible. Be very brief.

   a. Noting that it takes 3 lines to draw the "H" in a level-0 H-fractal, how many lines are required to draw a complete level-1 H-fractal?

   b. Noting that a level-0 sierpinsky triangle draws 1 black triangle, how many black triangles are required to draw a level-4 sierpinsky triangle?

   c. Say we start with a 4x5 board and fill the black squares as shown below (in the code from our notes, the black squares would be green, and the white squares would be cyan). Then we right-click where the star is to start a floodFill from there. On the left, write the numeric labels for the <u>depths</u> that result in each cell after the floodFill completes. On the right, write the numeric labels for the <u>ordinals</u>.


depths


ordinals

Hint #1: the numbers on the left are the *depth* at each cell, so some labels may occur more than once.
Hint #2: the numbers on the right are the *ordinals* at each cell, so labels must occur only once each.
Hint #3: the first label is 0, not 1 (so a 0 should be where the star is).
Hint #4: Our floodFill code recursively tries to go up, then down, then left, then right.

4. **Free Response**: **mostPopularSongTitle(path)** [40 pts]

Note: another Free Response follows, on the last page, before the bonus!

Background: you can store a playlist of song titles in a text file, one song title per line. You can then have a folder full of these text files, so the folder is a list of playlists. And naturally, these folders can also contain other such folders recursively. We will assume that each song title occurs at most once in each playlist text file.

With this in mind, write the function mostPopularSongTitle(path) that takes a path to a folder that contains playlist text files or other such folders (recursively), and returns the song title that occurs in the most playlists. If there is a tie, return a set containing each song title that occurs in the most playlists.

For example, if "Hotline Bling - Drake" is in text files playlist1 and playlist2, and only those two text files, then it will have occurred twice overall.

You may assume that the function readFile(textFilePath) exists and returns the string contents of the given textFilePath.

This page left blank for mostPopularSongTitle.

Note: another Free Response follows, on the next page, before the bonus!

5. **Free Response**: **nearestTenth decorator** [15 pts]
Write the decorator @nearestTenth that can be applied to any function with any number of parameters, and if that function returns a float, the decorated function returns that same float rounded to the nearest tenth. If the function returns a non-float, though, the decorated function returns that value unmodified.

For example, here is how the decorator can be used:

```
@nearestTenth
def f(x): return x*5

print(f(1))      # prints 5 (an int)
print(f(1.111))  # prints 5.6 (a float, but not 5.555)
print(f("z"))    # prints zzzzz (a string)
```

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

```
def bonusCT():
    def f(x):
        if (x < 5): print("f%d" % x, end="")
        if (x < 1): return 1
        return x + f(x-2) if (x%2 == 0) else x + f(x+2)
    def g(x):
        print("g%d" % x, end="")
        try: return f(x+1)
        except: return g(x+3)
    print(" -->",g(2))
bonusCT()
```