Assignment 23 Solution - Heaps and Hashing| DSA

Question-1:

Given preorder of a binary tree, calculate its [depth(or height)](https://www.geeksforgeeks.org/write-a-c-program-to-find-the-maximum-depth-or-height-of-a-tree/) [starting from depth 0]. The preorder is given as a string with two possible characters.

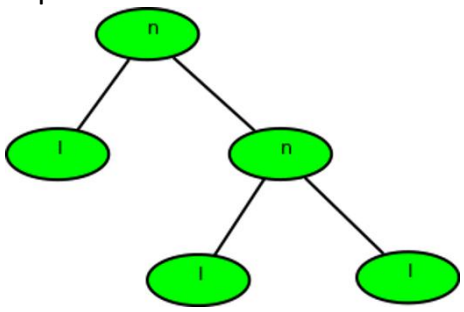
1. 'l' denotes the leaf
2. 'n' denotes internal node

The given tree can be seen as a full binary tree where every node has 0 or two children. The two children of a node can 'n' or 'l' or mix of both.

Examples :

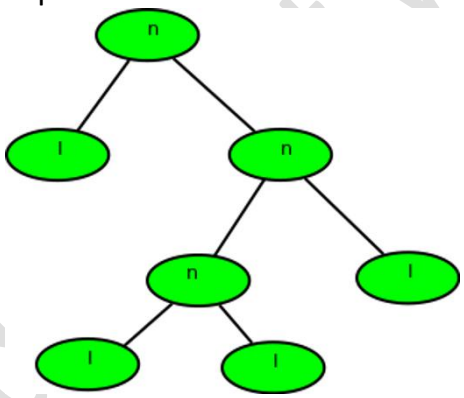
Input : nlnll
Output : 2

Explanation :



Input : nlnnlll
Output : 3

Explanation:



Solution Code:

```
package in.ineuron.pptAssignment23;

public class BinaryTreeDepth_1 {

	private static int findDepth(String preorder) {
		int depth = 0;
		int n = preorder.length();
```

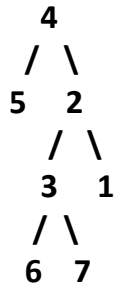
```
            for (int i = 0; i < n; i++) {
                if (preorder.charAt(i) == 'n') {
                    depth++;
                }
            }

            return depth;
        }

        public static void main(String[] args) {
            String preorder = "nlnll";
            int depth = findDepth(preorder);

            System.out.println("Depth of the binary tree: " + depth);
        }
    }
```

Question-2:

Given a Binary tree, the task is to print the left view of the Binary Tree. The left view of a Binary Tree is a set of leftmost nodes for every level.

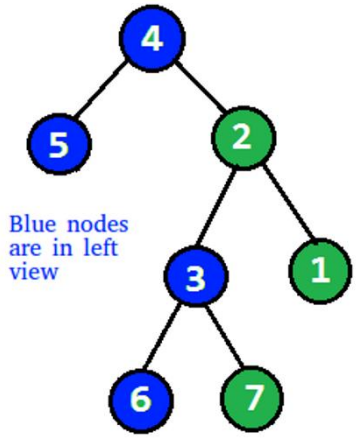
Examples:

Input:

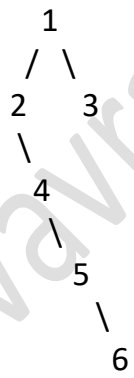


Output: 4 5 3 6

Explanation:



Input:



Output: 1 2 4 5 6

Solution Code:

```
package in.ineuron.pptAssignment23;

import java.util.*;

//Binary Tree node
class Node {
	int data;
	Node left, right;

	public Node(int item) {
		data = item;
		left = right = null;
	}
}

public class BinaryTree_2 {
	Node root;
	static int maxLevel = 0;

	// Recursive function to print the left view of a binary tree
	void leftViewUtil(Node node, int level) {
		if (node == null)
			return;

		// If this is the first node of its level
		if (maxLevel < level) {
			System.out.print(node.data + " ");
			maxLevel = level;
		}

		// Recur for the left and right subtrees
		leftViewUtil(node.left, level + 1);
		leftViewUtil(node.right, level + 1);
	}

	// Main function to print the left view of a binary tree
	void leftView() {
		leftViewUtil(root, 1);
	}

	// Driver code
	public static void main(String args[]) {
		/*
		 * Constructed binary tree is: 4 / \ 5 2 / \ 3 1 / \ 6 7
		 */
```

```
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(4);
        tree.root.left = new Node(5);
        tree.root.right = new Node(2);
        tree.root.right.left = new Node(3);
        tree.root.right.right = new Node(1);
        tree.root.right.left.left = new Node(6);
        tree.root.right.left.right = new Node(7);

        System.out.println("Left view of the binary tree:");
        tree.leftView();
    }
}
```

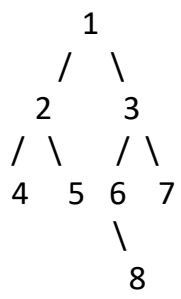
Question-3:

Given a Binary Tree, print the Right view of it.

The right view of a Binary Tree is a set of nodes visible when the tree is visited from the Right side.

Examples:

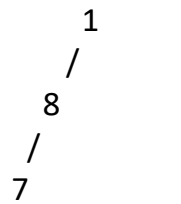
Input:



Output:

Right view of the tree is 1 3 7 8

Input:



Output:

Right view of the tree is 1 8 7

Solution Code:

```
package in.ineuron.pptAssignment23;

import java.util.*;

//Binary Tree node
class Node03 {
	int data;
	Node03 left, right;

	public Node03(int item) {
		data = item;
		left = right = null;
	}
}
```

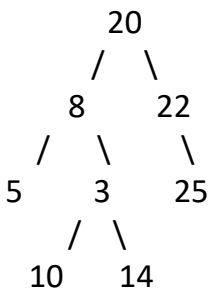
```
public class BinaryTree_3 {
    Node03 root;
    static int maxLevel = 0;
    // Recursive function to print the right view of a binary tree
    void rightViewUtil(Node03 node, int level) {
        if (node == null)
            return;

        // If this is the first node of its level
        if (maxLevel < level) {
            System.out.print(node.data + " ");
            maxLevel = level;
        }

        // Recur for the right and left subtrees
        rightViewUtil(node.right, level + 1);
        rightViewUtil(node.left, level + 1);
    }
    // Main function to print the right view of a binary tree
    void rightView() {
        rightViewUtil(root, 1);
    }
    // Driver code
    public static void main(String args[]) {
        /*
         * Constructed binary tree is: 1
                                       / \
                                      2   3
                                     / \ / \
                                     4 5 6 7
                                           \
                                           8
        */
        BinaryTree_3 tree = new BinaryTree_3();
        tree.root = new Node03(1);
        tree.root.left = new Node03(2);
        tree.root.right = new Node03(3);
        tree.root.left.left = new Node03(4);
        tree.root.left.right = new Node03(5);
        tree.root.right.left = new Node03(6);
        tree.root.right.right = new Node03(7);
        tree.root.right.right.right = new Node03(8);

        System.out.println("Right view of the binary tree:");
        tree.rightView();
    }
}
```

Question-4:

Given a Binary Tree, the task is to print the bottom view from left to right. A node x is there in output if x is the bottommost node at its horizontal distance. The horizontal distance of the left child of a node x is equal to a horizontal distance of x minus 1, and that of a right child is the horizontal distance of x plus 1.

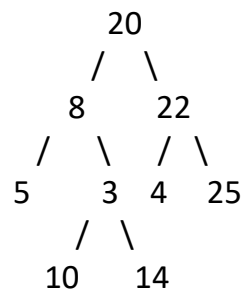
Examples:

Input:



Output: 5, 10, 3, 14, 25.

Input:



Output:5 10 4 14 25.

Explanation:

If there are multiple bottom-most nodes for a horizontal distance from the root, then print the later one in the level traversal.

3 and 4 are both the bottom-most nodes at a horizontal distance of 0, we need to print 4.

Solution Code:

```
package in.ineuron.pptAssignment23;

import java.util.*;

//Binary Tree node
class Node04 {
	int data;
	int hd; // horizontal distance
	Node04 left, right;

	public Node04(int item) {
		data = item;
		hd = Integer.MAX_VALUE;
```

```
            left = right = null;
        }
    }

    public class BinaryTree_4 {
        Node04 root;

        // Recursive function to calculate the horizontal distance of nodes
        // and print the bottom view of the binary tree
        void bottomViewUtil(Node04 node, int hd, int level, TreeMap<Integer, Pair> map) {
            if (node == null)
                return;

            // If the current node is the bottommost node at its horizontal distance,
            // update the map with the current node's data and level
            if (!map.containsKey(hd) || level >= map.get(hd).level) {
                map.put(hd, new Pair(node.data, level));
            }

            // Recur for the left and right subtrees with updated horizontal distance and
            // level
            bottomViewUtil(node.left, hd - 1, level + 1, map);
            bottomViewUtil(node.right, hd + 1, level + 1, map);
        }

        // Function to print the bottom view of the binary tree
        void bottomView() {
            if (root == null)
                return;

            // Map to store the horizontal distance and corresponding node data
            TreeMap<Integer, Pair> map = new TreeMap<>();

            // Calculate the horizontal distance and level of each node
            // and update the map with the bottommost node at each horizontal distance
            bottomViewUtil(root, 0, 0, map);

            // Print the bottom view nodes from left to right
            for (Map.Entry<Integer, Pair> entry : map.entrySet()) {
                System.out.print(entry.getValue().data + " ");
            }
        }

        // Pair class to store the node data and level
        class Pair {
            int data;
            int level;
```

```
        public Pair(int data, int level) {
            this.data = data;
            this.level = level;
        }
    }

    // Driver code
    public static void main(String args[]) {
        /*
         * Constructed binary tree is:       20
                                            /   \
                                           8     22
                                         /   \   /  \
                                        5     3 4   25
                                             / \
                                            10  14
         */
        BinaryTree_4 tree = new BinaryTree_4();
        tree.root = new Node04(20);
        tree.root.left = new Node04(8);
        tree.root.right = new Node04(22);
        tree.root.left.left = new Node04(5);
        tree.root.left.right = new Node04(3);
        tree.root.right.right = new Node04(25);
        tree.root.left.right.left = new Node04(10);
        tree.root.left.right.right = new Node04(14);

        System.out.println("Bottom view of the binary tree:");
        tree.bottomView();
    }
}
```


GitHub Link : https://github.com/devavratwadekar/ineuron_ppt_ProgramAssignmentCode/tree/master/PPTAssignment-23