Task 2:

a)

Implementation 1:

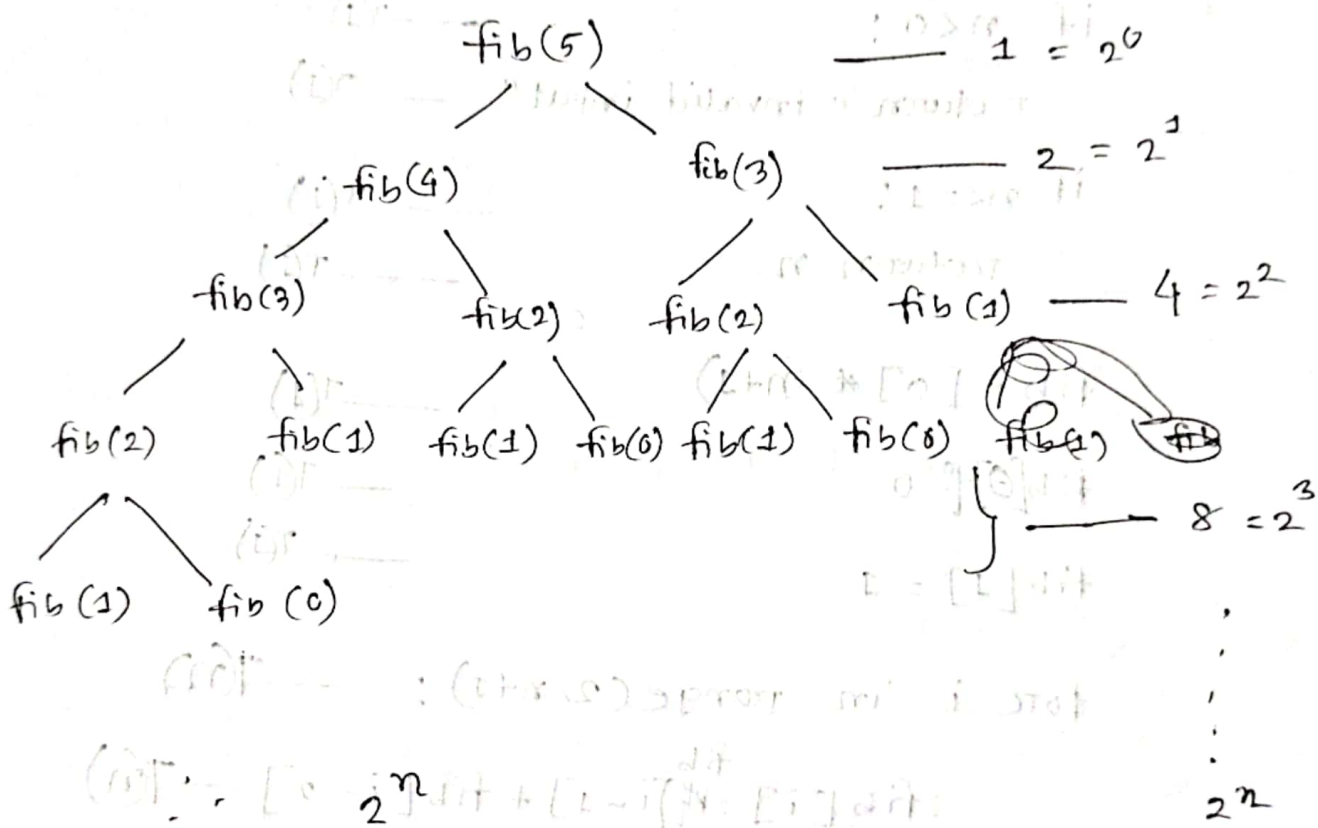
```
def fibonacci_1(n):                          ——T(1)
    if n<0:                                  ——T(1)
        print("invalid input")               ——T(1)
    elif n<=1:                               ——T(1)
        return n                             ——T(1)
    else:
        return fibonacci_1(n-1) + fibonacci_1(n-2)
                                                 └——T(2^n)
n = int(input("Enter number"))               ——T(1)
nth_fib = fibonacci_1(n)                     ——T(1)
print("The {}th fibonacci number is {}.".format
      (n, nth_fib))                          —— T(1)
```

$\therefore f(n) = T(2^n) + T(1) = O(2^n)$.

$O(2^n)$

let $n = 5$

explanation



$\therefore 2^n$

$\therefore O(2^n)$

(Answer)

Implementation 2:

```
def fibonacci_2(n):                                  ---- T(1)
    if n<0:                                          ---- T(1)
        return "invalid input"                       ---- T(1)
    if n<=1:                                         ---- T(1)
        return n                                     ---- T(1)
    fib = [0] * (n+1)                                ---- T(1)
    fib[0] = 0                                       ---- T(1)
    fib[1] = 1                                       ---- T(1)
    for i in range(2, n+1):                          ---- T(n)
        fib[i] = fib[i-1] + fib[i-2]                 ---- T(n)
    return fib[n]                                    ---- T(1)
n = int(input("Enter a number: "))                   ---- T(1)
nth_fib = fibonacci_2(n)                             ---- T(1)
print("The {}th fibonacci number is {}.".format(
    n, nth_fib))                                     ---- T(1)
```

$f(n) = T(1) + T(n) + T(n)$

$\therefore O(n)$

Implementation 1 is exponential time complexity and Implementation 2 is linear time complexity. Apart from faster. Implementation 2, linear is more faster than exponential.

Answer.