

题目描述：

一个有 N 个选手参加比赛，选手编号为 $1 \sim N$ ($3 \leq N \leq 100$)，有 M ($3 \leq M \leq 10$) 个评委对选手进行打分。打分规则为每个评委对选手打分，最高分 10 分，最低分 1 分。

请计算得分最多的 3 位选手的编号。如果得分相同，则得分高分值最多的选手排名靠前(10 分数量相同，则比较 9 分的数量，以此类推，用例中不会出现多个选手得分完全相同的情况)。

输入描述：

第一行为半角逗号分割的两个正整数，第一个数字表示 M ($3 \leq M \leq 10$) 个评委，第二个数字表示 N ($3 \leq N \leq 100$) 个选手。

第 2 到 $M+1$ 行是半角逗号分割的整数序列，表示评委为每个选手的打分，0 号下标数字表示 1 号选手分数，1 号下标数字表示 2 号选手分数，依次类推。

输出描述：

选手前 3 名的编号。

注：若输入为异常，输出 -1，如 M 、 N 、打分不在范围内。

补充说明：

示例 1

输入：

4,5

10,6,9,7,6

9,10,6,7,5

8,10,6,5,10

9,10,8,4,9

输出：

2,1,5

说明：

第一行代表有 4 个评委，5 个选手参加比赛

矩阵代表是 4×5 ，每个数字是选手的编号，每一行代表一个评委对选手的打分排序，

2 号选手得分 36 分排第 1，1 号选手 36 分排第 2，5 号选手 30 分(2 号 10 分值有 3 个，1 号 10 分值只有 1 个，所以 2 号排第一)

示例 2

输入：

2,5

7,3,5,4,2

8,5,4,4,3

输出：

-1

说明：

只有 2 个评委，要求最少为 3 个评委

示例 3

输入：

4,2

8,5

5,6

10,4

8,9

输出:

-1

说明:

只有 2 名选手参加, 要求最少为 3 名

示例 4

输入:

4,5

11,6,9,7,8

9,10,6,7,8

8,10,6,9,7

9,10,8,6,7

输出:

-1

说明:

第一个评委给第一个选手打分 11, 无效分数

import java.util.*;

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] nums =
            Arrays.stream(sc.nextLine().split(",")).mapToInt(Integer::parseInt).toArray();
        int pw = nums[0];
        int xs = nums[1];
        int[][] scores = new int[pw][xs];
        for (int i = 0; i < pw; i++) {
            scores[i] =

Arrays.stream(sc.nextLine().split(",")).mapToInt(Integer::parseInt).toArray();

        }
        System.out.println(getAns(pw, xs, scores));
    }

    private static String getAns(int pw, int xs, int[][] scores) {
        if (pw < 3) return "-1";
        if (xs < 3) return "-1";
        ArrayList<ArrayList<Integer>> xsScore = new ArrayList<>();
        for (int i = 0; i < xs; i++) {
            int sum = 0;
            ArrayList<Integer> list = new ArrayList<>();
            for (int j = 0; j < pw; j++) {
```

```

        int tmp = scores[j][i];
        if (tmp < 1 || tmp > 10) {
            return "-1";
        }
        list.add(tmp);
        sum += tmp;
    }
    list.sort((a, b) -> b - a);
    list.add(0, sum);
    list.add(0, i + 1);
    xsScore.add(list);
}
Collections.sort(xsScore, (a, b) -> {
    if (a.get(1) == b.get(1)) {
        for (int i = 2; i < a.size(); i++) {
            if (a.get(i) < b.get(i)) {
                return 1;
            } else if (a.get(i) == b.get(i)) {
                continue;
            } else {
                return -1;
            }
        }
    }
    return b.get(1) - a.get(1);
});
StringJoiner joiner = new StringJoiner(",");
for (int i = 0; i < 3; i++) {
    joiner.add(String.valueOf(xsScore.get(i).get(0)));
}
return joiner.toString();
}
}

```