Aayush Anand

20BCE7393

L21-22+L27-28

**ASSIGNMENT – 11**

Given a Binary Search Tree. Convert a given BST into a Special Max Heap with the condition that all the values in the left subtree of a node should be less than all the values in the right subtree of the node. This condition is applied on all the nodes in the so converted Max Heap.

Sample Input :

4

/ \

2 6

/ \ / \

1 3 5 7

Output: 1 2 3 4 5 6 7

Explanation:

7

/ \

3 6

/ \ / \

1 2 4 5

The given BST has been transformed into a Max Heap and its post-order traversal is:

1 2 3 4 5 6 7.

**Your task:**

You don't need to read input or print anything.

Input: root of the tree

Action: BST to max heap.

A method is required to print the post order traversal of the converted Binary search tree.

**CODE:**

import java.util.\*;

public class BSTtoMaxHeap {

    static int i;

    static class Node {

        int data;

        Node left, right;

        public Node(int data){

            this.data = data;

            this.left = this.right = null;

        }

    }

    static void inOrder(Node root, Vector<Integer> tree) {

        if (root == null)

            return;

        inOrder(root.left, tree);

        tree.add(root.data);

        inOrder(root.right, tree);

    }

    static void MaxHeap(Node root, Vector<Integer> tree) {

        if (root == null)

            return;

        MaxHeap(root.left, tree);

        MaxHeap(root.right, tree);

        root.data = tree.get(i++);

    }

    static void convert(Node root) {

        Vector<Integer> tree = new Vector<Integer>();

        inOrder(root, tree);

        MaxHeap(root, tree);

    }

    static void postOrder(Node root) {

        if (root == null)

            return;

        postOrder(root.left);

        postOrder(root.right);

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

    }

    public static void main(String[] args) {

        Node root = new Node(4);

        root.left = new Node(2);

        root.right = new Node(6);

        root.left.left = new Node(1);

        root.left.right = new Node(3);

        root.right.left = new Node(5);

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

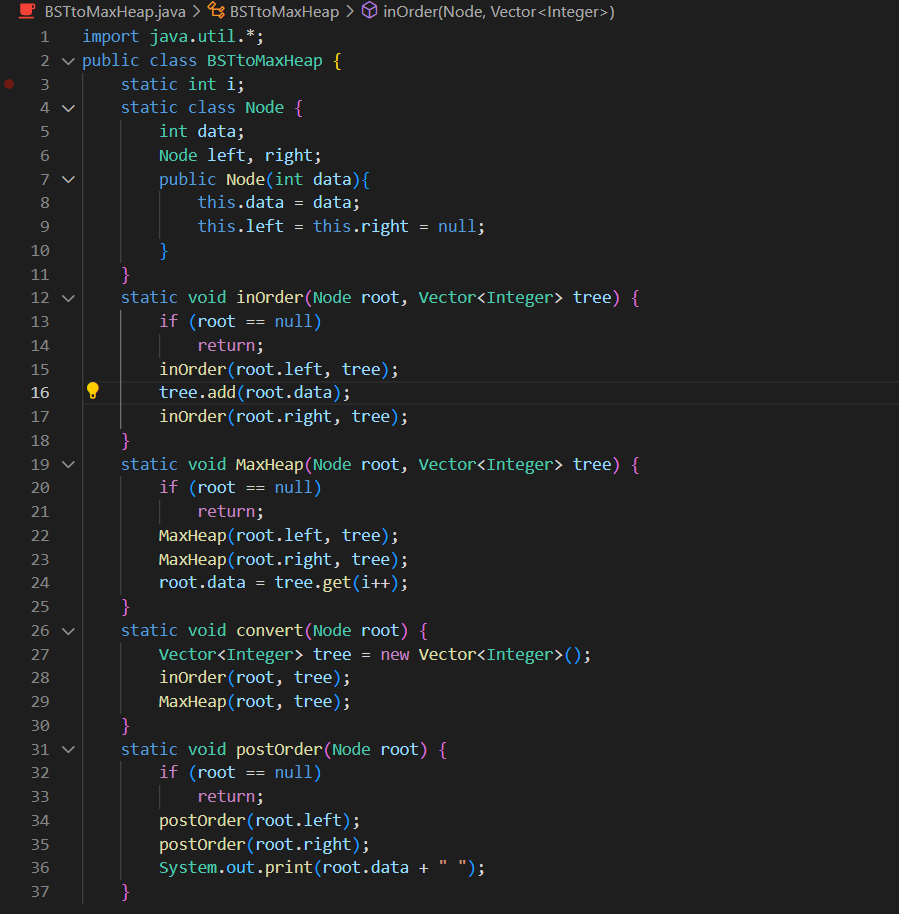
        convert(root);

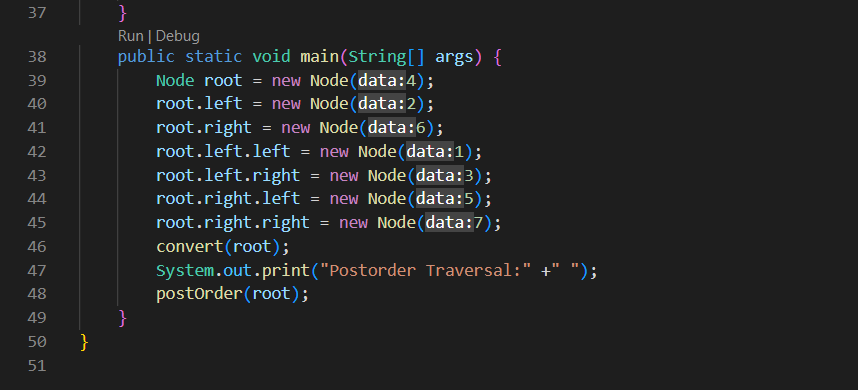
        System.out.print("Postorder Traversal:" +" ");

        postOrder(root);

    }

}

****

****

**OUTPUT:**

