Leetcode

Leetcode 1 Solution:-

```
class Solution {
    public List<TreeNode> findDuplicateSubtrees(TreeNode root) {
        List<TreeNode> res = new LinkedList<>();
        traverse(root, new HashMap<>(), new HashMap<>(), res);
        return res;
    }

    public int traverse(TreeNode node, Map<String, Integer> tripletToID,
                        Map<Integer, Integer> cnt, List<TreeNode> res) {
        if (node == null) {
            return 0;
        }
        String triplet = traverse(node.left, tripletToID, cnt, res) + "," +
node.val +
                "," + traverse(node.right, tripletToID, cnt, res);
        if (!tripletToID.containsKey(triplet)) {
            tripletToID.put(triplet, tripletToID.size() + 1);
        }
        int id = tripletToID.get(triplet);
        cnt.put(id, cnt.getOrDefault(id, 0) + 1);
        if (cnt.get(id) == 2) {
            res.add(node);
        }
        return id;
    }
}
```

Leetcode 2 Solution:-

```
class Solution {

    public String[] findRestaurant(String[] list1, String[] list2) {
        Map<String, Integer> mp = new HashMap<>();
        for (int i = 0; i < list2.length; ++i) {
            mp.put(list2[i], i);
        }
        List<String> ans = new ArrayList<>();
        int mi = 2000;
        for (int i = 0; i < list1.length; ++i) {
            if (mp.containsKey(list1[i])) {
                int t = i + mp.get(list1[i]);
                if (t < mi) {
                    ans = new ArrayList<>();
                    ans.add(list1[i]);
                    mi = t;
                } else if (t == mi) {
```

```
                    ans.add(list1[i]);
                }
            }
        }
        return ans.toArray(new String[0]);
    }
}
```