program for Find Largest Value in Each Tree Row

```java
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;

class TreeNodeR {
    int val;
    TreeNodeR left;
    TreeNodeR right;

    TreeNodeR(int val) {
        this.val = val;
        this.left = null;
        this.right = null;
    }
}

public class LargestValueInEachTreeRow {
    public static List<Integer> largestValues(TreeNodeR root) {
        List<Integer> result = new ArrayList<>();
        if (root == null) {
            return result;
        }

        Queue<TreeNodeR> queue = new LinkedList<>();
        queue.offer(root);

        while (!queue.isEmpty()) {
            int size = queue.size();
            int max = Integer.MIN_VALUE;

            for (int i = 0; i < size; i++) {
                TreeNodeR node = queue.poll();
                max = Math.max(max, node.val);

                if (node.left != null) {
                    queue.offer(node.left);
                }

                if (node.right != null) {
                    queue.offer(node.right);
                }
            }

            result.add(max);
        }

        return result;
    }

    public static void main(String[] args) {
        // Construct a sample binary tree
```

```
        TreeNodeR root = new TreeNodeR(1);
        root.left = new TreeNodeR(3);
        root.right = new TreeNodeR(2);
        root.left.left = new TreeNodeR(5);
        root.left.right = new TreeNodeR(3);
        root.right.right = new TreeNodeR(9);

        List<Integer> largestValues = largestValues(root);

        System.out.println("Largest value : " + largestValues);
    }
}
```

OUTPUT

Largest value : [1, 3, 9]

Program for Sort Characters By Frequency

```java
import java.util.*;

public class SortCharactersByFrequency {
    public static String frequencySort(String s) {
        // Create a frequency map to store character
        Map<Character, Integer> fMap = new HashMap<>();
        for (char c : s.toCharArray()) {
            fMap.put(c, fMap.getOrDefault(c, 0) + 1);
        }

        // Create a min-heap based on character frequencies
        PriorityQueue<Map.Entry<Character, Integer>> minHeap = new PriorityQueue<>(
                (a, b) -> b.getValue() - a.getValue()
        );

        minHeap.addAll(fMap.entrySet());

        // Build the sorted string
        StringBuilder sortedString = new StringBuilder();
        while (!minHeap.isEmpty()) {
            Map.Entry<Character, Integer> entry = minHeap.poll();
            char c = entry.getKey();
            int frequency = entry.getValue();
            for (int i = 0; i < frequency; i++) {
                sortedString.append(c);
            }
        }

        return sortedString.toString();
    }

    public static void main(String[] args) {
        String input = "tree";
        String sortedString = frequencySort(input);

        System.out.println("Sorted string : " + sortedString);
    }
}
```

OUTPUT

Sorted string : eert