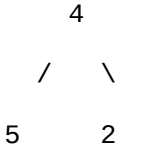
Assignment Questions 23

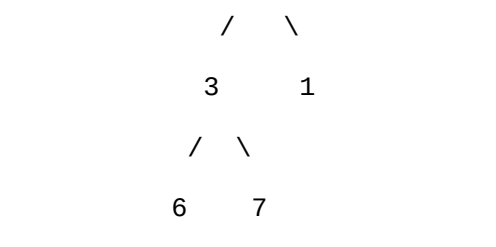
Question-1:
Given preorder of a binary tree, calculate its The preorder is given as a string with two possible characters.
1. ‘l’ denotes the leaf
2. ‘n’ denotes internal node
The given tree can be seen as a full binary tree where every node has 0 or two children. The two children of a node can ‘n’ or ‘l’ or mix of both.
Examples :
Input : nlnll
Output : 2

code:-

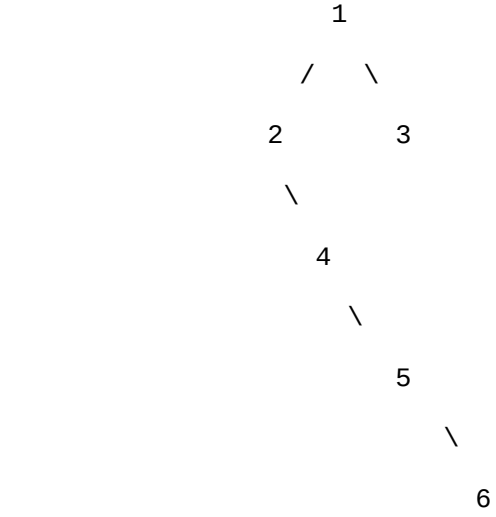
```
import java .io.*;

class GFG
{
      static int findDepthRec(String tree,int n, int index)
      {
            if (index >= n ||
                  tree.charAt(index) == 'l')
                  return 0;
            index++;
            int left = findDepthRec(tree,n, index);
            index++;
            int right = findDepthRec(tree, n, index);
            return Math.max(left, right) + 1;
      }
      static int findDepth(String tree,int n)
      {
            int index = 0;
            return (findDepthRec(tree,n, index));
      }
      static public void main(String[] args)
      {
            String tree = "nlnnlll";
            int n = tree.length();
            System.out.println(findDepth(tree, n));
      }
}
```

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
Input:



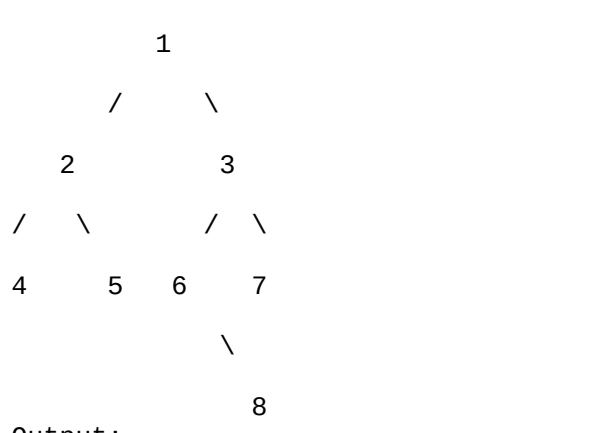
Output: 1 2 4 5 6

code:-

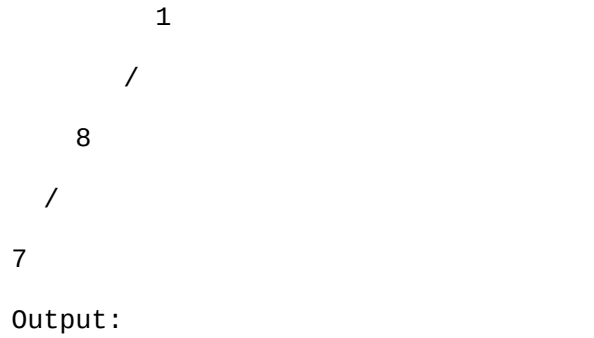
```
class Node {
      int data;
      Node left, right;

      public Node(int item)
      {
            data = item;
            left = right = null;
      }
}
class BinaryTree {
      Node root;
      static int max_level = 0;
      void leftViewUtil(Node node, int level)
      {
            if (node == null)
                  return;
            if (max_level < level) {
                  System.out.print(node.data + " ");
                  max_level = level;
            }
            leftViewUtil(node.left, level + 1);
            leftViewUtil(node.right, level + 1);
      }
```

```
      void leftView()
      {
            max_level = 0;
            leftViewUtil(root, 1);
      }
      public static void main(String args[])
      {
            BinaryTree tree = new BinaryTree();
            tree.root = new Node(10);
            tree.root.left = new Node(2);
            tree.root.right = new Node(3);
            tree.root.left.left = new Node(7);
            tree.root.left.right = new Node(8);
            tree.root.right.right = new Node(15);
            tree.root.right.left = new Node(12);
            tree.root.right.right.left = new Node(14);
            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:

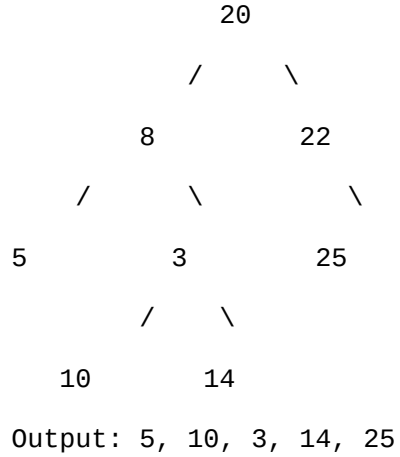


Output:
Right view of the tree is 1 8 7

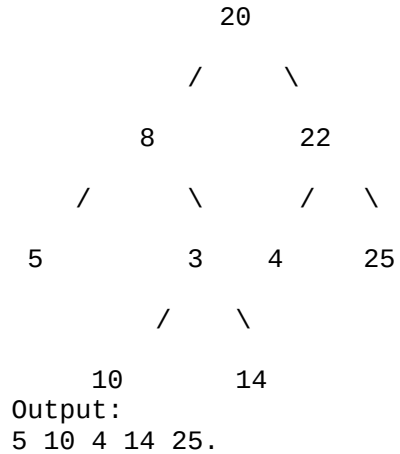
code:-

```
class Node {

	int data;
	Node left, right;

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

	int max_level;
}

class BinaryTree {

	Node root;
	Max_level max = new Max_level();
	void rightViewUtil(Node node, int level,Max_level max_level)
	{

		if (node == null)
			return;
		if (max_level.max_level < level) {
			System.out.print(node.data + " ");
			max_level.max_level = level;
		}
		rightViewUtil(node.right, level + 1, max_level);
		rightViewUtil(node.left, level + 1, max_level);
	}

	void rightView() { rightView(root); }
	void rightView(Node node)
	{

		rightViewUtil(node, 1, max);
	}
	public static void main(String args[])
	{
		BinaryTree tree = new BinaryTree();
		tree.root = new Node(1);
		tree.root.left = new Node(2);
		tree.root.right = new Node(3);
		tree.root.left.left = new Node(4);
		tree.root.left.right = new Node(5);
		tree.root.right.left = new Node(6);
		tree.root.right.right = new Node(7);
		tree.root.right.left.right = new Node(8);

		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.
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.

code:-

```
import java.util.*;
import java.util.Map.Entry;
class Node
```

```
{
	int data; //data of the node
	int hd; //horizontal distance of the node
	Node left, right; //left and right references
	public Node(int key)
	{
		data = key;
		hd = Integer.MAX_VALUE;
		left = right = null;
	}
}
class Tree
{
	Node root; //root node of tree
	public Tree() {}
	public Tree(Node node)
	{
		root = node;
	}
	public void bottomView()
	{
		if (root == null)
			return;
		int hd = 0;
		Map<Integer, Integer> map = new TreeMap<>();
		Queue<Node> queue = new LinkedList<Node>();
		root.hd = hd;
		queue.add(root);
		while (!queue.isEmpty())
		{
			Node temp = queue.remove();
			hd = temp.hd;
			map.put(hd, temp.data);
			if (temp.left != null)
			{
				temp.left.hd = hd-1;
				queue.add(temp.left);
			}
			if (temp.right != null)
			{
				temp.right.hd = hd+1;
				queue.add(temp.right);
			}
		}
		Set<Entry<Integer, Integer>> set = map.entrySet();
		Iterator<Entry<Integer, Integer>> iterator = set.iterator();
		while (iterator.hasNext())
		{
			Map.Entry<Integer, Integer> me = iterator.next();
			System.out.print(me.getValue()+" ");
		}
	}
}

public class BottomView
{
	public static void main(String[] args)
	{
		Node root = new Node(20);
```

```
            root.left = new Node(8);
            root.right = new Node(22);
            root.left.left = new Node(5);
            root.left.right = new Node(3);
            root.right.left = new Node(4);
            root.right.right = new Node(25);
            root.left.right.left = new Node(10);
            root.left.right.right = new Node(14);
            Tree tree = new Tree(root);
            System.out.println("Bottom view of the given binary tree:");
            tree.bottomView();
        }
}
```