PAT_A1012. The Best Rank (25)

1.Abstraction

1.1 Algorithm and idea

```
问题类型：
1.排序
2.按标签多次排序
```

1.2 Notice

```
* None
```

2.Problem:The Best Rank

To evaluate the performance of our first year CS majored students, we consider their grades of three courses only: C - C Programming Language, M - Mathematics (Calculus or Linear Algebra), and E - English. At the mean time, we encourage students by emphasizing on their best ranks -- that is, among the four ranks with respect to the three courses and the average grade, we print the best rank for each student.

For example, The grades of C, M, E and A - Average of 4 students are given as the following:

```
StudentID  C  M  E  A
310101     98 85 88 90
310102     70 95 88 84
310103     82 87 94 88
310104     91 91 91 91
```

Then the best ranks for all the students are *No.1* since the 1st one has done the best in C Programming Language, while the 2nd one in Mathematics, the 3rd one in English, and the last one in average.

Input

Each input file contains one test case. Each case starts with a line containing 2 numbers N and M (<=2000), which are the total number of students, and the number of students who would check their ranks, respectively. Then N lines follow, each contains a student ID which is a string of 6 digits, followed by the three integer grades (in the range of [0, 100]) of that student in the order of C, M and E. Then there are M lines, each containing a student ID.

Output

For each of the M students, print in one line the best rank for him/her, and the symbol of the corresponding rank, separated by a space.

The priorities of the ranking methods are ordered as A > C > M > E. Hence if there are two or more ways for a student to obtain the same best rank, output the one with the highest priority.

If a student is not on the grading list, simply output "N/A".

Sample Input

```
5 6
310101 98 85 88
310102 70 95 88
310103 82 87 94
310104 91 91 91
310105 85 90 90
310101
310102
310103
310104
310105
999999
```

Sample Output

```
1 C
1 M
1 E
1 A
3 A
N/A
```

3. Algorithm note

```
* None
```

4. Code

4.1 Edit 0:原址排序，单独记录

4.1.1 Algorithm abstraction

```
1.预先分配一个存储数组和4个排序数组
2.在存储数组中按照标签排序，每次排序完后将名次记录到排序数组中去。
```

4.1.2 Notice

```
* None
```

4.1.3 Code Block

```
#include <iostream>
#include <cstdio>
#include <cmath>
#include <algorithm>

using std::sort;
using std::cin;
using std::cout;
using std::endl;

struct Student{
    int id;//存放六位整数的ID
    int grade[4];//存放四个分数
}stu[2010];

char course[4]={'A','C','M','E'};//按优先级顺序，方便输出
int Rank[10000000][4]={0};//Rank[id][0]-Rank[id][4]为4门课对应的排名
int now;//cmp函数中使用,表示当前按now号分数排序stu数组
//注意这个函数的写法
bool cmp(Student a,Student b){
    return a.grade[now]>b.grade[now];
}

int main(){
    int n,m;
    scanf("%d %d",&n,&m);
    //读入分数，其中grade[0]-grade[3]分别代表A,C,M,E
    for(int i=0;i<n;++i){
        scanf("%d %d %d
%d",&stu[i].id,&stu[i].grade[1],&stu[i].grade[2],&stu[i].grade[3]);

stu[i].grade[0]=round((stu[i].grade[1]+stu[i].grade[2]+stu[i].grade[3])/3.0)+0.5;//rou
nd是四舍五入函数

    }
    for(now=0;now<4;now++){
        sort(stu,stu+n,cmp);
        Rank[stu[0].id][now]=1;//排序完，将最高分数的设为rank1
        for(int i=1;i<n;++i){
            if(stu[i].grade[now]==stu[i-1].grade[now]){
                Rank[stu[i].id][now]=Rank[stu[i-1].id][now];
            }else{
                Rank[stu[i].id][now]=i+1;
            }
        }
    }
    int query;//查询的考生id
    for(int i=0;i<m;++i){
        scanf("%d",&query);
        if(Rank[query][0]==0){
            printf("N/A\n");
        }else{
            int k=0;
            for(int j=0;j<4;++j){
                if(Rank[query][j]<Rank[query][k]){
                    k=j;
                }
            }
            printf("%d %c\n",Rank[query][k],course[k]);
        }
    }
    return 0;
}
```

4.2 Edit 1:指针数组排序，原址记录

4.2.1 Algorithm abstraction

```
1.预先分配一个记录数组和一个指针数组。
2.对指针数组进行排序，排序完后将名次写回记录数组中去。
```

4.2.2 Notice

```
* None
```

4.2.3 Code Block

```
#include <iostream>
#include <vector>
```

```
#include <algorithm>

using std::vector;
using std::sort;
using std::cin;
using std::cout;
using std::endl;
using std::ceil;
const int maxn=1000010;
char na[4]={'A','C','M','E'};
struct Stu{
    int grades[4];
    int rank[4];
    int lrank;
    Stu(){
        grades[0]=-1;
        lrank=-1;
    }
};
vector<Stu> stu(maxn);
vector<Stu*> stu_re(maxn);
int index=0;
int cmp(Stu *s1,Stu *s2){
    return s1->grades[index]>s2->grades[index];
}
void write_rank(vector<Stu*> &stu_re,int num,int p){
    index=p;
    sort(stu_re.begin(),stu_re.begin()+num,cmp);
    stu_re[0]->rank[index]=1;
    for(int i=1;i<num;++i){
        if(stu_re[i]->grades[index]==stu_re[i-1]->grades[index]){
            stu_re[i]->rank[index]=stu_re[i-1]->rank[index];
        }
        else{
            stu_re[i]->rank[index]=i+1;
        }
    }

}
int main(){
    int id;
    int stu_num,query_num;
    cin>>stu_num>>query_num;
    for(int i=0;i<stu_num;++i){
        cin>>id;
        cin>>stu[id].grades[1]>>stu[id].grades[2]>>stu[id].grades[3];

stu[id].grades[0]=ceil((stu[id].grades[1]+stu[id].grades[2]+stu[id].grades[3])/3);
        stu_re[i]=&stu[id];
    }
    for(int i=0;i<4;++i){
        write_rank(stu_re,stu_num,i);

    }
    for(int i=0;i<maxn;++i){
        if(stu[i].grades[0]==-1){
            continue;
        }
        else{
            int min_rank;
            int min=stu_num;
            for(int j=0;j<4;++j){
                if(min>stu[i].rank[j]){
                    min=stu[i].rank[j];
                    min_rank=j;
                }
            }
            stu[i].lrank=min_rank;
        }
    }
    for(int i=0;i<query_num;++i){
        cin>>id;
        if(stu[id].grades[0]!=-1){
            cout<<stu[id].rank[stu[id].lrank]<<" "<<na[stu[id].lrank]<<endl;
        }
        else{
            cout<<"N/A"<<endl;
        }
    }
    return 0;

}
```

5. Summary

Edit 0的代码要比Edit 1简练很多，要试着联系使代码紧凑简练。

比如以下两段代码，都是为了寻找排名最考前的学科编号。

```
    int query;//查询的考生id
    for(int i=0;i<m;++i){
        scanf("%d",&query);
        if(Rank[query][0]==0){
            printf("N/A\n");
        }else{
            int k=0;
            for(int j=0;j<4;++j){
                if(Rank[query][j]<Rank[query][k]){
                    k=j;
                }
            }

            printf("%d %c\n",Rank[query][k],course[k]);
        }
    }
```

```
    for(int i=0;i<maxn;++i){
        if(stu[i].grades[0]==-1){
            continue;
        }
        else{
            int min_rank;
            int min=stu_num;
            for(int j=0;j<4;++j){
                if(min>stu[i].rank[j]){
                    min=stu[i].rank[j];
                    min_rank=j;
                }
            }
            stu[i].lrank=min_rank;
        }
    }
```