Name: Md. Sifat Kamal

ID: 20101231

Section: 04

Ans. to the question no: 2

Implementation - 1

```
def fibonacci_1(n):
    if n<=0:
        print("Invalid input!")          } O(1)

    elif n<=2:
        return n-1                       } O(1)

    else:
        return fibonacci_1(n-1) + fibonacci_1(n-2)
n = int(input("Enter the number:"))
nth_fib = fibonacci_1(n)
print("The %d-th fibonacci number is %d" % (n, nth_fib))
```

Here in this code the time complexity of if and elif statement will O(1) or Big of 1. Here ~~we an~~ ~~Besides~~ another statement is else statement. In the else statement there is a recursion that means the function will call several times by himself in the else statement. Here for the time complexity we have to consider the worst case in the worst situation.

Lets consider the time complexity of the whole code will be $T(n)$.

So the equation will be :

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

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

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

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

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

$$\cancel{= 2 + 1 + T(n-1) + T(n-1)}$$

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

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

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

So, the equation becomes.

$$T(n) = 1 + 2 + 2^2 + 3^2 + \ldots + 2^{n-2} + 2^{n-1}$$

$$= \frac{2^0 (2^{n+1} - 1)}{2-1}$$

$$\cancel{= \frac{2^0 (2^{n+1} - 1)}{2 - 1}}$$

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

$= 2^n \cdot 2 - 1$

$= 2^n$

∴ The time complexity of the code will be 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:
```

O(n)

```
        for i in range(2,n):
            fibonacci_array.append(fibonacci_array[i
                + fibonacci_array[i-2]
        return fibonacci_array[-1]
n=int(input("Enter a number"))
nth_fib = fibonacci_2(n)
print("The %d-th fibonacci number is %d" % (n,nth_fib
```

In this code there are three statements. ~~Among them~~ for the code we have to consider worst case in worst case occur. So for this if condition will won't be true. So for this the time complexity will be O(1) or Big O of 1. Similliarly it will be also applicable for the elif statement. So the time complexity will be O(1). Now, since both became false so the code will enter to the else statement. Here in the else statement there is a for loop as well. and the ~~if~~ loop will be run from 2 to (n-1) times. So, the time complexity will be O(n-1) times. Total ~~comp~~ time complexity for the ~~loop~~ statement will be

$O(n-1) * O(1)$.

Now, The total equation will be:

$O(1) + O(1) + O(n) * O(1)$

~~= O(1) + O(n~~ Here we won't count the O(1) since $O(1) < O(n)$

$\therefore O(n)$

Ans: to the question no: 4

```
Procedure Multiply-matrix (A, B)
    Input A, B n×n matrix
    Output C n×n matrix
begin
    Initialize C as a n×n zero matrix
    for i = 0 to n-1
        for j = 0 to n-1
            for = 0 to n-1 :
                c[i, j] += A[i, k] * B[k, j]
            end for
        end for
    end for
end Multiply-matrix
```

In this code, there are three loops which are in nested loop. ~~For the every for loop.~~ Here for the every "for loop" it will ~~is~~ start from 0 ~~and end to~~ to (n-1) times. For the three for ~~the~~ loop the time complexity will be :

$$O(n-1) + O(n-1) * O(n-1) + O(n-1) * O(n-1) *$$

$$O(n-1)$$

~~$O(n-1)$ $O(n) + O$~~

$= O(n) + O(n) * O(n) + O(n) * O(n)$

$* O(n)$

$= O(n) + O(n^2) + O(n^3)$

=

Since $O(n^3) > O(n) > O(n^2)$ so;

$= O(n^3)$

Here I've considered $O(n^3)$ since

we have to take worst case

in worst case scene.