Some definitions:

- **Fibonacci Numbers**: \( f_0 = 0, f_1 = 1 \) and \( f_n = f_{n-1} + f_{n-2} \) ... some initial terms 0, 1, 1, 2, 3, 5, 8, 13, ...  
- **Binary Tree** – each parent node has at max 2 child nodes  
- **Extended Binary Tree** – defined as – an empty set is also extended binary tree. Also given \( T_1 \) and \( T_2 \) are Extended Binary Trees then \( T_1T_2 \) is also an extended Binary Tree where \( r \) is the root and \( T_1 \) and \( T_2 \) are its child.  
- **Full Binary Tree** is defined as – a single vertex is a Full Binary Tree, if \( T \) is Full Binary Tree then \( T_1T_2 \) is also Full Binary Tree if \( r \) is the root and \( T_1 \) and \( T_2 \) are the child subtrees of \( r \).  
- **Height of a tree** is defined as \( 1 + \max\{h(T_1), h(T_2)\} \) where \( T_1 \) and \( T_2 \) are left and right subtrees of root and \( h(T) \) represents the height of the tree \( T \).  

Examples:

Give recursive definitions for the set of even integers.

- **You have to take care of both +ve and –ve numbers.** So define it as 0 ∈ Set. If \( x \in \text{Set} \) then so does \( (x+2) \) and \( (x-2) \).  

Give a recursive definition of functions max and min so that max(a₁, a₂, ..., aₙ) and min(a₁, a₂, ..., aₙ) are the maximum and minimum of \( a_1, a_2, ..., a_n \) respectively.

- **Let max and min for 1 number be defined as itself.**  
- **Define max(a₁,a₂) = a₁ if a₁ >= a₂ else a₂ similarly min(a₁,a₂) = a₁ if a₁ <= a₂ else a₂.**  
- **Max(a₁, a₂, ..., aₙ₊₁) = max(max(a₁, a₂, ..., aₙ), aₙ₊₁) similarly define min.**  

Give a recursive definition of set of strings that are palindromes. A palindrome is a string that when reversed is same as original. Example aha, maam, madam, a toyota and so on.

- **We need two base cases here** \( \lambda \) (empty string) is a palindrome. Also all single characters \( s \) of the language \( L \) are palindromes. If a string \( W \) is palindrome and \( s \) is a single character in the language then \( sWs \) is also a palindrome.  

Using Induction prove that \( (w₁w₂)^r = w₂^r w₁^r \) where \( w₁w₂ \) is concatenation of \( w₁ \) and \( w₂ \) and \( w^r \) is the reverse of the string.

- **Let us first of all give a recursive definition of \( w^r \).** if \( w = \lambda \) then \( w^r = w \). Also all single character strings are their own reverse. So if \( w = w₁x \) where \( w₁ \) is a string of length 1 less than \( w \) and \( x \) is a single character then \( w^r = xw₁^r \).  

Now we will do induction on \( w₂ \) here. The base case will be \( (w₁\lambda)^r = w₁^r = \lambda w₁^r = \lambda^r w₁^r \). So this is true for the base case. For the inductive step, lets assume that \( w₂ = w₂x \) where \( w₂ \) is string of length 1 less than \( w₂ \) and \( x \) is the last symbol of \( w₂ \), then we have \( (w₁w₂)^r = (w₁w₂x)^r = x(w₁w₂)^r \) by the defn of general \( w^r \). This in turn equals \( x(w₂^r w₁^r) \) by induction hypothesis. Remember length\( (w₁) \) is less than length\( (w₂) \). Which is again \( (w₂x)^r w₁^r \) again by the definition. Finally this equals \( w₂^r w₁^r \). Proved.

**RECURSIVE ALGORITHM:**

An Algorithm is called recursive if it solves the problem by reducing it to an instance with smaller input size.

A recursive function almost always calls itself but with a smaller parameter.
An iterative program is one that makes use of various loops to solve the same problem and almost always starts with 1 and apply the recursive definition to larger integers.

Example: Discuss Merge Sort. Two operations, sort and merge.

\[ ... = \text{Sort}(...) \]  
Define Base case for recursion to end when no of element to sort is 1, that is in itself sorted.

**DISCUSS:** QUICK SORT. In quick sort no merging is required at the end, only combine.
Example:
QuickSort(3,2,5,8,1,7,9,4,6). Take first element as pivot. Pivot = 3.
So the two divided group is [2,1] and [5,8,7,9,4,6]
Solution is: QuickSort(2,1), 3, QuickSort(5,8,7,9,4,6)

Now QuickSort(2,1) pivot = 2. Call QuickSort(1), 2, QuickSort(empty).
QuickSort(1) simply returns 1. And so QuickSort(2,1) returns (1,2)

Now QuickSort(5,8,7,9,4,6)… pivot = 5. So division will be [4], 5, [8,7,9,6]
QuickSort(4) will return 4.
QuickSort(8,7,9,6) pivot = 8, groups = [6,7] , 8, [9]
QuickSort(9) returns 9.

And so on… show it by drawing tree structure.