题目描述：
现有两门选修课，每门选修课都有一部分学生选修，每个学生都有选修课的成绩，需要你找出同时选修了两门选修课的学生，先按照班级进行划分，班级编号小的先输出，每个班级按照两门选修课成绩和的降序排序，成绩相同时按照学生的学号升序排序。
输入描述：
第一行为第一门选修课学生的成绩，第二行为第二门选修课学生的成绩，每行数据中学生之间以英文分号分隔，每个学生的学号和成绩以英文逗号分隔，学生学号的格式为 8 位数字(2 位院系编号+入学年份后 2 位+院系内部 1 位专业编号+所在班级 3 位学号)，学生成绩的取值范围为[0,100]之间的整数，两门选修课选修学生数的取值范围为[1-2000]之间的整数。
输出描述：
同时选修了两门选修课的学生的学号，如果没有同时选修两门选修课的学生输出 NULL，否则，先按照班级划分，班级编号小的先输出，每个班级先输出班级编号(学号前五位)，然后另起一行输出这个班级同时选修两门选修课的学生学号，学号按照要求排序(按照两门选修课成绩和的降序，成绩和相同时按照学号升序)，学生之间以英文分号分隔。
补充说明：

示例1
输入：01202021,75;01201033,95;01202008,80;01203006,90;01203088,100
01202008,70;01203088,85;01202111,80;01202021,75;01201100,88
输出：01202
01202008;01202021
01203
01203088
说明：同时选修了两门选修课的学生01202021、01202008、01203088，这三个学生两门选修课的成绩和分别为150、150、185，01202021、01202008属于01202班的学生，按照成绩和降序，成绩相同时按学号升序输出的结果为01202008;01202021,01203088属于01203班的学生，按照成绩和降序，成绩相同时按学号升序输出的结果为01203088，01202的班级编号小于01203的班级编号，需要先输出。

示例2
输入：01201022,75;01202033,95;01202018,80;01203006,90;01202066,100
01202008,70;01203102,85;01202111,80;01201021,75;01201100,88
输出：NULL
说明：没有同时选修了两门选修课的学生，输出NULL。

```java
// 本题为考试单行多行输入输出规范示例，无需提交，不计分。
import java.util.*;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String s1 = in.nextLine();
        String s2 = in.nextLine();
        String[] split = s1.split(";");
        Map<String,Integer> map1 = jijue(s1);
        Map<String,Integer> map2 = jijue(s2);
        TreeMap<String, List<String>> map = new TreeMap<>();
        for (String s : map1.keySet()) {
            if (map2.containsKey(s)){
                String substring = s.substring(0,5);
                List<String> list = map.getOrDefault(substring, new ArrayList<>());
                list.add(s);
                map.put(substring,list);
            }
        }
        if (map.isEmpty()){
            System.out.println("NULL");
            return;
        }
        for (Map.Entry<String, List<String>> entry : map.entrySet()) {
            System.out.println(entry.getKey());
            List<String> list = entry.getValue();
            Collections.sort(list,(a,b)->{
                Integer a1 = map1.get(a);
                Integer a2 = map2.get(a);
                Integer b1 = map1.get(b);
                Integer b2 = map2.get(b);
                int ashu = a1+a2;
                int bshu = b1+b2;
                return bshu==ashu?Integer.valueOf(a.substring(5))-Integer.valueOf(b.substring(5)):bshu-ashu;
            });
            StringBuilder builder = new StringBuilder();
            for (String s : list) {
                builder.append(s);
                builder.append(";");
            }
            builder.deleteCharAt(builder.length()-1);
            System.out.println(builder);
        }
    }

    private static Map<String, Integer> jijue(String s1) {
        Map<String, Integer> res = new HashMap<>();
        String[] split = s1.split(";");
        for (String s : split) {
            String[] strings = s.split(",");
            res.put(strings[0],Integer.valueOf(strings[1]));
        }
        return res;
    }
}
```