Daily Practice Program Date 29-11-2023

Program 1: Insert into a Binary Search Tree

```
package ishwarchavan.com;

import java.io.*;
import java.util.*;

public class InsertionIntoBST {     //class created

	public static void main(String[] args){   //Main program started
		BST tree = new BST();  //object creation
		tree.insert(10);
		tree.insert(20);
		tree.insert(30);
		tree.insert(40);
		tree.insert(50);
		tree.insert(60);
		tree.inorder();  //function calling
	}
}

class Node {   //node creation
	Node left;
	int val;
	Node right;
	Node(int val) { this.val = val; }
}

class BST {
	Node root;

	public void insert(int key) {    // Function to insert a key

		Node node = new Node(key);
		if (root == null) {   //condition checking
			root = node;
			return;
		}
		Node prev = null;
		Node temp = root;
		while (temp != null) {
			if (temp.val > key) {   //if true then execute below statement
				prev = temp;
				temp = temp.left;
			}
			else if (temp.val < key) {    //other wise execute
				prev = temp;
				temp = temp.right;
			}
		}
		if (prev.val > key)
			prev.left = node;
		else
			prev.right = node;
	}
	public void inorder(){     // Function to print the inorder value
		Node temp = root;
		Stack<Node> stack = new Stack<>();
		while (temp != null || !stack.isEmpty()) {   //concdition checking
			if (temp != null) {   //if true then execute
				stack.add(temp);
				temp = temp.left;
			}
			else {            //otherwise execute
				temp = stack.pop();
```

```
                        System.out.print(temp.val + " ");
                        temp = temp.right;
                    }
                }
        }
}
```

Program 2: Duplicate Subtrees

```
package ishwarchavan.com;
import java.util.HashMap;

public class DuplicateSubtrees{    //class created

        static HashMap<String, Integer> m;
        static class Node {   //node creation
                int data;
                Node left;
                Node right;
                Node(int data){
                        this.data = data;
                        left = null;
                        right = null;
                }
        }
        static String inorder(Node node){   //function created
                if (node == null)
                        return "";

                String str = "(";
                str += inorder(node.left);
                str += Integer.toString(node.data);   //type casting
                str += inorder(node.right);
                str += ")";

                if (m.get(str) != null && m.get(str)==1 )   //condition checking
                        System.out.print( node.data + " ");

                if (m.containsKey(str))
                        m.put(str, m.get(str) + 1);
                else
                        m.put(str, 1);
                return str;
        }

        static void printAllDups(Node root) {     // Wrapper over inorder()

                m = new HashMap<>();
                inorder(root);   //function calling
        }

        public static void main(String args[]){      //main program started
                Node root = null;
                root = new Node(1);
                root.left = new Node(2);
                root.right = new Node(3);
                root.left.left = new Node(4);
                root.right.left = new Node(2);
                root.right.left.left = new Node(4);
                root.right.right = new Node(4);
                printAllDups(root);
        }
```

```
}
```