Name:_____ Section:___ Andrew Id: _____

15-112 Spring 2014 Quiz 8a

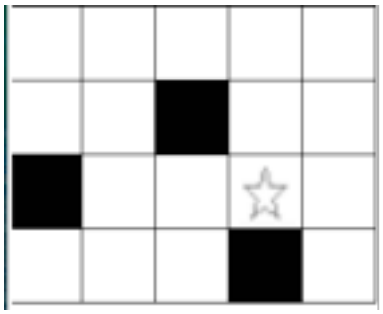
* 35 minutes. No calculators, no notes, no books, no computers.

* Do not discuss this quiz with anyone until after 5pm today. SHOW YOUR WORK, CIRCLE YOUR ANSWERS.

1. **Short Answer** [30 pts; 6 pts each]: Answer in as few words as possible. Be very brief.
   a. In just a few words, why does memoization speed up fib(n) so much more than fact(n)?

   b. The recursive case of listFiles(path) can be written in a single line. Write that one line of code here. Your response must be exactly one Python return statement and no more. Hints: you will want to use a list comprehension, and os.listdir(path), and you are allowed to use the flatten function that you wrote in hw9, which you may assume already exists (don't write it here).

   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. Write the numeric labels that result in each cell after the floodFill completes.



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

   d. If a Level-0 Sierpinski triangle is simple a black triangle, draw a Level-2 Sierpinski triangle, and circle all the Level-1 Sierpinski triangles inside of it.

   e. Here is the maze-solving code from the lecture notes with the last 2 lines removed. Write the missing 2 lines, with proper indenting.

```
def solve(row,col):
      if (row,col) in visited: return False
      visited.add((row,col))
      if (row,col)==(targetRow,targetCol): return True
      for drow,dcol in [NORTH,SOUTH,EAST,WEST]:
              if isValid(row,col,(drow,dcol)):
                    if solve(row+drow,col+dcol): return True
```

2. **Code Tracing** [10 pts]: Indicate what this will print:

```
def ct(x,depth=0):
    print "  "*depth + "ct(" + str(x) + ")"
    if (x < 5): result = x
    else: result = ct(x/4, depth+1) + 2*ct(x-4, depth+1)
    print "  "*depth + "--> " + str(result)
    return result
print ct(10) # this prints 11 lines!
```

3. **Reasoning Over Code** [10 pts]
Find arguments for the following function that make it return True. You only need one set of arguments for the function, even if there are multiple correct answers.

```
def rc(a):
    def f(a, x=32):
        if (a == [ ]): return (x == 0)
        elif (x%2 == 1): return f(a, x-1)
        else: return (a[0] == x) and f(a[1:], x/2 - 1)
    return f(a)
```

4. **Free Response: recursive sumOfOddDigits(n)** [25 pts]
   Without using any iteration, write the function sumOfOddDigits(n) that, given a possibly-negative integer n, returns the sum of all the odd digits in n. For example, sumOfOddDigits(-34188811) returns 6 (3+1+1+1).

Quiz is continued on next page!

5. **Free Response: recursive hanoiCount(n)** [25 pts]
Without using any iteration, write the function hanoiCount(n) that takes a non-negative integer n, and -- without creating or using any lists or tuples – returns the number of moves that our hanoi(n) function would return to solve the Towers of Hanoi puzzle with n disks. For half-credit, you may use lists in your solution. Note that your solution must use recursion fundamentally, and may not just return the closed-form solution (which is 2**n-1, by the way).

6. **Bonus/Optional: Code Tracing** [10 pts; 5 pts each]: Indicate what these will print:

```
def bonus1(n): return "n" if not(n) else str(n)[n%10%len(str(n))] + bonus1(n/10)
print bonus1(4321)
```

```
def bonus2(a, L=[]):
    if (len(L) > 1): L.append(a); return L
    else: L.append(a); return bonus2(L+a)
a = bonus2([1])[-1]; print (len(a), str(a).count("["))
```