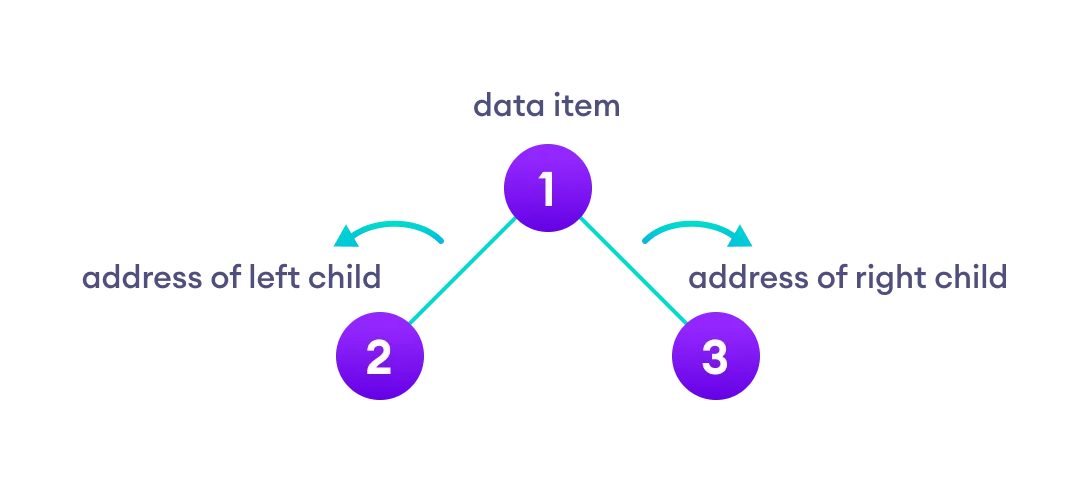
**LAB SESSION 7: BINARY TREES**

**AIM**: To implement Binary trees and perform the listed operations on such trees.

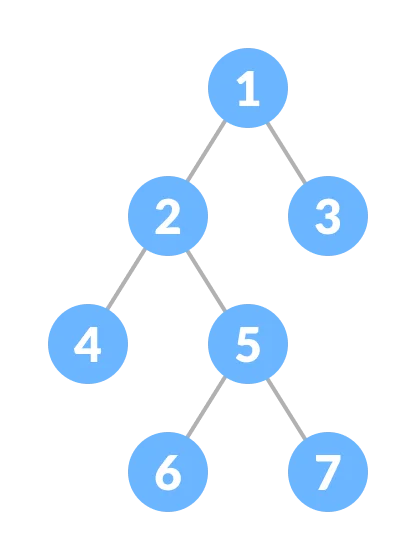
**PROBLEM DEFINITION:**

Develop C program to create a binary tree given its INORDER and POSTORDER traversal. Provide options to the user to perform the following operations on the binary tree:

1. Display height of the tree
2. Return the depth of a given node in the tree
3. Perform level order traversal
4. Perform Spiral order traversal

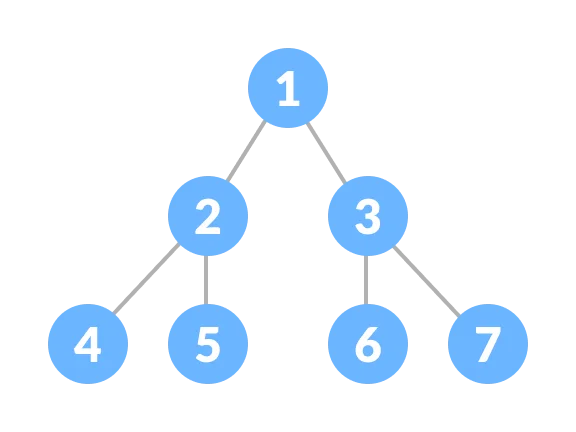
**THEORY:** A binary tree is a tree data structure in which each parent node can have at most two children. Each node of a binary tree consists of three items:

* data item
* address of left child
* address of right child

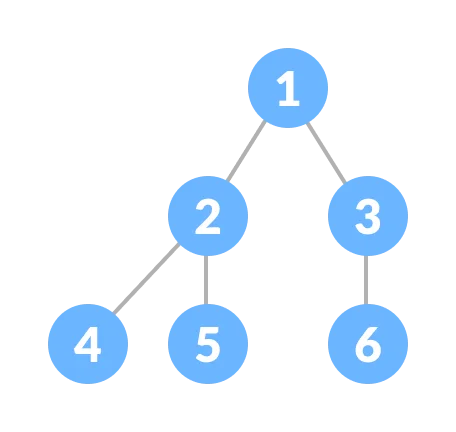
*Types of Binary Tree*

**1. Full Binary Tree**

A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children.

2. **Perfect Binary Tree**

A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.

**3. Complete Binary Tree**

A complete binary tree is just like a full binary tree, but with two major differences

* Every level must be completely filled
* All the leaf elements must lean towards the left.
* The last leaf element might not have a right sibling i.e. a complete binary tree doesn't have to be a full binary tree.

**Tree Traversals**

*Inorder traversal*

* First, visit all the nodes in the left subtree
* Then the root node
* Visit all the nodes in the right subtree
  + - inorder(root->left)
    - display(root->data)
    - inorder(root->right)

*Preorder traversal*

* Visit root node
* Visit all the nodes in the left subtree
* Visit all the nodes in the right subtree
  + - display(root->data)
    - preorder(root->left)
    - preorder(root->right)

*Postorder traversal*

* Visit all the nodes in the left subtree
* Visit all the nodes in the right subtree
* Visit the root node
  + - postorder(root->left)
    - postorder(root->right)
    - display(root->data)