Time complexity of Question No. 2 and 4.

2. No. Q. Ans:

implementation-1

```
def fibonacci_1(n):
    if n<=0:                     ---- O(1)
        Print("---")             ---- O(1)
    elif n<=2:                   ---- O(1)
        return n-1               ---- O(1)
    else:
        return fibonacci_n(n-1)+fibonacci_1(n-2)
```

$4 \times O(1) = O(1)$

$\rightarrow O(T(n-1)+T(n-2))$

∴ Time complexity,

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

Base case, $T(2) = 1$

Assume the reccurance is,

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

P.T.O

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

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

$$= 2^3 T(n-3) + 2^2 + 2 + 1$$

$$\vdots$$

$$= 2^{n+2}\, T(n-n+2) + 2^{n+1} + 2^n + \cdots$$

$$= 2^{n+2} \cdot 1 + 2^{n+1} + 2^n + \cdots$$ [using base case]

$$= 2^{n+2} + 2^{n+1} + 2^n + 2^{n-1} + \cdots$$

$$= 2^{(n+2)+1} - 1$$

$$= 2^{n+3} - 1$$

$$= 2^n \cdot 2^3 - 1$$

$$= 2^n$$

$\therefore$ Time complexity $= 2^n$

implementation-2

```
def fibonacci_2(n):
    fibonacci_array = [0,1]          ——> O(1)
    if n<0:                          ——> O(1)
        print(- - - -)
    elif n<=2:                       ——> O(1)
        return fibonacci [n-1]
    else:
        for i in range (2,n):
            fibonacci_array.append(fibonacci_array[i-2]
                                   + fibonacci_array [i-2])
                                     ——> O(n)
```

$\therefore$ Time complexity Becomes:- ~~O(1)+O(1)+O(1)~~

$= O(1) + O(1) + O(1) + O(n)$

$= O(n)$ Ans

4. No. Q. Ans:

Part B : Time complexity:

```
def Multiply_matrix(A,B):
    n = len(A)  ———— O(1)
    c = []      ———— O(1)
    for m in range(n):   ———— O(n)
        s = [0]*3
        c.append(s)      O(2)
    for i in range(n):   ———— O(n)
        for j in range(n):   ———— O(n)
            for k in range(n):   ———— O(n)
                c[i][j] = int(A[i][k])*int(B[k][j])
                          └ O(1)
    return c
```

$O(n \times n \times n) = O(n^3)$

∴ Time complexity:

$$O(1) + O(1) + O(n) + O(n^3)$$

$$= O(n^3) \text{ (ans)}$$