15-112 Practice Quiz 3

Code Tracing

```
def ct1(L):
    a = L
    b = copy.copy(L)
    c = copy.deepcopy(L)
    a[0] = b[1]
    b[1][1] = c[0]
    c[1].append(b[1])    #a → [[3], [3]] b→ [[1], [3]] c ⇒ [[1], [2, 5, [3]]]
    a[0][0] += (b[1].pop())[0]
    return (a,b,c)
# Be careful to get the brackets
# and commas right!
for val in ct1([[1],[2,5]]):
    print(val) # prints 3 lines
```

Reasoning Over Code

```
def r(n, row):
   a = [ ([0] * n) for r in range(n) ]
   counter = 1 # note: start at  1 not 0!
   for c in range(n):  # note: col first
       for r in range(n):
           a[r][c] = counter
           counter +=  1
   return (a[row] == [3, 7, 11, 15])
```

Big-Oh

```
def f1(n):
    L = [0] * n
    while (n > 0):
        for i in range(len(L)):
            L[i] += i ** 2
        n //= 2
    return L
```

O(________________)

```
def f2(n):
   k=1
   while (k**2 < n):
      k += 1
   return k
```

O(________________)

def f3(n):
   k=1
   while (n > 0):
      (n, k) = (n//4, k+1)
   return k
	O(________________)

def f4(n):
   k=1
   for i in range(n, n**2):
       for j in range(n**3):
           k += 1
   return k
	O(________________)

def f5(n):
   k=1
   for i in range(n, n**2):
      k += 1
   for j in range(n**3):
      k += 1
   return k
	O(________________)

Short Answers:

Give a brief explanation and the bigO of linearSearch, binarySearch, selectionSort, and bubbleSort

Give a brief explanation and the bigO of mergeSort (picture is acceptable)

Fill in the blank

```
def binarySearch(L, target):
    start = 0
    end = len(L) - 1
    while(start <= end):
        middle = _______________
        if(__________________):
            return True
        elif(____________________):
            end = middle-1
        else:
            start = middle+1
    return False

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)
```

Free Response

isFoiled(L)

Write the non-destructive function isFoiled(L) that takes a rectangular 2d list of ints L and returns True if L is foiled (a coined term) and False otherwise, where a list is foiled if every row in L is equal (==) to some column in L, where rows are read left-to-right and columns are read top-to-bottom.

For example, consider this list:

```
            [   [   1,  1,  2   ],
                [   2,  1,  1   ],
                [   1,  2,  1   ]   ]
Row0 is [1,1,2] which   equals  col1.
Row1 is [2,1,1] which   equals  col2.
Row2 is [1,2,1] which   equals  col0.
So this list is foiled.
```

wordSearchWithWrapAround()

Write wordSearchWithWrapAround(), defined as wordSearch, with the sole difference being, for the following board,

```
board = [['z', 'e', 'v'],
        ['t', 'c', 'a'],
        ['w', 'q', 't']]
```

wordSearchWithWrapAround(board, ‘cat’) would be: “cat, (1, 1), right”

So, essentially, it still goes in a particular direction, except it can wrap around the board, so that when going to the right, after the 2nd col (in the example above), it would continue to the 0th col to reach the letter ‘t’ and complete the word.

Note: You only need to redefine a single function from the framework of wordSearch() as defined in the class notes. You got this