**BST Questions**

**1- Check the Structural Identical BST**

**Code:**

package BST;

public class StructuralTree {

    static class Node {

        int data;

        Node left;

        Node right;

    };

    // Utility function to create a new Node

    static Node newNode(int data) {

        Node node = new Node();

        node.data = data;

        node.left = null;

        node.right = null;

        return node;

    }

    // Function to perform inorder traversal

    static void inorder(Node root) {

        if (root == null)

            return;

        inorder(root.left);

        System.out.print(root.data + " ");

        inorder(root.right);

    }

    // Function to check if two BSTs

    // are identical

    static int isIdentical(Node root1,

            Node root2) {

        // Check if both the trees are empty

        if (root1 == null && root2 == null)

            return 1;

        // If any one of the tree is non-empty

        // and other is empty, return false

        else if (root1 != null &&

                root2 == null)

            return 0;

        else if (root1 == null &&

                root2 != null)

            return 0;

        else {

            // Check if current data of both trees equal

            // and recursively check for left and right subtrees

            if (root1.data == root2.data &&

                    isIdentical(root1.left, root2.left) == 1 &&

                    isIdentical(root1.right, root2.right) == 1)

                return 1;

            else

                return 0;

        }

    }

    // Driver code

    public static void main(String[] args) {

        Node root1 = newNode(5);

        Node root2 = newNode(5);

        root1.left = newNode(3);

        root1.right = newNode(8);

        root1.left.left = newNode(2);

        root1.left.right = newNode(4);

        root2.left = newNode(3);

        root2.right = newNode(8);

        root2.left.left = newNode(2);

        root2.left.right = newNode(4);

        if (isIdentical(root1, root2) == 1)

            System.out.print("Both BSTs are identical");

        else

            System.out.print("BSTs are not identical");

    }

}

**2- Create a Print Function , To Print like a BST**

**Code:**

package BST;

class Trunk {

    Trunk prev;

    String str;

    Trunk(Trunk prev, String str) {

        this.prev = prev;

        this.str = str;

    }

};

// A Binary Tree Node

class Node {

    int data;

    Node left, right;

    Node() {

    }

    Node(int data) {

        this.data = data;

        this.left = this.right = null;

    }

}

class Main {

    public static void showTrunks(Trunk p) {

        if (p == null) {

            return;

        }

        showTrunks(p.prev);

        System.out.print(p.str);

    }

    public static void printTree(Node root, Trunk prev, boolean isLeft) {

        if (root == null) {

            return;

        }

        String prev\_str = "    ";

        Trunk trunk = new Trunk(prev, prev\_str);

        printTree(root.right, trunk, true);

        if (prev == null) {

            trunk.str = "———";

        } else if (isLeft) {

            trunk.str = ".———";

            prev\_str = "   |";

        } else {

            trunk.str = "`———";

            prev.str = prev\_str;

        }

        showTrunks(trunk);

        System.out.println(" " + root.data);

        if (prev != null) {

            prev.str = prev\_str;

        }

        trunk.str = "   |";

        printTree(root.left, trunk, false);

    }

    public static void main(String[] args) {

        // Construct above 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);

        root.right.left = new Node(6);

        root.right.right = new Node(7);

        root.left.left.left = new Node(8);

        root.left.left.right = new Node(9);

        root.left.right.left = new Node(10);

        root.left.right.right = new Node(11);

        root.right.left.left = new Node(12);

        root.right.left.right = new Node(13);

        root.right.right.left = new Node(14);

        root.right.right.right = new Node(15);

        // print constructed binary tree

        printTree(root, null, false);

    }

}