Problem 2

For problem 2.

Implementation 1,

here first the function will call and pass variable 'n' then it will check first condition,

$n <= 0$ and ~~then~~ it will print 'Invalid input'

again it will check elif, $n <= 2$ if true it will return $n-1$.

else function will not run,

$f-1(n-1) + f-1(n-2)$

∴ else loop will work

$5+1=4$

$5-2=3$

$\therefore 4+3=7$

then it will print the input.

here we are having a glimpse of recursive function that's why it will not take longer time to run.

It have time complexity of big O notation. $[O(n)]$

But this recursive approach seems to be more simpler and smaller.

Implementation 2,

here is a input 'n' and contains an array [0,1] 2 index
first it well check n<0 than well print 'invalid'
than it well check n<=2 or not and return of an array of [n-1]
finally it well check the range & it
well push value and well return array of [n-1]

we can see there, used for loop.
so it well run fast slightly fastes than previous method.
Iterative method.
But the time complexity will be same for both
of the implementation method.
It will be big O notation

we all know Big O notation expresses the run time in
terms of how quickly it grows relative to the input.

So this two method are more or all same.
But this iterative method is linear, as the loops runs
i.e.
from 2 to n it runs in O(n) time

Problem 3

Here, we have two implementation method of fibonacci series.

One is recursive method and the one is iterative method.

here in the image, we can see a graph containing 2 different line.

Red one is recursive method graph
and Blue one is iterative process graph

Recursive = O(2^n) (2 to the power n) (Exponential growth rate)

Iterative = O(n) linear growth

here graph shows total n = 30.
two algorithms are compared. n increases after passing 19 and the exponential growth rate of recursive one is clearly evident.

Problem 4

Here given pseudocode is Matrix Multiplication.

here matrix size is $n \times n$

here A & B is Matrix

~~Multi~~ Matrix multiplication, $C = A \cdot B$

$\therefore$ time complexity $= O(n^3)$ [big O n qubic]