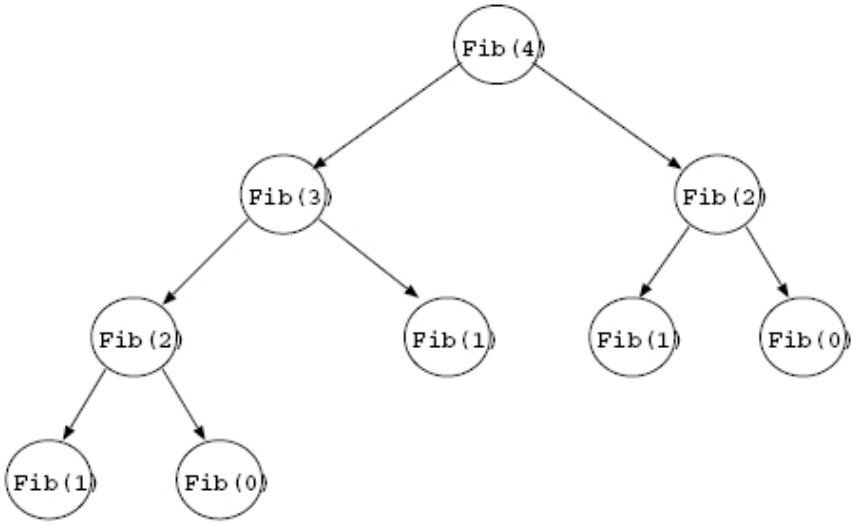
1. Draw a binary tree to illustrate the recursion calls involved in the execution of the routine `Fibonacci(4)`. Every node in this binary tree should be labeled `Fibonacci(i)` to represent a recursion call with parameter *i*. Hence, the root node of this binary tree should be labeled `Fibonacci(4)`.

```
Fibonacci(n)
 if (n == 0) or (n == 1) {
     return n
 }
 else {
     return Fibonacci(n-1) + Fibonacci(n-2)
 }
```

Solution:



2. A binary tree is a full (strict) binary tree if *every* node in the tree has either zero children or two children (*i*.e., no node can have only one child). Write the pseudo-code for a recursive function `is_strict_binary()` that can be used to determine if a given binary tree is a strict binary tree. The function accepts one parameter, which is a pointer to the node currently under consideration as rooting a sub-tree. The function returns `TRUE` if the sub-tree rooted at this node is a strict binary tree, or `FALSE` otherwise. The function will be called by the user by using a pointer to the root node of the tree as a parameter.

Solution:

```
is_strict_binary (struct Node *BT_Ptr) {
  if ( (BT_Ptr->Left_Ptr == NULL) && (BT_Ptr->Right_Ptr == NULL) ) {
    /* This node has zero children */
    return TRUE;
  } else {
    if ( (BT_Ptr->Left_Ptr != NULL) && (BT_Ptr->Right_Ptr != NULL) ) {
      /* This node has two children. Need to check both of them */
      return (is_strict_binary(BT_Ptr->Left_Ptr) & is_strict_binary(BT_Ptr->Right_Ptr));
    } else {
      /* This node has exactly one child */
      return FALSE;
    }
  }
```

3. Prove that for a strictly binary tree with N leaf nodes, there are altogether 2N-1 nodes in the tree. (Hint: Can you do this using mathematical induction?)

Solution: When there is only 1 leaf node in the tree, the statement is true. Assume that the statement is true when there are no greater than k leaf nodes in the strictly binary tree for some k > 1. Now, we want to show that when there are k+1 leaf nodes in the tree, there are 2(k+1)-1 = 2k+1 nodes in the tree. Consider the left and right sub-trees of the root node. Let k_l and k_r denote the numbers of leaf nodes in the left and right sub-trees, respectively. Clearly, k_l + k_r = k+1. As we are considering a strictly binary tree, $1 \leq k_l \leq$ k and $1 \leq k_r \leq$ k. Of course, the left sub-tree and the right sub-tree are also strictly binary trees. Consequently, there are $2k_l$-1 and $2k_r$-1 nodes in the left and right sub-trees, respectively. Therefore, the total number of nodes in the original tree is $2k_l$-1 + $2k_r$-1 + 1 = 2k+1.