【华为OD机考 统一考试机试C卷】学生排名/智能成绩表（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，C卷真题已基本整理完毕
抽到原题的概率为2/3到3/3，也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。
另外订阅专栏还可以联系笔者开通在线 OJ 进行刷题，提高刷题效率。
真题目录：华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明
专栏：2023华为OD机试(B卷+C卷+D卷)（C++JavaJSPy）
华为OD面试真题精选：华为OD面试真题精选
在线OJ：点击立即刷题，模拟真实机考环境

题目描述

小明来到学校当老师，需要将学生按考试总分或单科分数进行排名，你能帮帮他吗?

输入描述

第 1 行输入两个整数，学生人数 n 和科目数量 m。

- 0 < n < 100
- 0 < m < 10

第 2 行输入 m 个科目名称，彼此之间用空格隔开。

- 科目名称只由英文字母构成，单个长度不超过10个字符。
- 科目的出现顺序和后续输入的学生成绩一一对应。
- 不会出现重复的科目名称。

第 3 行开始的 n 行，每行包含一个学生的姓名和该生 m 个科目的成绩（空格隔开）

- 学生不会重名。
- 学生姓名只由英文字母构成，长度不超过10个字符。
- 成绩是0~100的整数，依次对应第2行种输入的科目。

第n+2行，输入用作排名的科目名称。若科目不存在，则按总分进行排序。

输出描述

输出一行，按成绩排序后的学生名字，空格隔开。成绩相同的按照学生姓名字典顺序排序。

样例1

输入

```
3 2
yuwen shuxue
fangfang 95 90
xiaohua 88 98
minmin 100 82
shuxue
```

输出

```
xiaohua fangfang minmin
```

说明:

> 按照shuxue成绩排名依次是 xiaohua fangfang minmin

样例2

输入

```
3 2
yuwen shuxue
fangfang 95 90
xiaohua 88 95
minmin 90 95
zongfen
```

输出

```
fangfang minmin xiaohua
```

说明:

> 排序科目不存在，按总分排序 , fangfang 和 minmin 总分相同，按姓名的字典序顺序， fangfang 排在前面

解题思路

C++

```cpp
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>

using namespace std;

// 定义学生类
class Student {
public:
    string name; // 学生姓名
    int totalScore; // 学生总分
    map<string, int> scores; // 存储学生各科成绩的映射

    // 构造函数，初始化学生姓名和总分
    Student(const string& name) : name(name), totalScore(0) {}

    // 添加成绩的方法，同时累加到总分
```

```
    void addScore(const string& subject, int score) {
        scores[subject] = score; // 设置指定科目的成绩
        totalScore += score; // 累加到总分
    }

    // 获取指定科目的成绩
    int getScore(const string& subject) const {
        auto it = scores.find(subject); // 查找科目对应的成绩
        return it != scores.end() ? it->second : 0; // 如果找到，则返回成绩，否则返回0
    }
};

int main() {
    int n, m; // n为学生数量，m为科目数量
    cin >> n >> m;
    vector<string> subjects(m); // 存储科目名称的向量
    for (int i = 0; i < m; ++i) {
        cin >> subjects[i]; // 输入科目名称
    }
    vector<Student> students; // 存储学生对象的向量

    for (int i = 0; i < n; ++i) {
        string name; // 学生姓名
        cin >> name; // 输入学生姓名
        Student student(name); // 创建学生对象
        for (int j = 0; j < m; ++j) {
            int score; // 成绩
            cin >> score; // 输入成绩
            student.addScore(subjects[j], score); // 添加成绩到学生对象
        }
        students.push_back(student); // 将学生对象添加到向量中
    }

    string rankSubject; // 用作排名的科目名称
    cin >> rankSubject; // 输入排名科目

    // 对学生进行排序
    sort(students.begin(), students.end(), [&](const Student& a, const Student& b) {
        int scoreA = rankSubject.empty() ? a.totalScore : a.getScore(rankSubject); // 获取a的排名科目成绩或总分
        int scoreB = rankSubject.empty() ? b.totalScore : b.getScore(rankSubject); // 获取b的排名科目成绩或总分

```

```
        if (scoreA != scoreB) {
            return scoreA > scoreB; // 成绩高的排前面
        }
        return a.name < b.name; // 成绩相同则按姓名字典序排列
    });

    // 输出排序后的学生姓名
    for (const auto& student : students) {
        cout << student.name << ' ';
    }
    cout << endl;

    return 0;
}
```

Java

```
import java.util.*;

public class Main {
    // 定义学生类
    static class Student {
        String name; // 学生姓名
        int totalScore; // 学生总分
        Map<String, Integer> scores; // 存储学生各科成绩的映射

        // 构造函数，初始化学生姓名、成绩映射和总分
        Student(String name) {
            this.name = name;
            this.scores = new HashMap<>();
            this.totalScore = 0;
        }

        // 添加成绩的方法，同时累加到总分
        void addScore(String subject, int score) {
            scores.put(subject, score);
            totalScore += score;
        }

        // 获取指定科目的成绩，若没有则返回0
```

```
        int getScore(String subject) {
            return scores.getOrDefault(subject, 0);
        }
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        // 读取学生人数和科目数量
        int n = scanner.nextInt();
        int m = scanner.nextInt();
        scanner.nextLine(); // 读取并忽略换行符

        // 读取科目名称
        String[] subjects = scanner.nextLine().split(" ");
        List<Student> students = new ArrayList<>();

        // 读取每个学生的姓名和成绩
        for (int i = 0; i < n; i++) {
            String[] tokens = scanner.nextLine().split(" ");
            Student student = new Student(tokens[0]);
            for (int j = 0; j < m; j++) {
                student.addScore(subjects[j], Integer.parseInt(tokens[j + 1]));
            }
            students.add(student);
        }

        // 读取用作排名的科目名称
        String rankSubject = scanner.nextLine();
        // 关闭扫描器
        scanner.close();

        // 对学生列表进行排序
        students.sort((s1, s2) -> {
            // 根据指定科目或总分进行比较
            int score1 = rankSubject.equals("") ? s1.totalScore : s1.getScore(rankSubject);
            int score2 = rankSubject.equals("") ? s2.totalScore : s2.getScore(rankSubject);
            if (score1 != score2) {
                return score2 - score1; // 降序排序
            } else {
                return s1.name.compareTo(s2.name); // 成绩相同则按姓名升序排序
```

```
            }
        });

        // 输出排序后的学生姓名
        students.forEach(student -> System.out.print(student.name + " "));
    }
}
```

javaScript

```javascript
// 引入 readline 模块用于读取命令行输入
const readline = require('readline');

// 创建 readline 接口实例
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// 定义学生类
class Student {
  constructor(name) {
    this.name = name; // 学生姓名
    this.totalScore = 0; // 学生总分
    this.scores = {}; // 存储学生各科成绩的映射
  }

  // 添加成绩的方法，同时累加到总分
  addScore(subject, score) {
    this.scores[subject] = score;
    this.totalScore += score;
  }

  // 获取指定科目的成绩，若没有则返回0
  getScore(subject) {
    return this.scores[subject] || 0;
  }
}

// 创建一个异步处理函数
```

```
async function processInput() {
  // 通过 readline 逐行读取输入
  const lines = [];
  for await (const line of rl) {
    lines.push(line);
  }

  // 解析输入数据
  const [n, m] = lines[0].split(' ').map(Number);
  const subjects = lines[1].split(' ');
  const students = [];

  // 读取每个学生的姓名和成绩
  for (let i = 0; i < n; i++) {
    const tokens = lines[i + 2].split(' ');
    const student = new Student(tokens[0]);
    for (let j = 0; j < m; j++) {
      student.addScore(subjects[j], parseInt(tokens[j + 1], 10));
    }
    students.push(student);
  }

  // 读取用作排名的科目名称
  const rankSubject = lines[n + 2];

  // 对学生列表进行排序
  students.sort((s1, s2) => {
    const score1 = rankSubject === '' ? s1.totalScore : s1.getScore(rankSubject);
    const score2 = rankSubject === '' ? s2.totalScore : s2.getScore(rankSubject);
    if (score1 !== score2) {
      return score2 - score1; // 降序排序
    } else {
      return s1.name.localeCompare(s2.name); // 成绩相同则按姓名升序排序
    }
  });

  // 输出排序后的学生姓名
  students.forEach(student => process.stdout.write(`${student.name} `));
  process.stdout.write('\n');
```

```
  // 关闭 readline 接口
  rl.close();
}

// 调用异步处理函数
processInput();
```

Python

```python
# 导入必要的库
from collections import defaultdict

# 定义学生类
class Student:
    def __init__(self, name):
        self.name = name  # 学生姓名
        self.total_score = 0  # 学生总分
        self.scores = defaultdict(int)  # 存储学生各科成绩的字典，默认值为0

    # 添加成绩的方法，同时累加到总分
    def add_score(self, subject, score):
        self.scores[subject] = score
        self.total_score += score

    # 获取指定科目的成绩
    def get_score(self, subject):
        return self.scores[subject]

# 主函数
def main():
    # 读取学生人数和科目数量
    n, m = map(int, input().split())

    # 读取科目名称
    subjects = input().split()
    students = []

    # 读取每个学生的姓名和成绩
    for _ in range(n):
        tokens = input().split()
```

```
        student = Student(tokens[0])
        for j in range(m):
            student.add_score(subjects[j], int(tokens[j + 1]))
        students.append(student)

    # 读取用作排名的科目名称
    rank_subject = input()

    # 对学生列表进行排序
    students.sort(key=lambda s: (-s.get_score(rank_subject) if rank_subject else -s.total_score, s.name))

    # 输出排序后的学生姓名
    for student in students:
        print(student.name, end=' ')

# 调用主函数
if __name__ == "__main__":
    main()
```

C语言

```

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX_STUDENTS 100
#define MAX_SUBJECTS 10
#define MAX_NAME_LEN 11

// 定义学生结构体
typedef struct {
    char name[MAX_NAME_LEN]; // 学生姓名
    int totalScore; // 学生总分
    int scores[MAX_SUBJECTS]; // 存储学生各科成绩的数组
} Student;

// 定义全局变量
int n, m; // n为学生数量，m为科目数量
char subjects[MAX_SUBJECTS][MAX_NAME_LEN]; // 存储科目名称的数组
```

```
Student students[MAX_STUDENTS]; // 存储学生对象的数组
char rankSubject[MAX_NAME_LEN]; // 用作排名的科目名称

// 定义比较函数，用于 qsort 函数
int cmp(const void *a, const void *b) {
    Student *studentA = (Student *)a;
    Student *studentB = (Student *)b;
    int scoreA = studentA->totalScore;
    int scoreB = studentB->totalScore;
    if (strcmp(rankSubject, "") != 0) {
        for (int i = 0; i < m; ++i) {
            if (strcmp(rankSubject, subjects[i]) == 0) {
                scoreA = studentA->scores[i];
                scoreB = studentB->scores[i];
                break;
            }
        }
    }
    if (scoreA != scoreB) {
        return scoreB - scoreA; // 成绩高的排前面
    }
    return strcmp(studentA->name, studentB->name); // 成绩相同则按姓名字典序排列
}

int main() {
    scanf("%d%d", &n, &m);
    for (int i = 0; i < m; ++i) {
        scanf("%s", subjects[i]);
    }

    for (int i = 0; i < n; ++i) {
        scanf("%s", students[i].name);
        students[i].totalScore = 0;
        for (int j = 0; j < m; ++j) {
            scanf("%d", &students[i].scores[j]);
            students[i].totalScore += students[i].scores[j];
        }
    }

    scanf("%s", rankSubject);
```

```
    // 对学生进行排序
    qsort(students, n, sizeof(Student), cmp);

    // 输出排序后的学生姓名
    for (int i = 0; i < n; ++i) {
        printf("%s ", students[i].name);
    }
    printf("\n");

    return 0;
}
```

完整用例

用例1

```
3 2
yuwen shuxue
fangfang 95 90
xiaohua 88 95
minmin 100 82
shuxue
```

用例2

```
3 2
yuwen shuxue
fangfang 95 90
xiaohua 88 95
minmin 90 95
zongfen
```

用例3

```
5 3
Math English Science
Alice 80 90 100
Bob 80 95 90
```

```
Charlie 80 85 95
David 80 100 85
Eve 80 80 80
Math
```

用例4

```
4 2
Literature History
Alice 90 10
Bob 85 15
Charlie 80 20
David 75 25
zongfen
```

用例5

```
3 2
Physics Chemistry
Alice 100 0
Bob 50 50
Charlie 0 100
Chemistry
```

用例6

```
3 3
Math Physics Chemistry
Alice 100 100 100
Bob 90 90 90
Charlie 80 80 80
Math
```

用例7

```
3 3
Math English Science
Alice 95 85 75
Bob 85 75 95
```

```
Charlie 75 95 85
Literature
```

用例8

```
1 1
PE
Single 100
PE
```

用例9

```
3 2
Biology Biochemistry
Alice 60 70
Bob 70 60
Charlie 80 80
Biology
```

用例10

```
2 2
Mathematics History
Alexanderson 95 85
Benjaminson 85 95
Mathematics
```

文章目录