2. Implementation – 1

```
def fibonacci_1(n):
    if n<=0:
        print('Invalid input')
    elif n<=2:
        return n-1
    else:
        return fibonacci_1(n-1)+fibonacci_1(n-2)
```

$$T(n) = T(n-1)+T(n-2)+1$$
$$= 2T(n-1)+1 \quad [T(n-1) \approx T(n-2)]$$
$$= 2\{2T(n-2)+1\}+1$$
$$= 2^2 T(n-2)+2^1+2^0$$
$$= 2^2\{2T(n-3)+1\}+2^1+2^0$$
$$= 2^3 T(n-3)+2^2+2^1+2^0$$

$$T(n+2) = 2^{n+2} T(n-n+2)+2^{n+1}+\ldots\ldots 2^0$$
$$= 2^{n+2}.1+2^{n+1}+\ldots\ldots$$
$$= 2^{n+2}-1$$
$$= 2^n.2-1$$
$$= O(2^n)$$

Implementation – 2

```
def fibonacci_2(n):
    fibonacci_array = [0,1]                    } O(1)
    if n<0:
        print('Invalid input !')               } O(1)
    elif n<=2:
        return fibonacci_array[n-1]            } O(1)
    else:
        for i in range(2,n):
            fibonacci_array.append(fibonacci_array
                                   [i-1]+fibonacci_array[i-2])
                                               } O(1)*n
                                               = O(n)
        return fibonacci_array[-1]
```

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

= $O(n)$

4.

```
for i=0 to n-1                                   O(n)
    for j=0 to n-1                               O(n)
        for k=0 to n-1                           O(n)
            C[i,j] += A[i,k] * B[k,j]
        end for
    end for
end for
```

Time complexity = $O(n(n(n)))$

$= O(n^3)$