TASK 2a :

Implementation 1 : Recursive.

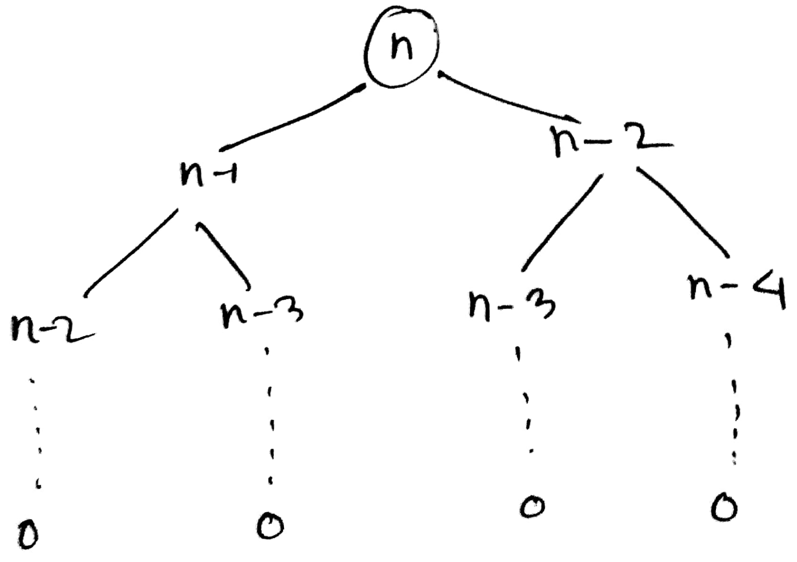
From the code given,

$$T(0) = 0$$
$$T(n) = T(n-1) + T(n-2)$$

By expanding,

$$T(n) = T(n-1) + T(n-2)$$

Recursion Tree :



$$T(n) = \theta(2^n)$$

So, the time complexity is 2^n

Implementation - 02

```
                                                  Times
def fibonacci_2 (n):
    if  n<0:                                        1
        return "Invalid inpnt"                      1
    if  n<=0:                                       1
        return n                                    1
    fib = [0] * (n+1)                               1
    fib[0] = 0                                      1
    fib[1] = 1                                      1
    for i in range (2, n+1):                        n
        fib[i] = fib[i-1] + fib[i-2]          [redacted](n-1)
    return fib[n]                                   1
```

T(n) = [redacted] 2n + 7

$\therefore$ T(n) = [redacted] θ(n)

Time complexity of Implementation 2 is θ(n[redacted])

Here, By comparing two complexity, we get that $\theta(n)$ is better ~~than~~ or faster than $\theta(2^n)$.

To check ~~che~~ this, we plug,

$n = 15$

① $T(15) = (2^{15}) = 32768$ unit of time

② $T(15) = (15) = 15$ unit of time

So, implementation 2 is faster.