Part 2

a)

```
tetraMultiple(n)
        if (n == 0 or n == 1 or n == 2)
                return 0
        if (n == 3)
                return 1
        return (tetraMultiple(n-1) + tetraMultiple(n-2) + tetraMultiple(n-3) + tetraMultiple(n-4))

tetraRecursive(n, a, b, c, d)
        if (n == 0)
                Return a
        If (n == 1)
                return b
        if (n == 2)
                Return c
        If (n== 3)
                return a + b+ c + d
        return tetraRecursive(n-1,a+b+c+d, a, b, c)

tetraLinear(n)
        return tetraRecursive(n,0,0,0,1)
```

c)

Linear:

No branching. Parameters a,b,c,d are incremented and returned upon reaching base case. Since parameters are incremented with each call, repeated calculations are not a problem. Algorithm essentially performs n - 2 recursive calls, so algorithm is O(n)

Multiple:

Branching due to 4 recursive calls each time (if not at base case). Tree-like structure emerges from drawing execution diagram, and it can be seen that 4^n function calls occur at most. So this algorithm is $O(4^n)$. Issue with this algorithm is the repetition of subproblems. Eg. T(n) will call T(n-1), T(n-2), T(n-3) and T(n-4), and these functions will call the other functions again (eg T(n-1) will call T(n-2) again and so on).