Name:_____ Section:___ Andrew Id: _____

15-112 Spring 2016 Quiz 9xa

*** Up to 45 minutes. No calculators, no notes, no books, no computers.**

*** No iteration (only use recursion)! * To receive credit (in Code Tracing), show your work.**

1. **Code Tracing** [50 pts]:Indicate what this prints. Place your answer (and nothing else) in the box below the code.

```
def ct1(n):
    if (n == 0):
        return 0
    else:
        return (n%2) + ct1(n//10)
print(ct1(4567654))
```


```
def ct2(L):
    if (len(L) < 2):
        return L
    else:
        return [L[1]] + ct2(L[2:])
print(ct2(list(range(1,6))))
```


```
def ct3(s):
    if (s == ""):
        return s
    else:
        mid = len(s)//2
        return s[mid] + ct3(s[:mid])
print(ct3("cdefgh"))
```


```
# ct4() uses class A
class A(object):
    def __init__(self, x):
        self.x = x
        self.y = x+1

def ct4():
    a1 = A(2)
    a2 = A(a1.x + a1.y)
    return (a2.x, a2.y)
print(ct4())
```

```
# ct5() uses class B
class B(object):
    def __init__(self, board, row, col, target):
        self.board = board
        self.row = row
        self.col = col
        self.target = target
    def f(self, other):
        target = other.target
        board = self.board
        (rows, cols) = (len(board), len(board[0]))
        for row in range(self.row, rows):
            for col in range(self.col, cols):
                if (board[row][col] == target):
                    return (row, col)
        return None

def ct5():
    board = [ ([0]*10) for row in range(20) ]
    (rows, cols) = (len(board), len(board[0]))
    for row in range(rows):
        for col in range(cols):
            board[row][col] = row+col
    b1 = B(board, 8, 4, 0)
    b2 = B(None, 13, 14, 15)
    return b1.f(b2)
print(ct5())
```

Name:_____ Section:___ Andrew Id: _____

2. **Reasoning Over Code** [10 pts; 5 pts each]:
Find arguments for the functions rc1 and rc2 that make them return True. You only need one set of arguments for each function, even if there are multiple correct answers.

```
def rc1Helper(n):
    if (n == 1):
        return True
    elif (n%10 > 0):
        return False
    else:
        return rc1Helper(n//10)

def rc1(n):
    m = n
    while (rc1Helper(n) == False):
        n += 1
    return ((n > 123) and (n - m == 50))
```

n =

```
def rc2Helper(L, depth=0):
    if (sum(L) > 500):
        return (L[-1], depth)
    else:
        L2 = [val*10 for val in L]
        return rc2Helper(L2, depth+1)

def rc2(L):
    assert(L == sorted(L))
    return (rc2Helper(L) == (500, 1))
```

L =

3. **Free Response: maxEven** [40 pts]
   Without using iteration, write the recursive function maxEven(L) that returns the largest even integer in the list L (which you may assume contains only integers). If no such number exists, return None.

4. **Bonus/Optional: Code Tracing** [5 pts]
   Indicate what this prints. Place your answer (and nothing else) in the box below the code):

```
def bonusCt1(n):
    def z(n): return (n+2 if (n>10) else sum([z(m) for m in range(10*n, 10*n+2)]))
    def zk(n, k): return n if (k == 0) else zk(z(n),k-1)
    return zk(n, 100)
print(bonusCt1(1))
```