# COMP 352 ASSIGNMENT 1

# PART A

## MultipleRecursion for Tetranacci

1-This java function  is multiple recursive function.
2-It calculate the nth tetranacci but the complexity of this algorithm is worst than the tail recursion.
3- The reason its complexity is exponential because each time the function is called it calls itself again four times which leads to exponential time complexity and also linear growth of stack.

## tailRecursiveTetranacci for Tetranacci

1-This java function use linear recursive function.
It computes the nth Tetranacci by updating the last four numbers by recursive call.
3-The time complexity of this algorithm is linear because each the function calls itself once tail recusively.It has space complexity **O(1)** because its tail recursion and in  tail recursion the function do its thing before even calling itself.

# PART B

## tailRecursiveTetranacci(Linear Tetranacci)

**Time Complexity:** The time complexity of this function is **O(n)** as the function call it self one time till the n decrements and reach to the base case.

**Space Complexity** : The space complexity is **O(1)** because the stake doesnt grow linearly.

**PSEUDOCODE**

Function **tailRecursiveTetranacci**(k,i,j,m,d)
If (k is less than or equal to 0) return i;
BaseCase
If (k is less than or equal to 1) return j;
BaseCase
If (k is less than or equal to 2) return m;
BaseCase
If (k is less than or equal to 3) return d;
BaseCase
else
return tailRecursivetetetetranacci(k-1,j,m,d,i+j+m,d)
END

# MultipleRecursion

**Time Complexity** : The time complexity of multiple recursion is **O(4^n)** it is because each time u call function it call itself three times again(k-1)(k-2)(k-3)(k-4) due to which it becomes exponential in its time complexity.

**Space Complexity** :
Space complexity is **O(n)** because the stack grows linearly.

**PSEUDOCODE**

multiplerecursion(k)
Declaring tetranacci variable
//BASE CASE 1
If (k is 0 or 1 or 2)
Return 0;

//BASE CASE 2
Else if
If (k is 3 or 4)
Return 1;

//recursive call for each function
Else
Return multiplerecursion(k-1)
+multiplerecursion(k-2)
+multiplerecursion(k-3)
+multiplerecursion(k-4)
end