```
Q.1 : class TreeNode {
    int val;
    TreeNode left, right;
    TreeNode(int x) { val = x; }
}

public class BinaryTreeToString {
    public String treeToStr(TreeNode t) {
        StringBuilder sb = new StringBuilder();
        preorder(t, sb);
        return sb.toString();
    }

    private void preorder(TreeNode node, StringBuilder sb) {
        if (node == null) {
            return;
        }

        sb.append(node.val);

        if (node.left != null || node.right != null) {
            sb.append("(");
            preorder(node.left, sb);
            sb.append(")");

            if (node.right != null) {
                sb.append("(");
                preorder(node.right, sb);
                sb.append(")");
            }
        }
    }

    public static void main(String[] args) {
        BinaryTreeToString solution = new BinaryTreeToString();

        // Example Usage:
        // Constructing a sample binary tree: 1(2(4)(5))(3)
        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(2);
        root.right = new TreeNode(3);
        root.left.left = new TreeNode(4);
        root.left.right = new TreeNode(5);

        String result = solution.treeToStr(root);
        System.out.println(result);
    }
}
```

Q.2 :

Sol:

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class FindDuplicateFiles {
    public List<List<String>> findDuplicate(String[] paths) {
```

```
    Map<String, List<String>> contentToFiles = new HashMap<>();

    for (String path : paths) {
        String[] parts = path.split(" ");
        String directory = parts[0];

        for (int i = 1; i < parts.length; i++) {
            String file = parts[i];
            int indexOfContentStart = file.indexOf('(');
            String fileName = file.substring(0, indexOfContentStart);
            String content = file.substring(indexOfContentStart + 1, file.length() - 1);

            String fullPath = directory + "/" + fileName;

            contentToFiles.computeIfAbsent(content, k -> new ArrayList<>()).add(fullPath);
        }
    }

    List<List<String>> result = new ArrayList<>();
    for (List<String> files : contentToFiles.values()) {
        if (files.size() > 1) {
            result.add(files);
        }
    }

    return result;
}

public static void main(String[] args) {
        FindDuplicateFiles solution = new FindDuplicateFiles();

        // Example Usage:
        String[] paths = {
            "root/a 1.txt(abcd) 2.txt(efgh)",
            "root/c 3.txt(abcd)",
            "root/c/d 4.txt(efgh)",
            "root 4.txt(efgh)"
        };

        List<List<String>> result = solution.findDuplicate(paths);
        System.out.println(result);
    }
}
```