Name:_________________________________ Section:___ Andrew Id: ____________________

15-112 Spring 2017 Quiz 4a

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

1. **Code Tracing** [20 pts]:Indicate what these print or (for graphics) draw. Place your answers (and nothing else) in the boxes below the code.

```
def ct1(A, B, C, D, E):
    result = [ ]
            #  0     1     2     3     4     5     6     7     8     9
    pairs = [(A,B),(A,C),(A,D),(A,E),(B,C),(B,D),(B,E),(C,D),(C,E),(D,E)]
    for i,pair in enumerate(pairs):
        (L, M) = pair
        if (L is M): result.append(i)
        elif (L == M): result.append(10*i)
    return result
def f(L):
    L[0] += 1
    return L
A = list(range(3))
B = copy.copy(A)
C, D, E = A, B+[ ], f(B)
print(ct1(A, B, C, D, E))
```

```
def drawCt2(canvas, width, height):
    # Draw a picture of what this draws on the screen.
    # Assume the box below is drawn by: canvas.create_rectangle(0, 0, 200, 200)
    for x in range(50, 200, 50):
        if (x < 100):
            canvas.create_line(x, x, x/2, 4*x)
        elif (x > 123):
            canvas.create_oval(x, 0, 200, 100)
        else:
            canvas.create_polygon(x, 100, 2*x, 50, x, 150)
            canvas.create_text(x, 150, anchor=NW, text="CT2!")
```

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

```
def rc1(L):
    if (not isinstance(L, list)): return False
    result = [ ]
    while (L != [ ]):
        result.extend([L.pop(), L.pop(0)])
        L = L[1:-1]
    return (result == list(range(2,6)))
```

```
L =
```

3. **Short Answers** [10 pts]:
Unlike the rest of this quiz, the questions in this section (and just this section) cover check5 (2d Lists). Answer each of the following in **just a few brief words** or a line or two of code, as appropriate.

   a. Given a rectangular 2d list L, write an expression (not a statement) that evaluates to a tuple containing the dimensions of L (rows x cols).

   b. Write one line of Python that assigns a non-rectangular 2d list into the variable L.

   c. Say that a 2d list L is incorrectly allocated as such:
   ```
   (rows, cols) = (2, 2)
   L = [ [0] * cols ] * rows
   ```
   Then, we set L[0][0] to 42. What other value in L will now be 42?

   d. Given a 500x500 list L, write a list comprehension that evaluates to a 1D list of the values in the last (rightmost) column of L. Do not write any statements here, just one list comprehension.

   e. Draw the box-and-arrow diagram of L and M after this code is run:
   ```
   L = [[1],[2]]
   M = [L[0], L[0][0]]
   ```

Name:_________________________________ Section:___ Andrew Id: ___________________

4. Fill in the Blank [10 pts]:

Fill in the 5 blanks with the missing code from the case studies in the notes.

```
def sieve(n):
    isPrime = [ True ] * (n+1) # assume all are prime to start
    isPrime[0] = isPrime[1] = False # except 0 and 1, of course
    primes = [ ]
    for prime in range(n+1):

        if (________________________________________________________________________):

            # we found a prime, so add it to our result
            primes.append(prime)
            # and mark all its multiples as not prime
            for multiple in range(2*prime, n+1, prime):

                ______________________________________________________________________

    return primes




def swap(a, i, j):

    ______________________________________________________________________________________




def selectionSort(a):
    n = len(a)
    for startIndex in range(n):

        minIndex = ___________________________________________________________________

        for i in range(startIndex+1, n):

            if (____________________________________________________________________):

                minIndex = i
        swap(a, startIndex, minIndex)
```

5. **Free Response: nearMedians(L)** [40 pts]
The median of a sorted list L is the middle value (or the average of the two middle values if the length of L is even). Assuming L only contains integers, we will say that a value is "near-median" if it is within 10, inclusive, of the median.

With this in mind, write the function nearMedians(L) that takes an arbitrary Python value, and if it is a (possibly unsorted) non-empty list only containing integers, the function returns a sorted list of the near-median values in L. Otherwise, the function returns None.

Here is a sample test function for you. You may wish to carefully look it over, as it may help you further understand the problem spec:

```
def testNearMedians():
    print('Testing nearMedians()...', end='')
    assert(nearMedians([1, 49, 50, 51, 99]) == [49, 50, 51])
    assert(nearMedians([49, 1, 50, 99, 51]) == [49, 50, 51])
    assert(nearMedians([1, 48, 52, 99]) == [48, 52])
    assert(nearMedians([48, 1, 99, 52]) == [48, 52])
    assert(nearMedians([1, 1, 1, 1, 1]) == [1, 1, 1, 1, 1])
    assert(nearMedians([ ]) == None)
    assert(nearMedians(["ugh"]) == None)
    assert(nearMedians("ugh") == None)
    print('Passed')
```

Name:___________________________________ Section:___ Andrew Id: _____________________

This page is intentionally blank for your nearMedians solution.

6. **Bonus/Optional: Code Tracing** [7.5 pts] Indicate what these print. Place your answers (and nothing else) in the boxes below the code.

```
def bonusCt1(n):
    L = [n//n]*n; L = L*n
    return sum([sum(L)]*len(L))
print(bonusCt1(3))
```

```
def bonusCt2(L):
    def f(g,L):
        r = []
        for v in L: r.extend([v] if g(v) else [])
        return len(r)
    def g(z): return (z%5)*(z%3)
    return f(g,list(L))
print(bonusCt2(range(1500)))
```

```
def bonusCt3(L):
    s = str(list(L))
    while (s.count("'") < 20): s = str(list(s))
    return s.count("'")
print(bonusCt3(range(2)))
```