1)

a) f = Θ(g) c*f(n) can be >=/<= c*g(n)
b) f = O(g) n^2/3 will always be larger eventually, no matter what C*f(n) equals
c) f = Θ(g)n >log(n)^x, which results in just c*n
d) f = Θ(g)just constants result in c*n
e) f = Θ(g) just constants result in c*log(n), for both f(n) and g(n)
f) f = Θ (g) log(n^2) = 2log(n), which results in both being c*log(n), and thus being theta.

2)

a) if c < 1, then lim of g(n) = 1, resulting in g(n) = Θ(1). Any constant applied can result in an upper or lower bound.
b) if c = 1, then lim of g(n) = ∞, and lim of (n) = ∞. This results in any constant can be applied to (n) to upper or lower bound g(n)
c) if c > 1, then the lim of (g)n = c^n, which is the same as c^n. Any constant can be applied to uppoer or lower bound g(n) in that case.

3.a)

```
def fibExp(n):
    if n == 0:
        return 1
    if n == 1:
        return 1
    if n == 2:
        return 1
    return (fibExp(n-1) + fibExp(n-2) * fibExp(n-3))

print (fibExp(10))
```

f^n due to the function being ran to the power of n. The more N exists, the more it recurses down, exponentially.

b)

```
def fabLinear(n):

    if n == 0:
        return 1
    if n == 1:
        return 1
    if n == 2:
        return 1
    array = [1,1,1]
    for x in range(3, n + 1):
        array.append(x)
    for x in range(3,n + 1):
```

```
        array[x] = array[x-1] + array[x-2] * array[x-3]
    add = (len(array) - 3) *2
    mult = (len(array) - 3)
    return array[n],add, mult
```
add and mult returns the number of additions and multiplications, respectively