15-112 Spring 2023 Lecture 3/4
Quiz 8
27 minutes

Name: _____

Andrew ID: _____@andrew.cmu.edu

Section: _____

- **You may not use any books, notes, or electronic devices during this quiz.**
- **You may not ask questions about the quiz except for language clarifications.**
- **Show your work on the quiz (not scratch paper) to receive credit.**
- **If you use scratch paper, you must submit it with your andrew id on it, and we will ignore it.**
- **All code samples run without crashing unless we state otherwise. Assume any imports are already included as required.**
- **You may use almostEqual() and rounded() without writing them. You must write everything else.**
- **To receive any credit, you must not use 'for' or 'while' in any code you write on this quiz, and you must use recursion properly.**

1. Fill-in-the-blank [10 pts, 2 pts each]

For each of the following examples from the course notes, fill in the blank(s) with the missing code. Assume that any required helper functions are already defined for you.

1.

```
def fibonacci(n):
    if ____________________________________________________________________:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)
```

2.

```
def gcd(x, y):
    # Euclid's (very fast) algorithm:
    if y == 0:
        return x
    else:
        return gcd(_________________________________ , _____________________________________)
```

3.

```
def move(n, source, target, temp):
    if n == 1:
        return [(source, target)]
    else:
        return (move(n-1, source, temp,   target) +
                move(  1, source, target, temp) +

                move(_______________ , _____________ , _____________ , ___________)

def solveTowersOfHanoi(n):
    return move(n, 0, 2, 1)
```

4.

```
def quickSort(L):
    if len(L) < 2:
        return L
    else:
        smaller, pivot, larger = partition(L)

        return _____________________ + ___________________ + _____________________
```

5.

```
def mergeSort(L):
    if len(L) < 2:
        return L
    else:
        i = len(L)//2
        sortedLeftHalf = mergeSort(L[:i])
        sortedRightHalf = mergeSort(L[i:])

        return ____________________________________________________________________
```

2. Code Tracing [15 pts, 5 pts each]
Indicate what these print. Place your answers (and nothing else) in the box next to each block of code.

```
def ct1(n):
    if n == 0:
        return 0
    else:
        return n**2 + ct1(n//2)
print(ct1(5))
```

```
def ct2(s):
    if len(s) < 2:
        return s
    else:
        c = s[0]
        d = s[-1]
        t = s[1:-1]
        return ct2(t) + c + d
print(ct2('abcde'))
```

```
def ct3(L):
    if len(L) < 2:
        return L
    else:
        i = len(L)//2
        M = L[:i]
        N = L[i:]
        return ct3(M) + [sum(L)] + ct3(N)
print(ct3([1,2,3,4]))
```

3. FR: squarishCount(L) [35 pts]

Background: given a list L of integers, we will say that a value L[i] is "squarish" (a coined term) if L[i] == i^2. For example, if L = [0, 3, 2, 9, 5, 4], the squarish values in L are L[0] and L[3], because L[0] == 0^2 and L[3] == 3^2.

With this in mind, and without using 'for' or 'while' loops, and using recursion properly, write the function squarishCount(L), that takes a list L of integers and returns the number of squarish values in L.

Here is some test cases for you:

```
    assert(squarishCount([ 0, 3, 2, 9, 5, 4]) == 2)
    assert(squarishCount([ 0, 1, 4, 9, 16]) == 5)
    assert(squarishCount([ 1, 4, 9, 25]) == 0)
```

Note: you may want to use a wrapper function or something similar here.

4. FR: letterCounts[s] [40 pts]
Without using 'for' or 'while' loops, and without using the 'count' method, write the recursive function letterCounts(s) that takes a string s and returns a dictionary d mapping each letter in s to the count of the number of times it occurs in s. Note that d should only contain uppercase keys, but the counts should include both uppercase and lowercase letters. Non-letters should be ignored.

Here are some test cases for you:

```
assert(letterCounts('This is a test!!!') ==
                    {'T': 3, 'S': 3, 'E': 1, 'A': 1, 'I': 2, 'H': 1})
assert(letterCounts('eE789') == { 'E': 2 })
assert(letterCounts('?!?@?#?@') == dict())
```

5. Bonus CT [5 pts, 2.5 pts each]
Indicate what these print. Place your answers (and nothing else) in the box next to each block of code.

```
def bonusCt1(x):
    def f(x, y=None):
        if y == None: y = x
        return x + f(x, y-1) if y else 0
    return f(x) + bonusCt1(x//2) if x else 0
print(bonusCt1(7))
```

```
def bonusCt2(L):
    count = 0
    def f(L, i=0):
        nonlocal count
        if i == 0: L = L[:]
        if i == len(L)-1: return L, False
        else:
            b = L[i] < L[i+1]
            if b:
                L[i], L[i+1] = L[i+1], L[i]
                count += 1
            L, bb = f(L, i+1)
            return L, b or bb
    def g(L):
        L, b = f(L)
        return g(L) if b else L
    L = g(L)
    return (L, count)
print(bonusCt2([2, 4, 1, 3]))
```