题目描述

- 一个有N个选手参加比赛，选手编号为1~N（3<=N<=100），有M（3<=M<=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隔的两个正整数，第一个数字表示M（3<=M<=10）个评委，第二个数字表示N（3<=N<=100）个选手。
- 第2到M+1行表示每位评委给选手的打分，0给下划线字表示1号选手分数，1给下划线字表示2号选手分数，依次类推。

输出描述

- 选手的3名的编号。
- 注：若输入为异常，输出-1，如M、N、打分不在规定内。

用例

输入	4,5 10,6,9,7,6 9,10,8,7,5 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 如样代表是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个10分值有3个，1个10分值只有1个，所以2号排第一）

输入	2,5 7,3,5,4,2 8,5,4,4,5
输出	-1
说明	只有2个评委，要求最少为3个评委

输入	4,2 8,5 5,6 10,4 8,9
输出	-1
说明	只有2位选手参加，要求最少为3名

输入	4,5 11,6,9,7,8 9,10,6,7,5 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

题目解析

考察 数组排序 。主要难点在于下面密码的设计

> 如果得分相同，则得分高分值最多的选手排名靠前

则这里直接将选手分数降序排序后，join("")为字符串数值，高分多的选手拼字符串数值越大

JavaScript算法源码

```
// JavaScript Node.js环境下，需要处理输入流数据
const readline = require("readline");

const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});

const lines = [];
let m, n;
rl.on("line", (line) => {
  lines.push(line);

  if (lines.length === 1) {
    [m, n] = lines[0].split(",").map(Number);
  }

  if (m && lines.length === m + 1) {
    lines.shift();
    const scores = lines.map((line) => line.split(",").map(Number));
    console.log(getResult(scores, m, n));
    lines.length = 0;
  }
});

function getResult(scores, m, n) {
  if (m < 3 || m > 10 || n < 3 || n > 100) return -1;

  const players = [];

  for (let j = 0; j < n; j++) {
    const player = [];
    for (let i = 0; i < m; i++) {
      if (scores[i][j] > 10 || scores[i][j] < 1) return "-1";
      player.push(scores[i][j]);
    }
    player.sort((a, b) => b - a);
    players.push({
      idx: j,
      sum: player.reduce((p, c) => p + c),
      score: player,
    });
  }

  return players
    .sort((a, b) => {
      if (a.sum != b.sum) return b.sum - a.sum;

      for (let i = 0; i < m; i++) {
        if (a.score[i] == b.score[i]) continue;
        return b.score[i] - a.score[i];
      }

      return 0;
    })
    .slice(0, 3)
    .map((x) => x.idx + 1)
    .join(",");
}
```

Java算法源码

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.Scanner;
import java.util.StringJoiner;

public class Main {
  // 加入注释
  public static void main(String[] args) {
    // 输入流处理模板
    Scanner sc = new Scanner(System.in).useDelimiter("[,\n]");

    int m = sc.nextInt();
    int n = sc.nextInt();

    int[][] scores = new int[m][n];
    for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) {
        scores[i][j] = sc.nextInt();
      }
    }

    System.out.println(getResult(m, n, scores));
  }

  // 算法入口
  public static String getResult(int m, int n, int[][] scores) {
    if (m < 3 || m > 10 || n < 3 || n > 100) return "-1";

    HashMap<Integer, Integer[]> players = new HashMap<>();

    for (int j = 0; j < n; j++) { // 列
      Integer[] player = new Integer[m];
      for (int i = 0; i < m; i++) {
        if (scores[i][j] > 10 || scores[i][j] < 1) return "-1"; // 分数要求1-10检查，否则不合法
        player[i] = scores[i][j];
      }
      Arrays.sort(player, (a, b) -> b - a); // 单每个选手的降序排序
      players.put(j, player);
    }

    StringJoiner sj = new StringJoiner(",");

    players.entrySet().stream()
        .sorted(
            (e, b) -> {
              Integer[] playerA = a.getValue();
              Integer[] playerB = b.getValue();

              int sumA = sum(playerA);
              int sumB = sum(playerB);

              if (sumA != sumB) { // 总分比较排序
                return sumB - sumA;
              }

              for (int i = 0; i < m; i++) {
                if (playerA[i] == playerB[i]) continue;
                return playerB[i] - playerA[i]; // 分数高分优先
              }

              return 0;
            }
        ).limit(3)
        .forEach(p -> sj.add(p.getKey() + 1 + ""));

    return sj.toString();
  }

  public static int sum(Integer[] arr) {
    return Arrays.stream(arr).reduce(Integer::sum).orElse(0);
  }
}
```

Python算法源码

```
# 输入处理
import functools

m, n = map(int, input().split(","))
scores = [list(map(int, input().split(","))) for _ in range(m)]


def cmp(a, b):
    if a["sum"] != b["sum"]:
        return b["sum"] - a["sum"]

    for i in range(m):
        if a["score"][i] == b["score"][i]:
            continue
        return b["score"][i] - a["score"][i]

    return 0


# 算法入口
def getResult():
    if m < 3 or m > 10 or n < 3 or n > 100:
        return "-1"

    players = []

    for j in range(n):
        player = []
        for i in range(m):
            if scores[i][j] > 10 or scores[i][j] < 1:
                return "-1"
            player.append(scores[i][j])

        player.sort(reverse=True)
        players.append({
            "idx": j,
            "sum": sum(player),
            "score": player
        })

    players.sort(key=functools.cmp_to_key(cmp))

    return ",".join(list(map(lambda x: str(x["idx"] + 1), players[:3])))


# 算法结束
print(getResult())
```