Name:_____ Section:___ Andrew Id: _____

15-112 Spring 2014 Quiz 5

* 20 minutes. No calculators, no notes, no books, no computers. No sets, maps, or recursion!

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

1. **Code Tracing** [40 pts]: Indicate what this will print, filling in the spaces provided on the right:

```
def ct1(a):
    print a[1:7:2]
    x = a.pop(0)
    print x, a.index(x)
ct1(range(4) + range(-2,2))

def ct2(a):
    print [(a[i]*a[i-1]) for i in xrange(1,len(a))]
    b = sorted(a)
    b[0] = 42
    print a[:2], b[:2]
ct2([5,2,4,3])

def ct3(a):
    b = copy.copy(a)
    c = a[0:len(a)]
    d = a
    a[0] += 1
    b[0] += 2
    c[0] = 3
    d.append(4)
    print a, b, c, d
a = [5]
ct3(a)
print a

def ct4(a):
    b = [ ]
    for v in a:
        if (type(v) == tuple):
            b.extend([v[0], len(v)])
    print b
ct4((1, (2), (3,), (4,5),))
```

_____

_____

_____

_____

_____

_____

_____

2. **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 rc1(L):
    assert((type(L) == list) and (None not in L))
    i = 0
    while (L[i] != None):
        j = L[i]
        L[i] = None
        i = j
    a = [None]*2
    return (L == a + [-1] + a)
```

3. **Free Response: checkNearlySorted(L)** [50 pts]
We will say that a list is "nearly-sorted" (a coined term) if it is not sorted but it requires exactly one swap of two of its values to become sorted from least to greatest. For example, a=[5,9,7,8,6] is nearly-sorted, since swapping a[1] and a[4] results in a sorted list. Similarly, a=[1,2,3] is not nearly-sorted, since it is already sorted.

With this in mind, write the function checkNearlySorted(L) that takes a list L of integers, and returns False if the list is not nearly sorted. If the list is nearly sorted, the function does not return True, but rather it returns the tuple (i,j), where i<j and swap(L, i, j) would make the list sorted. So, from the example above, checkNearlySorted([5,9,7,8,6]) should return (1,4). For full credit, your function must be non-destructive. Also, do not worry about how efficient your algorithm is.

4. **Bonus/Optional**: [2.5 pts] Without using the builtins sort() or sorted(), assuming L is a list of integers, write a single expression (not a statement) that evaluates to True if L is fully sorted from greatest to least, and False otherwise.