题目描述：
现有两个整数数组，需要你找出两个数组中同时出现的整数，并按照如下要求输出：
1、有同时出现的整数时，先按照同时出现次数（整数在两个数组中都出现并且出现次数较少的那个）进行归类，然后按照出现次数从小到大依次按行输出。
2、没有同时出现的整数时，输出 NULL。
输入描述：
第一行为第一个整数数组，第二行为第二个整数数组，每行数据中整数与整数之间以英文逗号分隔，整数的取值范围为[-200,200]，数组长度的范围为[1,10000]之间的整数。
输出描述：
按照出现次数从小到大依次按行输出，每行输出的格式为:出现次数:该出现次数下的整数升序排序的结果。
格式中的":"为英文冒号，整数间以英文逗号分隔。
补充说明：

示例 1
输入：
5,3,6,-8,0,11
2,8,8,8,-1,15
输出：
NULL
说明：
两个整数数组没有同时出现的整数，输出 NULL。
示例 2
输入：
5,8,11,3,6,8,8,-1,11,2,11,11
11,2,11,8,6,8,8,-1,8,15,3,-9,11
输出：
1:-1,2,3,6
3:8,11
说明：
两个整数数组中同时出现的整数为-1、2、3、6、8、11，其中同时出现次数为 1 的整数为-1,2,3,6(升序排序)，同时出现次数为 3 的整数为 8,11(升序排序)，先升序输出出现次数为 1 的整数，再升序输出出现次数为 3 的整数。

```
import java.util.*;
import java.util.stream.Collectors;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String line1 = sc.nextLine();
        String line2 = sc.nextLine();
        List<Integer> collect1 = Arrays.stream(line1.split(",")).map(Integer::parseInt).collect(Collectors.toList());
```

```
        List<Integer> collect2 = Arrays.stream(line2.split(",")).map(Integer::parseInt).collect(Collectors.toList());
        //统计每个数组中数字出现次数
        HashMap<Integer,Integer> map1 = new HashMap<>();
        HashMap<Integer,Integer> map2 = new HashMap<>();
        for (Integer num : collect1){
            map1.put(num,map1.getOrDefault(num,0)+1);
        }
        for (Integer num : collect2){
            map2.put(num,map2.getOrDefault(num,0)+1);
        }
        HashMap<Integer,List<Integer>> resMap = new HashMap<>();
        for (int num : map1.keySet()){
            if (map2.containsKey(num)){
                //比较两个数字的出现次数,取较少的次数
                int times = map1.get(num)<map2.get(num)?map1.get(num):map2.get(num);
                if (resMap.containsKey(times)){
                    List<Integer> nums = resMap.get(times);
                    nums.add(num);
                    resMap.put(times,nums);
                    continue;
                }
                ArrayList<Integer> nums = new ArrayList<>();
                nums.add(num);
                resMap.put(times,nums);
            }
        }
        if (resMap.size() == 0) {
            System.out.println("NULL");
        }else{
            for(Integer times : resMap.keySet()){
                StringBuilder sb = new StringBuilder();
                sb.append(times).append(":");
                List<Integer> sameNums = resMap.get(times);
                Collections.sort(sameNums);
                for (Integer num : sameNums){
                    sb.append(num).append(",");
                }
                sb.deleteCharAt(sb.length()-1);
                System.out.println(sb);
            }
        }
    }
}
```