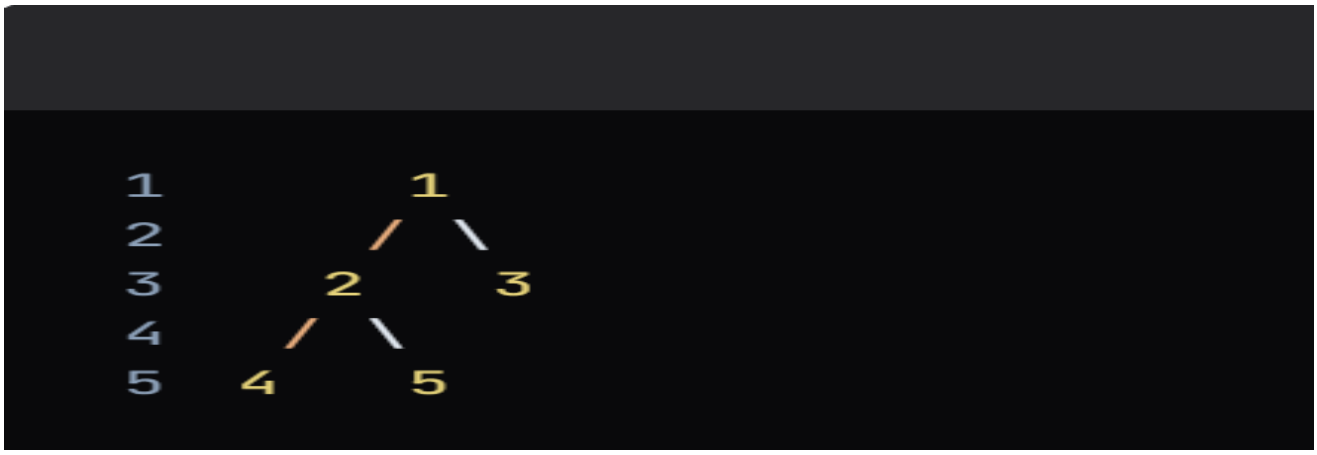
# Practical 10

**Aim : Write a Program to Understand and implement Tree traversals i.e. Pre-Order Post-Order, In-OrderAlgorithm**

## Traversal Methods:




- Pre-order Traversal:
  - The method visits the root node first, then recursively visits the left subtree, followed by the right subtree.
  - The output for the given tree will be: 1 2 4 5 3.
- In-order Traversal:
  - The method visits the left subtree first, then the root node, and finally the right subtree.
  - The output for the given tree will be: 4 2 5 1 3.
- Post-order Traversal:
  - The method visits the left subtree first, then the right subtree, and finally the root node.
  - The output for the given tree will be: 4 5 2 31

## Code:-

```
public class TreeTraversals {
    // Node class representing a node in the tree
    static class Node {
        int data;
        Node left, right;

        Node(int data) {
            this.data = data;
            this.left = null;
            this.right = null;
        }
    }

    // Main class with tree creation and traversal methods
    public static void main(String[] args) {
        // Create a sample binary tree
        Node root = new Node(1);
        root.left = new Node(2);
        root.right = new Node(3);
        root.left.left = new Node(4);
        root.left.right = new Node(5);

        // Pre-order traversal (Root -> Left -> Right)
        System.out.print("Pre-order traversal: ");
        preOrderTraversal(root);
```

```
        System.out.println();

        // In-order traversal (Left -> Root -> Right)
        System.out.print("In-order traversal: ");
        inOrderTraversal(root);
        System.out.println();

        // Post-order traversal (Left -> Right -> Root)
        System.out.print("Post-order traversal: ");
        postOrderTraversal(root);
        System.out.println();
    }

    // Pre-order traversal method
    public static void preOrderTraversal(Node root) {
        if (root == null) {
            return;
        }
        System.out.print(root.data + " "); // Visit root first
        preOrderTraversal(root.left); // Then visit left subtree
        preOrderTraversal(root.right); // Then visit right subtree
    }

    // In-order traversal method
    public static void inOrderTraversal(Node root) {
        if (root == null) {
```

```
        return;
    }
    inOrderTraversal(root.left); // Visit left subtree first
    System.out.print(root.data + " "); // Then visit root
    inOrderTraversal(root.right); // Then visit right subtree
  }

  // Post-order traversal method
  public static void postOrderTraversal(Node root) {
    if (root == null) {
        return;
    }
    postOrderTraversal(root.left); // Visit left subtree first
    postOrderTraversal(root.right); // Then visit right subtree
    System.out.print(root.data + " "); // Then visit root
  }
}
```

## Output:-

```
redhat.java\jdt_ws\CC Program_b79456da\bin' 'TreeTraversals
Pre-order traversal: 1 2 4 5 3
In-order traversal: 4 2 5 1 3
Post-order traversal: 4 5 2 3 1
PS C:\Users\HP\OneDrive\Desktop\CC Program>
```