Programming question

**Algorithm binaryOdd(k):**

**Input: Nonnegative integer k**

**Output: The kth Oddnacci number Fk**

**if k≤3 then**

**return 1**

**else**

**return binaryOdd(k - 1) + binaryOdd(k - 2)+binaryOdd(k-3)**

**Algorithm linearOddnacci(k):**

**Input: A nonnegative integer k**

**Output: Pair of Oddnacci numbers (Fk , Fk-1, FK-2)**

**if k = 3 then**

**return (1, 1, 1)**

**else[**

**(i, j, m) = linearOddnacci( k – 1 )**

**return (i +j+m, i ,j)**

**]**

b/

binaryOdd()：

to get binaryOdd(k),you need binaryOdd(k-1),k-2,k-3.

And to get k-1 ,you need k-2 k-3 k-4

To get k-2 you need k-3,k-4,k-5

So you need to get k-3 for 3 times.

Take k-3 as j

And to get j( k-3 )for 3times, you need to get j-3(k-6) 9 times.

Take j-3 as m

And to get m(j-3) for 9 times, you need to get m-3(j-6)for 27 times.

………………………………………………………………

So it is exponential complexity

================================================================

LinearOdd()

You just call LinearOdd Once in each recursion, so it is linear.

C/

No

No

For binaryodd() ,it is not linear . Tail recursion occurs when a linearly recursive method makes its recursive call as its last step; as in the array reversal method.

For Linearodd(), it has to call itself first to get the array [I,j,m]. Then it can return [i+j+m, i, j]. So the recursion cannot be last step.