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。

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.Scanner;
import java.util.Map;
import java.util.HashMap;
import java.util.TreeMap;
import java.util.List;
import java.util.ArrayList;
import java.util.Collections;

public class Main {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String str1 = in.nextLine();
        String[] numbers1 = str1.split(",");
        int len1 = numbers1.length;
        Map<String, Integer> map1 = new HashMap<>(len1);
        for (int i = 0; i < len1; i++) {
            if (map1.get(numbers1[i]) == null) {
                map1.put(numbers1[i], 1);
            } else {
                int oldValue = map1.get(numbers1[i]);
                map1.put(numbers1[i], oldValue + 1);
            }
        }

        String str2 = in.nextLine();
        String[] numbers2 = str2.split(",");
        int len2 = numbers2.length;
        Map<String, Integer> map2 = new HashMap<>(len2);
        for (int i = 0; i < len2; i++) {
            if (map2.get(numbers2[i]) == null) {
```

```
                map2.put(numbers2[i], 1);
            } else {
                int oldValue = map2.get(numbers2[i]);
                map2.put(numbers2[i], oldValue + 1);
            }
        }

        Map<String, Integer> map3 = new HashMap<>();
        for (Map.Entry<String, Integer> entry : map1.entrySet()) {
            String key = entry.getKey();
            if (map2.get(key) != null) {
                int value = entry.getValue() < map2.get(key) ? entry.getValue() :
map2.get(key);
                map3.put(key, value);
            }
        }

        if (map3.size() == 0) {
            System.out.println("NULL");
        } else {
            TreeMap<Integer, List<Integer>> map4 = new TreeMap<>();
            for (Map.Entry<String, Integer> entry : map3.entrySet()) {
                Integer key = entry.getValue();
                if (map4.get(key) == null) {
                    List<Integer> temp = new ArrayList<>();
                    temp.add(Integer.parseInt(entry.getKey()));
                    map4.put(key, temp);
                } else {
                    map4.get(key).add(Integer.parseInt(entry.getKey()));
                }
            }


            for (Map.Entry<Integer, List<Integer>> entry : map4.entrySet()) {
                System.out.print(entry.getKey() + ":");
                List<Integer> temp = entry.getValue();
                Collections.sort(temp);
                for (int i = 0; i < temp.size(); i++) {
                    System.out.print(temp.get(i));
                    if (i != temp.size() - 1) {
                        System.out.print(",");
                    }
                }
                System.out.println();
```

```
            }
        }
    }
}
```