Name:_____ Section:___ Andrew Id: _____

15-112 Fall 2016 Quiz9a

*** Up to 35 minutes. No calculators, no notes, no books, no computers. * Show your work!**

1. **Short Answer** [15 pts]
   Be very brief.

   a. Write myMap using a list comprehension.

   b. If myFilter(L) is implemented using recursion by halves, what is the maximum recursion depth for a call to myFilter(L)?

   c. Very briefly, why can't you write myReduce with a list comprehension?

2. **Reasoning Over Code** [5 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:

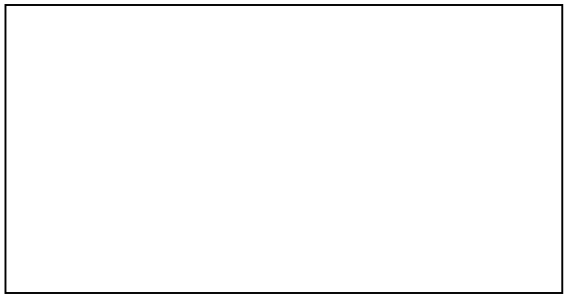
```
def rc(n):
    def f(n, k):
        if (n == 0): return [ ]
        elif (n%2 == k): return f(n//10, k)
        else: return [str(n%10)] + f(n//10, k)
    def g(n, k):
        return int(''.join(reversed(f(n, k))))
    return ((isinstance(n, int)) and
            (2000 < n < 4000) and
            (g(n, 0) == n//100) and
            (g(n, 1) == n%100))
```

n =

3. **Code Tracing** [10 pts]:Indicate what this prints. Place your answer (and nothing else) in the box below the code.

```
def ct1(n, d=0):
    def f(s, n, d): return '%s%d' % (s, 10*n+d)
    if (n <= 1): return [f('A', n, d)]
    else: return ct1(n-1, d+1) + [f('B', n, d)] + ct1(n-2, d+1)
print(ct1(3)) # prints a list of 5 strings
```

```
def ct2():
    f = lambda L: list(filter(lambda v: v%3>0, L))
    g = lambda L: list(map(lambda v: chr(ord('a') + v), L))
    h = lambda L: reduce(lambda a,b: a+str(len(b))+b, L, '')
    print(f(range(2, 6)))       # prints a list of length 3
    print(g([1, 3, 5]))         # prints a list of length 3
    print(h(['a', 'bc', 'd']))  # prints a string of length 7
ct2()
```



Name:_____ Section:___ Andrew Id: _____

4. **Free Response: A class** [35 pts]

Write the class A so that the following test code passes. You may not hardcode any test cases.

```
a1 = A(42)
assert(a1.x == 42)
assert(str(a1) == 'A instance with x == 42')
assert(str(A(99)) == 'A instance with x == 99')
a2 = a1.plusOne()
assert(isinstance(a2, A))
assert(str(a2) == 'A instance with x == 43')
assert(a2 == A(43))
assert(a2 != 43) # don't crash!
s = set()
s.add(A(1))
assert(A(1) in s)
```

5. **Free Response: sumOfSquares(L)** [35 pts]
Without using iteration or map/filter/reduce, write the recursive function sumOfSquares(L), that you can abbreviate as sos(L), that takes a list L of arbitrary values and returns the sum of the squares of the integers in L, ignoring all other values. So:
sumOfSquares([3, set(), 2.4, 5, "wow"]) == 3**2 + 5**2
Also the function returns 0 if there are no integers in the list.

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

```
def bonusCt1(n):
    def f(n): return 0 if (n <= 0) else 2*n-1+f(n-1)
    def g(n): return min(700, f(n)) if (f(n) > 100) else g(f(n))
    return g(n)
print(list(map(bonusCt1,range(2,6))))
```

```
def bonusCt2(L):
    f = lambda L: reduce(lambda x,y: x+[len(y)], L, [ ])
    g = lambda L: reduce(lambda x,y: x+[len(str(y))], L, [ ])
    h = lambda L: list(map(lambda x,y: x**y, g(L), f(L)))
    return h(L)
print(bonusCt2([[42], "two"]))
```