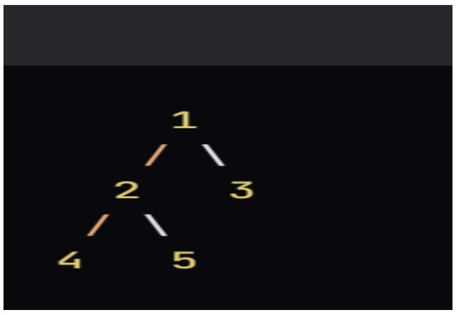
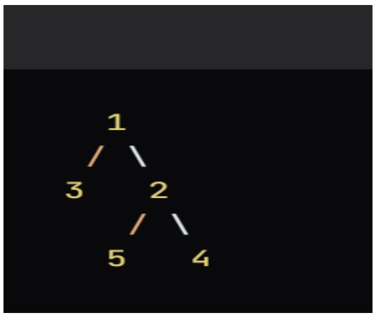
Practical 11

Aim : Write a Program to verify and validate mirrored trees or not.



```
      1
     / \
    2   3
   / \
  4   5
```



```
    1
   / \
  3   2
     / \
    5   4
```

Mirrored trees:-

To determine if two binary trees are mirrored, you need to check the following conditions:

1. Both Trees are Empty: If both trees are None (empty), they are considered mirrored.
2. One Tree is Empty: If one tree is None and the other is not, they are not mirrored.
3. Node Values: The values of the current nodes in both trees must be the same.
4. Children Comparison: The left child of the first tree must be compared with the right child of the second tree, and the right child of the first tree must be compared with the left child of the second tree. This means:
    - If tree1 has a left child, it should match with tree2's right child.

- **If tree1 has a right child, it should match** with tree2's left child.

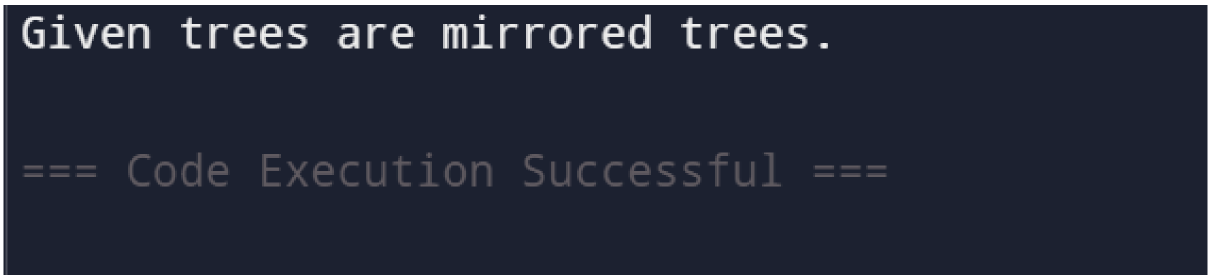
Algorithm :

1. Node Structure:
    - Define a Node class with attributes for data, left, and right.
2. Function areMirror(root1, root2):
    - Base Case:
        - If both root1 and root2 are null, return true.
        - If one is null and the other is not, return false.
    - Check Values:
        - If root1.data is not equal to root2.data, return false.
    - Recursive Check:
        - Return the result of:
            - areMirror(root1.left, root2.right) AND
            - areMirror(root1.right, root2.left).
3. Main Method:
    - Create two binary trees.
    - Call areMirror(root1, root2) and print the result.

Program :-

```
public class Main {
```

```
// 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;
    }
}

// Method to check if two trees are mirror images
public static boolean areMirror(Node root1, Node root2) {
    // Base case: Both trees are empty
    if (root1 == null && root2 == null) {
        return true;
    }
    // One tree is empty and the other is not
    if (root1 == null || root2 == null) {
        return false;
    }
    // Check if root data matches and subtrees are mirrors
    return root1.data == root2.data &&
           areMirror(root1.left, root2.right) &&
           areMirror(root1.right, root2.left);
    }

    // Main method for creating sample trees and checking if they are mirrored
    public static void main(String[] args) {
        // Create mirrored trees
        Node root1 = new Node(1);
        root1.left = new Node(2);
        root1.right = new Node(3);
        root1.left.left = new Node(4);
        root1.left.right = new Node(5);

        Node root2 = new Node(1);
        root2.left = new Node(3);
        root2.right = new Node(2);
        root2.right.left = new Node(5);
        root2.right.right = new Node(4);

        // Check if the trees are mirror images
        boolean mirrored = areMirror(root1, root2);
        if (mirrored) {
            System.out.println("Given trees are mirrored trees.");
        } else {
            System.out.println("Given trees are not mirrored trees.");
        }
    }
}
```



```
Given trees are mirrored trees.


=== Code Execution Successful ===
```