Name:_____ Section:___ Andrew Id: _____

15-112 Fall 2016 Quiz10a

*** Up to 30 minutes. No calculators, no notes, no books, no computers. * Show your work!**

1. **Short Answer** [20 pts]
   Be very brief.

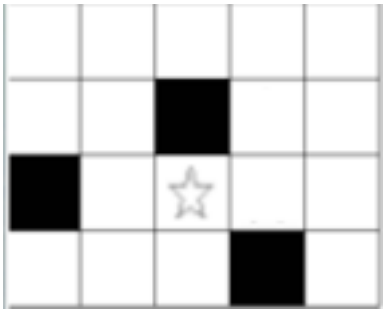
   a. In just a few words, why is it natural to use recursion to solve folder-and-file problems?

   b. Unlike recursive fib(n), recursive fact(n) is fast even without memoization. In just a few words, why is this true?

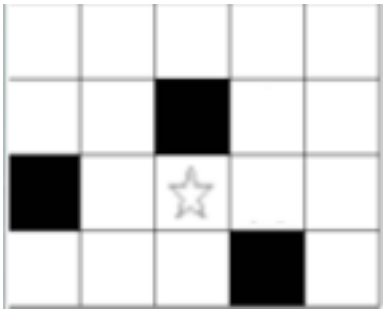
   c. In just a few words, why don't we need callWithLargeStack when running fib(n)?

   d. Draw a level-2 Sierpinsky Triangle if level-0 is a solid black triangle.

   e. 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 depths that result in each cell after the floodFill completes. On the right, write the numeric labels for the ordinals.



depths



ordinals

   Hint #1: the numbers on the left are depths, so some labels may occur more than once.
   Hint #2: the numbers on the right are ordinals, 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.

2. **Reasoning Over Code** [10 pts]:
Find an argument for the following function that makes it return True. Place your answer (and nothing else) in the box below the code:

```
def rc1(L):
    for val in L:
        if (not isinstance(val, int)) or (val < 2):
            return False
    def f(L, n):
        if (len(L) < 2): return True
        else:
            return ((L[0]*n == L[-1]) and
                    f(L[1:-1], n+1))
    return ((len(L) == 6) and
            (len(set(L)) == 6) and
            f(L, 2))
```

L =

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

```
def ct1(x=3, y=5):
    def myDecorator(f, x=1): return lambda *args: f(*args) + x
    def f(x, y=1, z=2): return 100*x + 10*y + z
    @myDecorator
    def g(n): return f(n, z=3)
    def h(*args): return [g(n) for n in args]
    return h(x,y)
print(ct1(2))
```

```
def ct2(n):
    if (n <= 4): # note the unusual condition
        return [ [ n ] ]
    else:
        L = ct2(n//2)
        M = copy.copy(L)
        for a in L: M += [[n] + a]
        return M
print(ct2(11))
```

Name:_____ Section:___ Andrew Id: _____

4. **Free Response: findSmallestDivishNumber(digits)** [50 pts]

Background: We will say that a positive integer N is 'divish' (a coined term) if:

* N contains no 0's, and
* Each sequence of 3 digits in N is a multiple of 7, and
* Each sequence of 3 digits in N is unique

For example, we see that 11266 is divish because 112, 126, and 266 are all multiples of 7. As another example, we see that 112665112 is not divish because it contains the digit sequence 112 twice. Also note that any positive integer N with fewer than 3 digits and no 0's is divish.

With this in mind, write the function findSmallestDivishNumber(digits), that you can abbreviate as f(d), that takes a positive integer number of digits and uses backtracking to return the smallest divish number with that given number of digits, or None if no such number exists. To receive any credit, you must use backtracking here.

Hint: From the definition, we see that if a number N is divish, then all of its prefixes are also divish.
Important Note: unlike most integer problems, here you may use strings without penalty.

The top of this page is blank

5. **Bonus/Optional: Code Tracing** [5 pts; 2.5 pts each] What will these print? Place your answer in the boxes.

```
def bonusCt1(n):
    i = lambda n: -1 + i(n-1) + 2*k(n) if n else 0
    j = lambda n:  3*(i(n) - k(n)) + 1 + j(n-1) if n else 0
    k = lambda n: 1 + k(n-1) if n else 0
    return i(n)+j(n)-k(n)
print(bonusCt1(5))
```

```
def bonusCt2(a):
    return [[]] if (len(a)<1) else [s[:i]+[a[0]]*(i+1)]+s[i:]
                for s in bonusCt2(a[1:]) for i in range(len(s)+1)]
print(bonusCt2(bonusCt2([2,3]))[-1])
```