Answer to the question no 2

For 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_1(n-1)+fibonacci_1(n-2)   <-- O(T(n-1)+T(n-2))
```

So,

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

$\Rightarrow T(n) = T(n-1) + T(n-1) + c$

$\Rightarrow T(n) = 2T(n-1) + c \quad \text{---------- (1)}$

$= 2\{2T(n-2)+c\} + c \quad [\because \text{using (1)}]$

$= 4T(n-2) + 2c + c$

$= 4\{2T(n-3)+c\} + 2c + c \quad [\text{using (1)}]$

$= 8T(n-3) + 4c + 2c + c$

$= 2^3 T(n-3) + 2^2 c + 2^1 c + 2^0 c$

$\vdots$

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

$= 2^{n+2} \cdot 1 + 2^{n+1} c + 2^n c + \ldots + 2^0 c$

$[T(n-n+2) = T(2) = O(1)]$

$= 2^0 + 2^1 + 2^2 \ldots 2^n + 2^{n+1} + 2^{n+2}$

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

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

$= 2^{n+3}$

$= 2^n$

∴ Time complexity of this program $= 2^n$

Answer to the question no 2

Implementation 2

```
def fibonacci_2(n):
    fibonacci_array = [0,1]                    → O(1)
    if n<0:                                    → O(1)
        print("invalid input")                 → O(1)
    elif n<=2:                                 → O(1)
        return fibonacci_array[n-1]            → O(1)
    else:
        for i in range(2,n):                   → O(n-2) ⇒ O(n)
            fibonacci_array.append(.....)      → O(1)
        return fibonacci_array[-1]             → O(1)
```

So,

Time complexity = $O(1) + O(1) + O(1) + \{O(1) + O(1) + \{O(n) * O(1)\} + O(1)$

= $O(n)$

Answer to the question 4

```
def Multiply_matrix(A,B):
    n = len(A)                          → O(1)
    C = []                              → O(1)
    for i in range(n):                  → O(n) * O(3) = O(n)
        t = [0] * n
        C.append(t)                     → O(3)
        del t
    for i in range(n):                  → O(n) * O(n²) = O(n³)
        for j in range(n):              → O(n) * O(n) = O(n²)
            for k in range(n):          → O(n) * 1 = O(n)
                c[i][j] += A[i][k] * B[k][j]   → O(1)
    return C                            → O(1)
```

So,

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

$$= O(n^3)$$

∴ Time complexity of this program is $O(n^3)$