Page No.
Date : / /

(Lecture - 40)

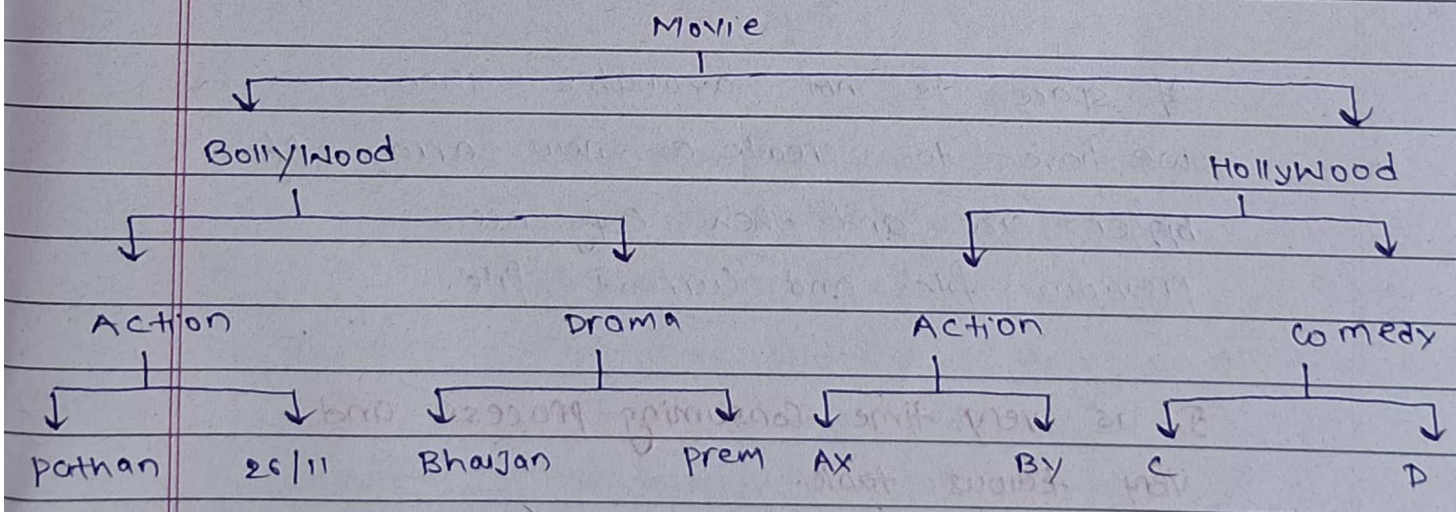
(Tree - Introduction)

Tree is a data structure which is a hierarchi-cal structure that is used to represent and organise data in a way that is easy to navigate and search.

It is a collection of nodes that are connected by edges and has a hierarchical relationship between the nodes.

Real life Example of tree :

Folders in a Computer

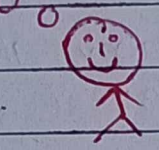


In this form, the folder is present in system. This is the real life example of Tree data structure.

Teacher's Signature.....

Page No:
Date: / /

For implementation of Tree Data structure which is more suitable for data storage? Array or Unlcedlist



Let's understand it With the example:

If we want to add a folder under the movie named (follywood) and add some movies.

using Array:

We check that space is available or not then we insert the data (movie).

If space is not available then, we have to create a new array of bigger size and then copy all previous file and current file.

It is very time Consuming process and very tdious task.

Using Unlced list:

We find a space and insert data in that space and only point the memory Wlith the previous Memory.

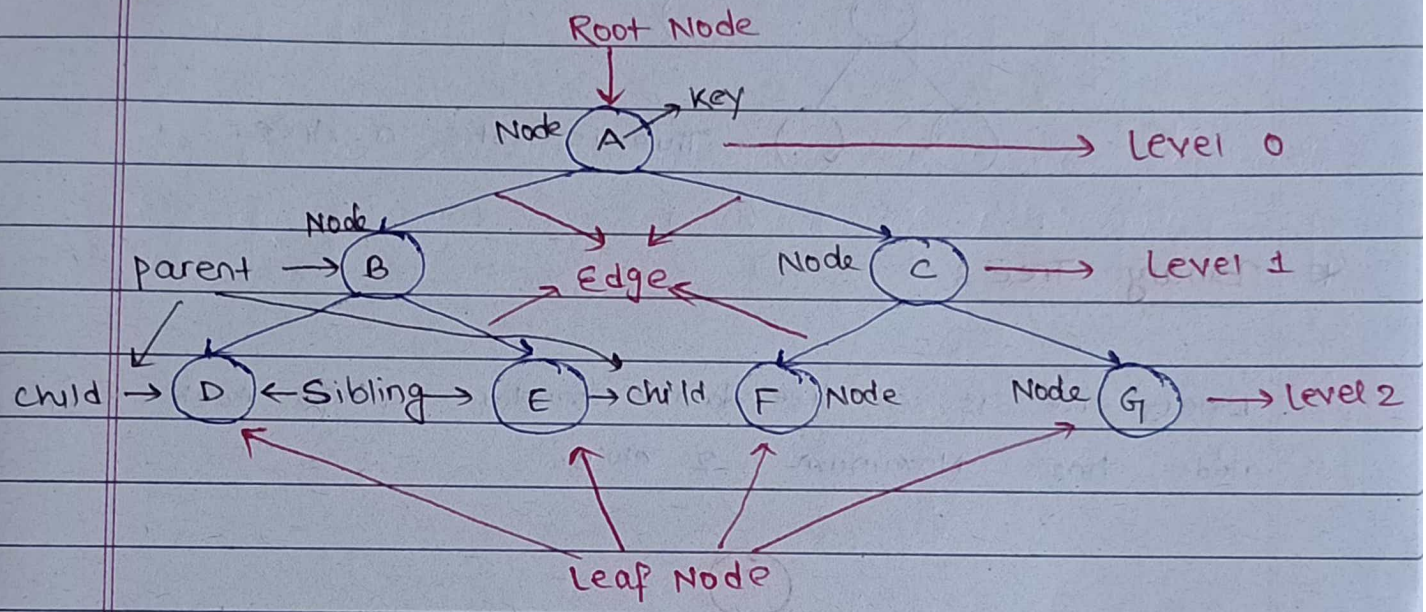
Teacher's Signature.....

Page No.
Date : / /

Everytime we need to store the data in free space and point it with previous data Node.

By linkedlist we can easily store the tree data structure.

We use linked list in tree data structure,



Parent Node : The node which is predecessor of a node is called parent node.
B is parent of D & E.

Child Node : The node which is sucessor of a node.
F & G is child of C.

Root Node : The topmost node of a tree.
A is root Node

Leaf Node : The Node which do not have any child.
D, E, F, G is Leaf Node.

Ancestor : Any precedesor node on path to the root.
For D → Ancestors are : A, B

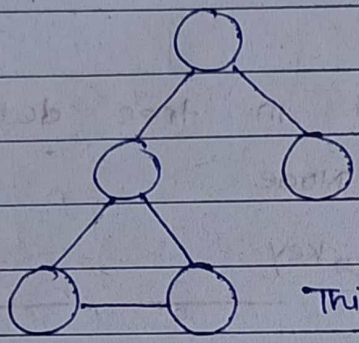
Sibling : Children of same parent nodes.
(D & E)

Teacher's Signature.....

No. of Nodes = N

then, No. of edges = N - 1

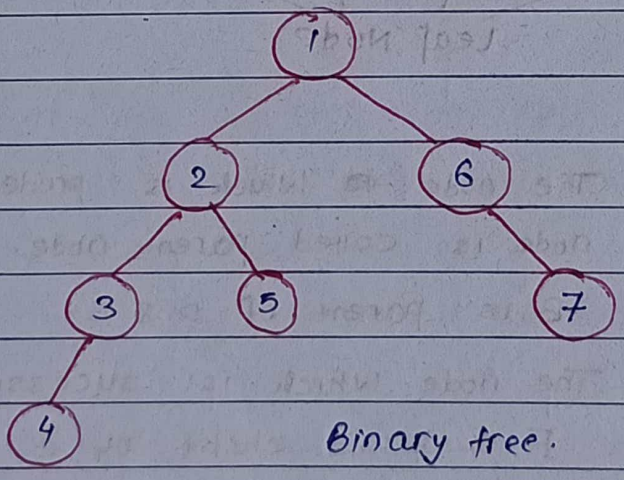
Tree cannot create closed loop



This is not a tree.

* Binary Tree :

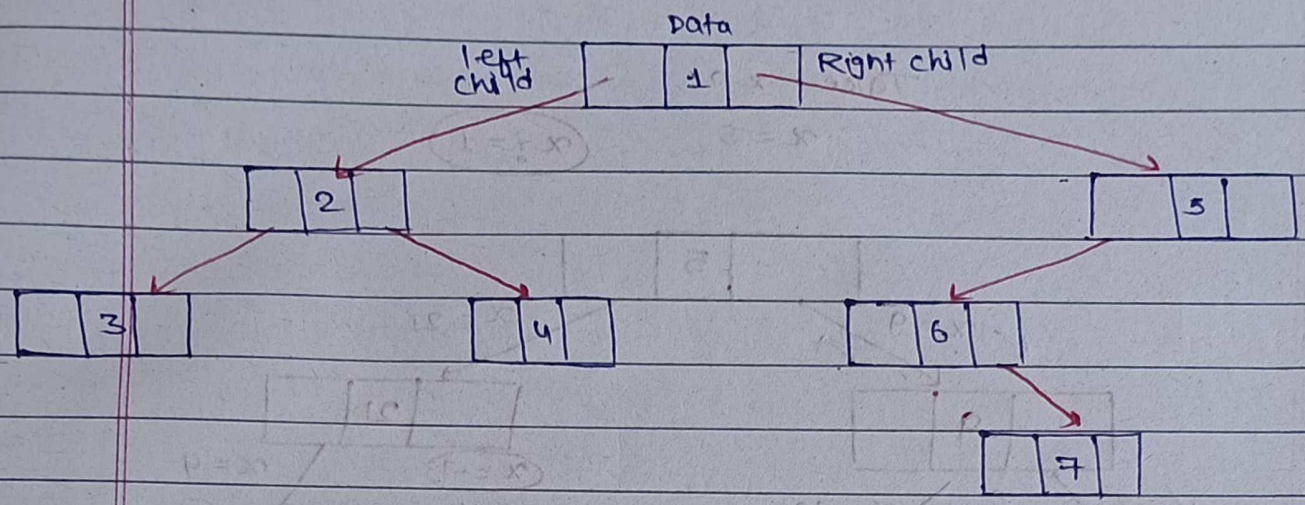
Binary tree is a tree data structure where each node has Maximum 2 child.



Binary tree.

Teacher's Signature.....

How to represent it in Memory.



Creation of Node:

```
class Node {
    public:
    int data;
    Node * left;
    Node * right;

    Node (int value) {
        data = value;
        left = NULL;
        right = NULL;
    }
}
```

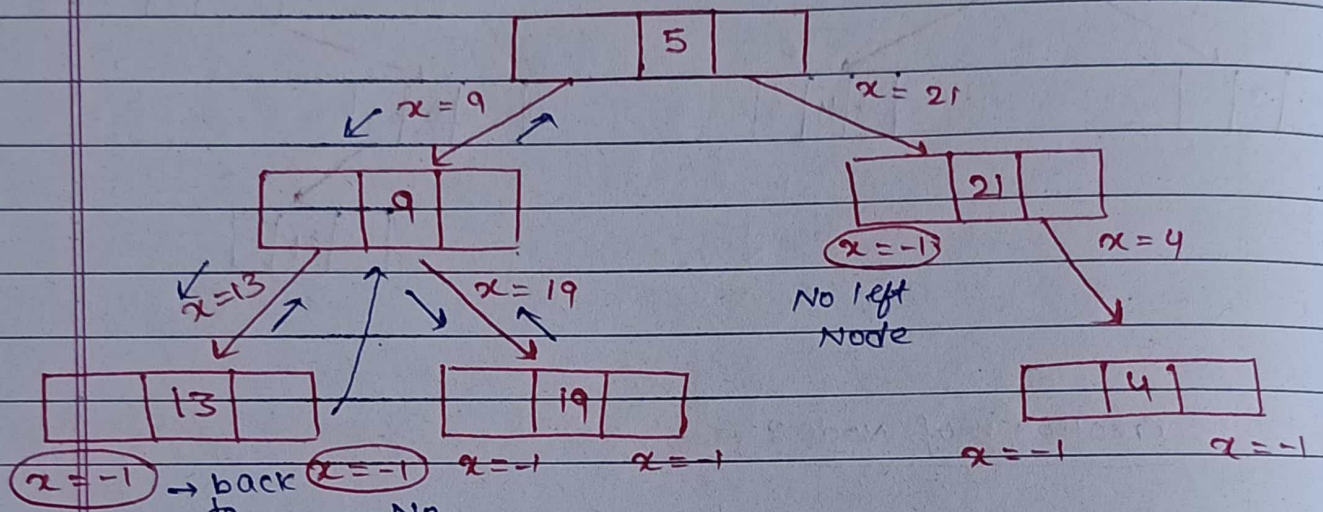
Teacher's Signature.....

Page No.:
Date: / /

Creation of Tree :

Take x ?

x = 5 (x != -1)




```
BinaryTree ( ) {
        int x;
        cin >> x;

        if (x == -1)
            return NULL;

        Node * root = new Node(x);

        root -> left = BinaryTree ( );
        root -> right = BinaryTree ( );

        return root;
}
```

Teacher's Signature.....

Page No.:
Date: / /

Code:

```
#include <iostream>
using namespace std;

class Node {
    public:
    int data;
    Node * left;
    Node * right;

    Node (int value) {
        data = value;
        left = NULL;
        right = NULL;
    }
};

Node * BinaryTreeCreation() {
    int x;
    Cout << " Enter value: ";
    cin >> x;

    if (x == -1)
    return NULL;

Node * root = new Node (x);
Cout << " Enter left child of " << x << "\n";
    root -> left = BinaryTreeCreation();
```

```
    cout << "Enter the right child of " << x << "\n";
        root -> right = BinaryTreeCreation();

    return root;
}

int main(){

    Node * root = BinaryTreeCreation();

    return 0;
}
```

On this code, we can only create the tree. But how to check that the tree is correctly created or not

For that we need to print the tree.

We have 3 traversal types which is used in trees:

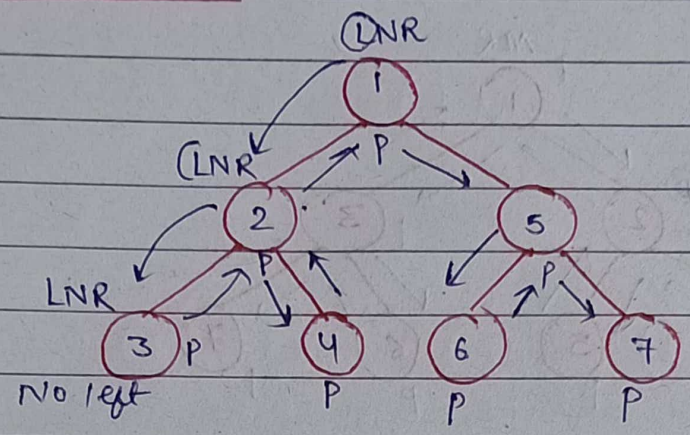
1. Inorder Traversal
2. preorder Traversal
3. postorder Traversal

Inorder : L N R
preorder : N L R
postorder : L R N

Teacher's Signature.....

Page No.:
Date: / /

Inorder Traversal



go to Node
& print

→ 3 2 4 1 6 5 7

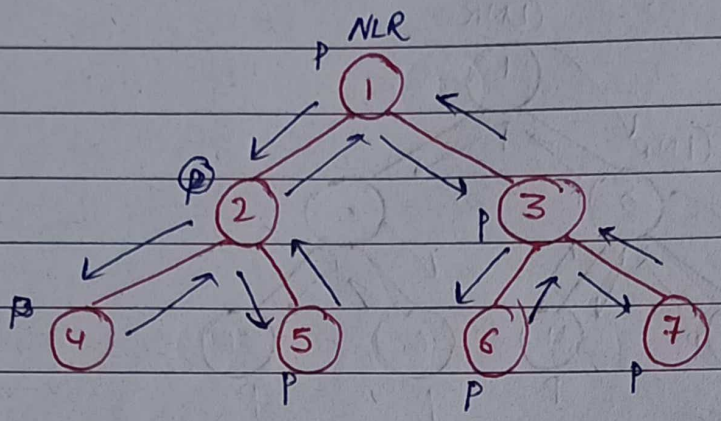
Code :

```
Void Inorder ( Node * root ) {

    if (root == NULL)
        return;

    Inorder (root → left);    // L
    Cout << root → data ;     // N
    Inorder ( root → right);  // R

}
```

Teacher's Signature.....

Page No:
Date: / /

Preorder Traversal :

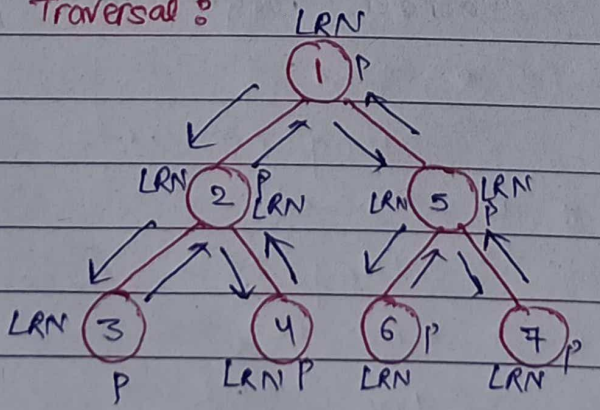


1 2 4 5 3 6 7

Code :

```
Void preorder (Node * root){
    if (root == NULL)
        return;

    Cout << root → data;      // N
    preorder ( root → left );  // L
    preorder (root → right);   // R

}
```

Postorder Traversal :



3 4 2 6 7 5 1

Teacher's Signature.....

Code :

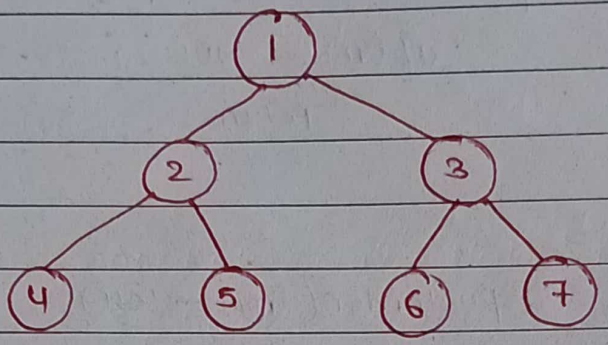
```
Void postorder (Node * root) {
        if (root = NULL) {
            return;
        }

        postorder ( root → left);    // L
        postorder ( root → right);   // R
        Cout << root → data;         // N
    }
```

If we wont to store -1 in the tree then.
first we have to ask :
whether you want to create Node or Not.
if yes (1)
or No (0) } boolean.

```
int main () {
        Node * root = BinaryTreeCreation ();
        Cout << " Inorder Traversal : ";
        Inorder (root);

        Cout << " preorder Traversal : ";
        preorder (root);

        Cout << " postorder Traversal : ";
        postorder (root);

        return 0;
}
```

Teacher's Signature.....

→ **Level order Traversal :**



→ 1 2 3 4 5 6 7

We dont print it by recursion.

↳ Because Level order Traversal follow breadth first search (BFS).

↳ And recursion follow the depth first search (DFS).

In level order traversal, it uses Queue for its Traversal.

q = | 2, 3, 4, 5, 6, 7

(1) print
↳ said (2, 3) को print कर दो

(2) print
↳ said 4, 5 को print

(3) print
↳ said 6, 7 को print

(4) print

(5) print

(6) print

(7) print

Teacher's Signature.....

→ 1 2 3 4 5 6 7

Code :

```
Void LevelorderTraversal (Node * root) {

    if (root == NULL)
        return;

    queue <Node *> q;
    q.push(root);

    while ( ! q.empty()) {
    Node * temp = q.front();
    q.pop();
    cout << temp→data;

    if (temp→left)
      q.push (temp→left);

    if (temp→right)
    q.push (temp→right);

}
```

Teacher's Signature.....

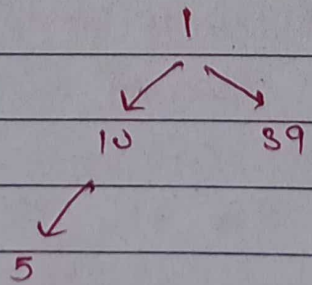
Page No.:
Date: / /

problem 1: Inorder Traversal (GFG)

code:

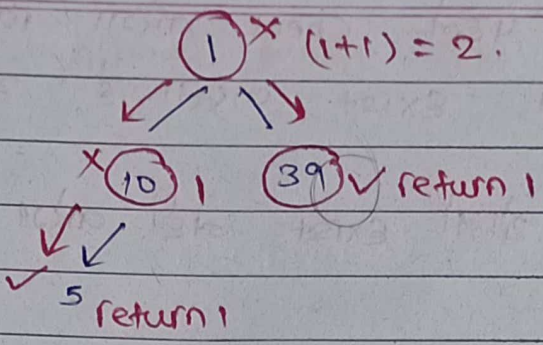
```
class solution {
    public:
    Void inorderTraversal (Node * root, Vector <int> ans){
            if (root == NULL)
                return;

        inorderTraversal (root → left, ans);
         Cout << root → data << " ";
        inorderTraversal (root → right, ans);
    }

Vector <int> inOrder (Node * root) {
        Vector <int> ans;
        inorderTraversal (root, ans);
      return ans;
    }
};
```

problem 2: Count leaves in Binary tree



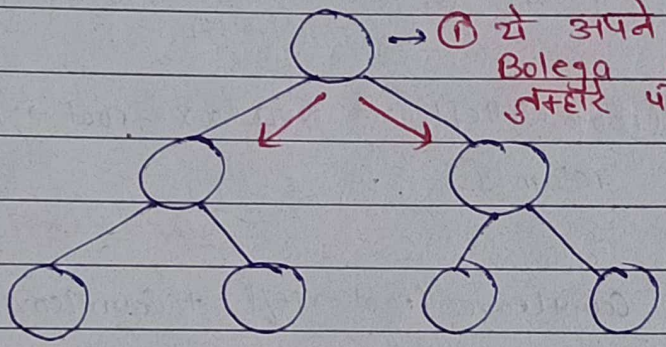
leaves: Node having No child.

Teacher's Signature.....

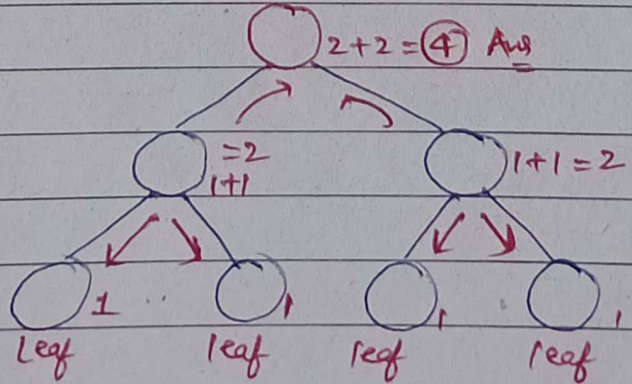



```
root → left == NULL && root → right == NULL
        return 1;
```

return क्या करना है।



② left & right dono check करेगा। क्या इस leaf Node है।
अगर होगा तो return 1.
नही तो वो अपने left or right वाले को बोलेगा कि leaf Node count kar ke do.



Page No.:
Date: / /

हर बार वो पहले check करेगा कि उसका left or right Exist करता है या नहीं।

left or right दोनों Exist नहीं करेगा तो return 1.

Code :

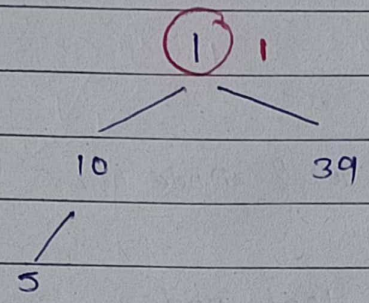
```
int CountLeaves(Node * root) {

    if (root == NULL)
        return 0;

    if (root → left == NULL && root → right == NULL)
        return 1;

    return CountLeaves(root → left) + CountLeaves(root → right);
}
```

Traverse : T.C → O(N)

create : T.C → O(N)
S.C → O(N)

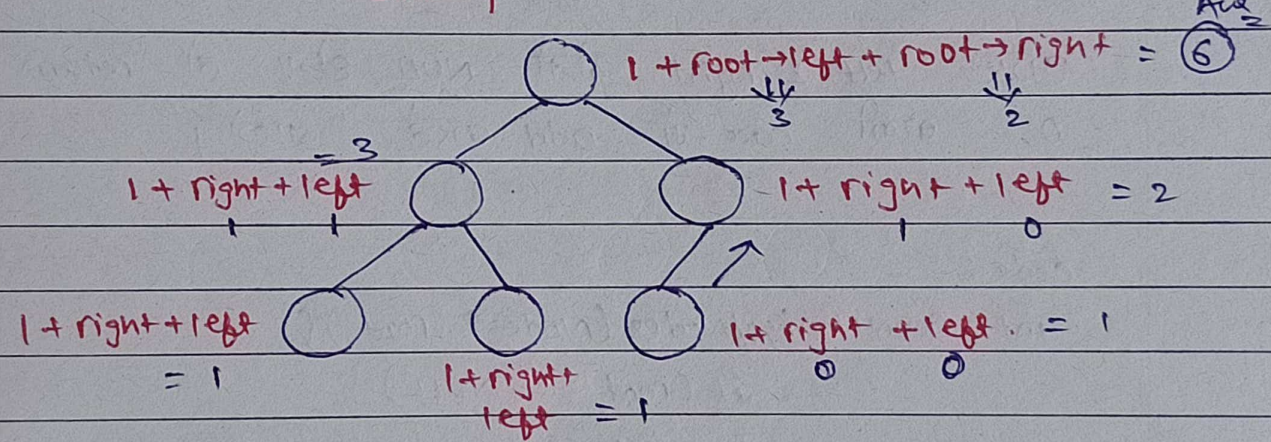
Teacher's Signature.....

* Problem 3 : Size of Binary Tree :



Count No. of Nodes.

① root node बोलेगा अपने left or right से अपने sub tree को count करने को बोलो। और उसमें 1 add कर देगा अपना।



Code :

```
if (root == NULL)
    return 0;

return 1 + getsize(root→left) + getSize(root→right);
```

Problem 4: Count Non-leaf Nodes in Tree

Approach 1

Step ① Count total node — ①

Step ② Count leaf node — ②

Step ③ → ① - ②

Approach 2

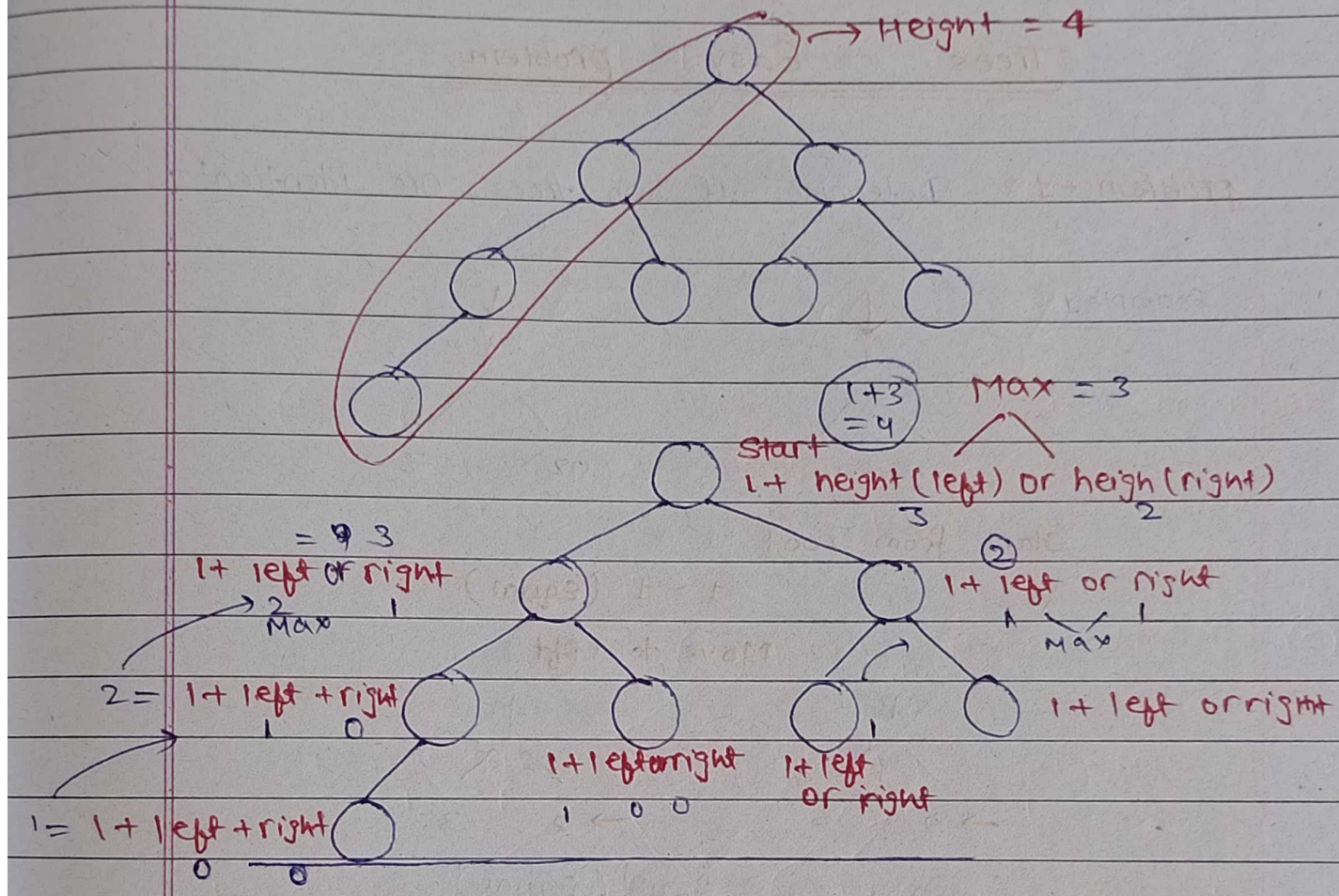
अगर left & right दोनों NULL होगा तो return 0, बाकी case में add करते जाएंगे।

Code:

```
int CountNonLeafNodes (Node * root ){
        if (root == NULL)
            return 0;

        if (root→left == NULL && root→right == NULL)
            return 0;

return 1+ CountNonLeafNodes (root→left) + CountNonLeafNodes
                                        (root→right);
}
```

Teacher's Signature.....

Problem 5 : Height of Binary tree



Code :-

```
int height(struct Node * root) {
    if (root == NULL)
        return 0;

    if (root → left == NULL && root → right == NULL)
        return 1;

    return 1 + max(height(root → left), height(root → right));

}
```