// Java implementation to flatten the

// binary search tree into a skewed

// tree in increasing order

import java.io.\*;

// Class of the node

class Node

{

    int val;

    Node left, right;

    // Helper function that allocates a new node

    // with the given data and NULL left and right

    // pointers.

    Node(int item)

    {

        val = item;

        left = right = null;

    }

}

class Transactions

{

    public static Node node;

    static Node prevNode = null;

    static Node headNode = null;

    // Function to to flatten the binary

    // search tree into a skewed tree in

    // increasing / decreasing order

    static void flattenBTToSkewed(Node root,

                                  int order)

    {

        // Base Case

        if(root == null)

        {

            return;

        }

        // Condition to check the order

        // in which the skewed tree to

        // maintained

        if(order > 0)

        {

            flattenBTToSkewed(root.right, order);

        }

        else

        {

            flattenBTToSkewed(root.left, order);

        }

        Node rightNode = root.right;

        Node leftNode = root.left;

        // Condition to check if the root Node

        // of the skewed tree is not defined

        if(headNode == null)

        {

            headNode = root;

            root.left = null;

            prevNode = root;

        }

        else

        {

            prevNode.right = root;

            root.left = null;

            prevNode = root;

        }

        // Similarly recurse for the left / right

        // subtree on the basis of the order required

        if (order > 0)

        {

            flattenBTToSkewed(leftNode, order);

        }

        else

        {

            flattenBTToSkewed(rightNode, order);

        }

    }

    // Function to traverse the right

    // skewed tree using recursion

    static void traverseRightSkewed(Node root)

    {

        if(root == null)

        {

            return;

        }

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

        traverseRightSkewed(root.right);

    }

    // Driver Code

    public static void main (String[] args)

    {

       //    50

       //   / \

      //  30  60

// / /

// 10 55

        Transactions tree = new Transactions();

        tree.node = new Node(50);

        tree.node.left = new Node(30);

tree.node.left.left = new Node(10);

        tree.node.right = new Node(60);

tree.node.right.left = new Node(55);

        // Order of the Skewed tree can

        // be defined as follows -

        // For Increasing order - 0

        // For Decreasing order - 1

        int order = 0;

        flattenBTToSkewed(node, order);

        traverseRightSkewed(headNode);

    }

}