查找元素

- [ ] 小范围的查找：直接遍历
- [ ] 大范围查找：利用查找算法（二叉排序树、平衡二叉树、红黑树、B树、B+树）
- [ ] 另一种思路：根据计数排序算法，**先排序，后二分查找**；

一、乙类题目：

1、成绩排名

读入 *n*（>0）名学生的姓名、学号、成绩，分别输出成绩最高和成绩最低学生的姓名和学号。

输入格式：每个测试输入包含 1 个测试用例，格式为

```
第 1 行：正整数 n
第 2 行：第 1 个学生的姓名 学号 成绩
第 3 行：第 2 个学生的姓名 学号 成绩
  ... ... ...
第 n+1 行：第 n 个学生的姓名 学号 成绩
```

其中`姓名`和`学号`均为不超过 10 个字符的字符串，成绩为 0 到 100 之间的一个整数，这里保证在一组测试用例中没有两个学生的成绩是相同的。

输出格式：

对每个测试用例输出 2 行，第 1 行是成绩最高学生的姓名和学号，第 2 行是成绩最低学生的姓名和学号，字符串间有 1 空格。

输入样例：

```
3
Joe Math990112 89
Mike CS991301 100
Mary EE990830 95
```

输出样例：

```
Mike CS991301
Joe Math990112
```

核心代码如下：

```
for(int i=0;i<n;i++){
    scanf("%s%s%d", &student[i].name, &student[i].id, &student[i].score);
    if(student[i].score > student[score_max].score) score_max = i;
    else if(student[i].score < student[score_min].score) score_min = i;
}
```

2、人口普查

某城镇进行人口普查，得到了全体居民的生日。现请你写个程序，找出镇上**最年长和最年轻的人**。这里确保每个输入的日期都是合法的，但不一定是合理的——假设已知镇上没有超过 200 岁的老人，而今天是 2014 年 9 月 6 日，所以超过 200 岁的生日和未出生的生日都是不合理的，应该被过滤掉。

输入格式：

输入在第一行给出正整数 *N*，取值在(0,105]；随后 *N* 行，每行给出 1 个人的姓名（由不超过 5 个英文字母组成的字符串）、以及按 `yyyy/mm/dd`（即年/月/日）格式给出的生日。题目保证最年长和最年轻的人没有并列。

输出格式：

在一行中顺序输出有效生日的个数、最年长人和最年轻人的姓名，其间以空格分隔。

输入样例：

```
5
John 2001/05/12
Tom 1814/09/06
Ann 2121/01/30
James 1814/09/05
Steve 1967/11/20
```

输出样例：

```
3 Tom John
```

核心代码如下：

```
#include <iostream>
#include <string.h>
using namespace std;

struct Document{
    char name[6];
    int yy, mm, dd;
};

bool LessDate(Document a, Document b){  //a 的日期小于 b，返回ture
```

```
    if(a.yy != b.yy) return a.yy <= b.yy;
    if(a.mm != b.mm) return a.mm <= b.mm;
    else return a.dd <= b.dd;
}

bool MoreDate(Document a, Document b){  //a 的日期大于 b，返回ture
    if(a.yy != b.yy) return a.yy >= b.yy;
    if(a.mm != b.mm) return a.mm >= b.mm;
    else return a.dd >= b.dd;
}

int main(){
    Document oldest, youngest;      //最年长、最年轻
    Document left, right, person;   //年龄的左右边界
    oldest.yy = right.yy = 2014;    //初始化
    youngest.yy = left.yy = 1814;
    oldest.mm = right.mm = youngest.mm = left.mm = 9;
    oldest.dd = right.dd = youngest.dd = left.dd = 6;
    int n,count=0;                  //count记录合法日期人数
    scanf("%d", &n);
    for(int i=0;i<n;i++){
        scanf("%s %d/%d/%d", &person.name, &person.yy, &person.mm,
&person.dd);
        if(LessDate(person, right) && MoreDate(person, left)){
            count++;
            if(LessDate(person, oldest)) oldest = person;       //无需按元
素分开赋值
            if(MoreDate(person, youngest)) youngest = person;
        }
    }
    if(count == 0) printf("0");       //存在所有人的日期都不合法的极端情况
    else printf("%d %s %s\n", count, oldest.name, youngest.name);
    return 0;
}
```

二、甲类题目：

1、Sign In and Sign Out

At the beginning of every day, **the first person** who signs in the computer room will unlock the door, and **the last one** who signs out will lock the door. Given the records of signing in's and out's, you are supposed to find the ones who have unlocked and locked the door on that day.

Input Specification:

Each input file contains one test case. Each case contains the records for one day.

The case starts with a positive integer *M*, which is the total number of records, followed by *M* lines, each in the format:

```
ID_number Sign_in_time Sign_out_time
```

where times are given in the format `HH:MM:SS`, and `ID_number` is a string with **no more than 15 characters**.

Output Specification:

For each test case, output in one line the **ID numbers** of the persons who have unlocked and locked the door on that day. The two ID numbers must be separated by one space.

Note: It is guaranteed that the records are **consistent**. That is, the sign in time must be earlier than the sign out time for each person, and there are no two persons sign in or out at the same moment.

Sample Input:

```
3
CS301111 15:30:28 17:00:10
SC3021234 08:00:00 11:25:25
CS301133 21:45:00 21:58:40
```

Sample Output:

```
SC3021234 CS301133
```

Key Codes:

```
#include <iostream>
using namespace std;
struct Student{
    char id[16];
    int in_hh, in_mm, in_ss;
    int out_hh, out_mm, out_ss;
};
//若a的进入时间更早，则返回true;函数构造同上面的人口普查中的LessDate
bool EarlierTime(Student a, Student b){}
//若a的退出时间更晚，则返回true;函数构造同上面的人口普查中的MoreDate
bool LaterTime(Student a, Student b){}

int main(){
    int n;
    scanf("%d", &n);
    Student student, unlocked, locked;

    unlocked.in_hh = 23;                    //初始化
    unlocked.in_mm = unlocked.in_ss = 59;
    locked.out_hh = locked.out_mm = locked.out_ss =0;

    for(int i=0;i<n;i++){
```

```
        scanf("%s %d:%d:%d %d:%d:%d", &student.id, &student.in_hh,
&student.in_mm, &student.in_ss, &student.out_hh, &student.out_mm,
&student.out_ss);
        if(EarlierTime(student, unlocked)) unlocked = student;
        if(LaterTime(student, locked)) locked = student;
    }
    printf("%s %s\n", unlocked.id, locked.id);
    return 0;
}
```

2、World Cup Betting

Chinese Football Lottery provided **a "Triple Winning" game**. The rule of winning was simple:

First select any three of the games. **Then** for each selected game, **bet on one** of the three possible results -- namely `W` for win, `T` for tie, and `L` for lose. There was an **odd** assigned to each result. The winner's odd would be **the product of the three odds times 65%**.

For example, 3 games' odds are given as the following:

```
 W    T    L
1.1  2.5  1.7
1.2  3.1  1.6
4.1  1.2  1.1
```

To obtain the maximum profit, one must buy `W` for the 3rd game, `T` for the 2nd game, and `T` for the 1st game. If each bet takes 2 yuans, then the maximum profit would be (**4.1×3.1×2.5×65%−1)×2=39.31** yuans (accurate up to 2 decimal places).

Input Specification:

Each input file contains one test case. Each case contains the betting information of 3 games. Each game occupies a line with three **distinct odds** corresponding to `W`, `T` and `L`.

Output Specification:

For each test case, print in one line the best bet of each game, and the maximum profit **accurate up to 2 decimal places**. The characters and the number must be separated by one space.

Sample Input:

```
1.1 2.5 1.7
1.2 3.1 1.6
4.1 1.2 1.1
```

Sample Output:

```
T T W 39.31
```

Key Codes:

```
#include <iostream>
using namespace std;
int main(){
    double max[3] = {0}, odds[3] = {0}, profits = 1;
    int max_idx[3];
    char g_result[3] = {'W', 'T', 'L'};

    for(int i=0;i<3;i++){
        scanf("%lf%lf%lf", &odds[0], &odds[1], &odds[2]);

        if(odds[0]-odds[1] > 1e-15) max_idx[i] = 0;
        else max_idx[i] = 1;
        if(odds[max_idx[i]]-odds[2] < 1e-15) max_idx[i] = 2;
        max[i] = odds[max_idx[i]];
        profits = profits * max[i];
    }
    profits = (profits*0.65 - 1)*2;
    printf("%c %c %c %.2f", g_result[max_idx[0]], g_result[max_idx[1]],
g_result[max_idx[2]], profits);
    return 0;
}
```

3、Boys vs Girls

This time you are asked to tell the difference between **the lowest grade of all the male students** and **the highest grade of all the female students**.

Input Specification:

Each input file contains one test case. Each case contains **a positive integer *N*,** followed by ***N* lines of student information**.

- Each line contains **a student's** `name`**,** `gender`**,** `ID` **and** `grade`, separated by a space
- `name` and `ID` are strings of **no more than 10 characters** with no space,
- `gender` is either `F` (female) or `M` (male)
- `grade` is **an integer between 0 and 100**.

It is guaranteed that all the grades are **distinct**.

Output Specification:

For each test case, output in 3 lines.

- The first line gives the name and ID of the **female student** with the highest grade
- the second line gives that of the **male student** with the lowest grade.

- The third line gives the difference $grade_F - grade_M$.
- **If one such kind of student is missing,** output `Absent` in the corresponding line, and output `NA` in the third line instead.

Sample Input :

```
3
Joe M Math990112 89
Mike M CS991301 100
Mary F EE990830 95
```

Sample Output :

```
Mary EE990830
Joe Math990112
6
```

Key Codes:

```
#include <iostream>
using namespace std;
struct Student{
    char name[11], ID[11];;
    char gender;
    int grade;
};
int main(){
    int n, M_num=0, F_num=0;
    Student student, boy, girl, lowest, highest;
    lowest.grade = 100;
    highest.grade = 0;

    scanf("%d", &n);
    for(int i=0;i<n;i++){
        scanf("%s %c %s %d", &student.name, &student.gender, &student.ID,
&student.grade);
        if(student.gender == 'M'){
            M_num++;    //统计男女生人数
            if(student.grade < lowest.grade) lowest = student;
        }
        else{
            F_num++;
            if(student.grade > highest.grade) highest = student;
        }
    }
    if(M_num != 0 && F_num != 0)
        printf("%s %s\n%s %s\n%d", highest.name, highest.ID, lowest.name,
lowest.ID, highest.grade-lowest.grade);
    else if(M_num == 0) printf("%s %s\nAbsent\nNA", highest.name,
highest.ID);
```

```
    else if(F_num == 0) printf("Absent\n%s %s\nNA", lowest.name,
lowest.ID);
    return 0;
}
```