3. Recursive Factorial

Algorithm recursiveFactorial (n):

Input: A non-negative integer n

Output : n!

if (n = 0 || n = 1) then 3

return 1 1

return n * recursiveFactorial (n-1) 2 + T(n-1)

By Guessing Method

$$T(n) = \begin{cases} 4 & , n \leq 1 \\ T(n-1) + 2 & , n > 1 \end{cases}$$

$T(1) = 4$

$T(2) = T(1) + 2 = 4 + 2$

$T(3) = T(2) + 2 = 4 + 2 + 2$

$T(4) = T(3) + 2 = 4 + 2 + 2 + 2$

A pattern is emerging !

$T(n) = 4 + (n-1) * 2 = 2(n-1) + 4$

$= 2n + 2$

$= 2(n+1)$

So, $T(n)$ is $O(n)$, i.e., the recursive factorial has linear complexity.

To prove:

By Mathematical Induction, basis step, $n = 1$, $T(1) = 2(1+1) = 4$

Induction step $n = k$, $T(k) = 2(k+1)$

Now, $n = k+1$, $T(k+1) = T(k) + 2 = 2(k+1) + 2$

$= 2[(k+1) + 1]$

$\therefore T(n) = 2(n+1)$

proved.

3. B. Proof of Correctness of Algorithm

By induction,

Base : The values 0! and 1! are correctly computed as 1 by the base case of recursion.

Recursion: Assuming recursiveFactorial (n−1) correctly computes (n−1)!, we have to show that the output of recursiveFactorial (n) is correct. But, the output of recursiveFactorial (n) is

n* recursiveFactorial (n−1)

= n* (n−1)! = n!, as required

∴ The algorithm correctly computes n! for every n.

Proved