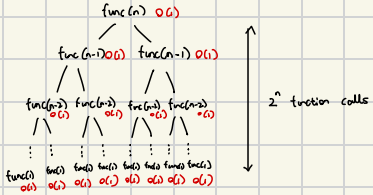
Homework Question 3:

a)

```
def fun1(n)
    if n==0
        return 1
    else
        part 1 = fun1(n-1)
        part 2 = fun(n-1)
        res = part 1 + part 2
        return res
```

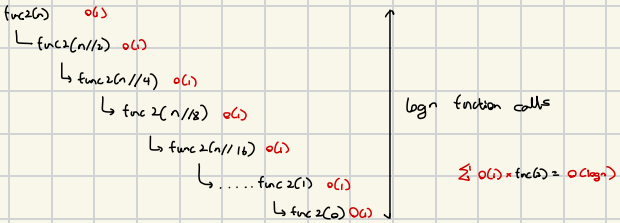


$\sum_{i} O(1) * func(i) = O(2^n)$

each function call:
Time complexity: O(1)
recursive function calls:
Time complexity: $O(2^n)$
overall: $O(2^n)$

b)

```
def func2(n):
    if (n==0):
        return 1
    else:
        res= fun2(n/2)
        res += n
        return res
```



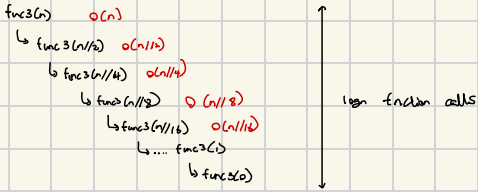
$\sum_{i} O(1) * func(i) = O(logn)$

each function call:
Time complexity: O(1)
recursive function calls:
Time complexity: O(logn)
overall: O(logn)

c)

```
def fun3(n):
    if (n==0):
        return 1
    else:
        res = fun3 (n//2)
        for i in range(1,n+1):
            res+=i
        return res
```

↳ directly affected by the input size reduced on recursion



each function call:
Time complexity: $O(\frac{n}{2^x})$ → at each level the amount of work halves from previous
recursive function call:
Time complexity: O(logn) → number of layers is equal logn

Overall: O(n) ← Total sums to O(n)