选修课

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。

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。

```
import java.util.*;

public class Main {
    // 本题为考试单行多行输入输出规范示例，无需提交，不计分。

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        String[] gradeA = in.nextLine().split(";");
        HashMap<String, Student> map = new HashMap<>();
        getGradeA(map, gradeA);
        String[] gradeB = in.nextLine().split(";");
        getGradeB(map, gradeB);
        // 按班级划分学生
        Map<String, List<Student>> classMap = new TreeMap<>();
        for(String stuNo : map.keySet()){
            Student stu = map.get(stuNo);
            List<Student> list = classMap.getOrDefault(stu.classNO, new
ArrayList<>());
            list.add(stu);
            classMap.put(stu.classNO, list);
        }
        // 输出
        putRes(classMap);
    }
    public static void getGradeA(HashMap<String, Student> map, String[]
gradeA){
        // 处理 A 成绩
        for (int i = 0; i < gradeA.length; i++) {
            String[] arr = gradeA[i].split(",");
            String classNo = arr[0].substring(0,5);
            String stuNo = arr[0];
            int grade = Integer.parseInt(arr[1]);
            if(map.containsKey(stuNo)){
                map.get(stuNo).gradeA = grade;
            }else {
```

```
                Student s1 = new Student();
                s1.classNO = classNo;
                s1.stuNo = stuNo;
                s1.gradeA = grade;
                map.put(stuNo, s1);
            }
        }
    }

    public static void getGradeB(HashMap<String, Student> map, String[] gradeB){
        for (int i = 0; i < gradeB.length; i++) {
            String[] arr = gradeB[i].split(",");
            String classNo = arr[0].substring(0,5);
            String stuNo = arr[0];
            int grade = Integer.parseInt(arr[1]);
            if(map.containsKey(stuNo)){
                Student s2 = map.get(stuNo);
                s2.gradeB = grade;
                s2.allGrade = s2.gradeA + s2.gradeB;
            }else {
                Student s1 = new Student();
                s1.classNO = classNo;
                s1.stuNo = stuNo;
                s1.gradeB = grade;
                s1.allGrade = s1.gradeA + s1.gradeB;
                map.put(stuNo, s1);
            }
        }
    }

    public static void putRes(Map<String, List<Student>> classMap){
        boolean firstClass = true;
        for(String classNo : classMap.keySet()){
            List<Student> students = classMap.get(classNo);
            students.sort((a,b) -> {
                if(a.allGrade != b.allGrade){
                    return b.allGrade - a.allGrade;
                }else {
                    return a.stuNo.compareTo(b.stuNo);
                }
            });

            List<String> select = new ArrayList<>();
```

```
                for(Student stu : students){
                    if(stu.gradeA > 0 && stu.gradeB > 0){
                        select.add(stu.stuNo);
                    }
                }

                if(select.size() > 0){
                    if(firstClass){
                        firstClass = false;
                    }else{
                        System.out.println();
                    }
                    System.out.print(classNo + "\n" + String.join(";", select));
                }
            }
            if(firstClass){
                System.out.println("NULL");
            }
        }

        static class Student{
            public String classNO;
            public String stuNo;
            public int gradeA;
            public int gradeB;
            public int allGrade;
        }
}
```