# Quiz 3 Solution

Kevin Lin

September 22, 2016

1. **Environment Diagrams**

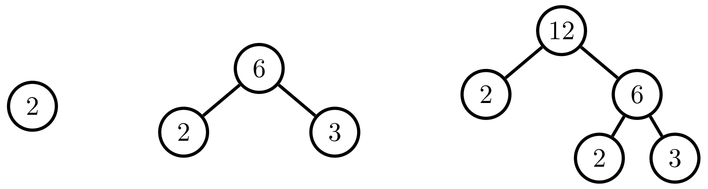
```
def reverse(lst):
    if len(lst) <= 1:
        return lst
    return reverse(lst[1:]) + [lst[0]]

lst = [1, [2, 3], 4]
rev = reverse(lst)
```

https://goo.gl/gnwMQ0

2. **What Would Python Display?** Draw box-and-pointer diagrams!

```
>>> L = [1, 2, 3]
>>> B = L
>>> B
[1, 2, 3]
>>> A = L[1:3]
>>> L[0] = A
>>> L = L + A
>>> B
[[2, 3], 2, 3]
>>> B[0] = A[:]
>>> L[0][0] = A
>>> L[0][0][0][0][0][0][1]
3
>>> B
[[2, 3], 2, 3]
```

3. **Trees** We can represent the factorization of a number with a *full binary tree*: a tree that has either two subtrees or none at all. Define `factor_tree` which takes an integer n greater than one and returns a factor tree for n.



Recall that a factor tree contains only the **prime factors of n** with the exception of the root, n, itself. The `tree` abstraction appears below for your reference.

```
def tree(root, branches=[]):
    return [root] + branches

def root(tree):
    return tree[0]

def branches(tree):
    return tree[1:]

def factor_tree(n):
    for i in range(2, n):
        if n % i == 0:
            return tree(n, [factor_tree(i),
                            factor_tree(n // i)])
    return tree(n)
```

Now, write a procedure, `count`, which counts the number of instances that a prime factor, p, appears in the factor tree t.

```
def count(t, p):
    if p != root(t):
        return sum(count(b, p) for b in branches(t))
    else:
        return 1 + sum(count(b, p) for b in branches(t))
```