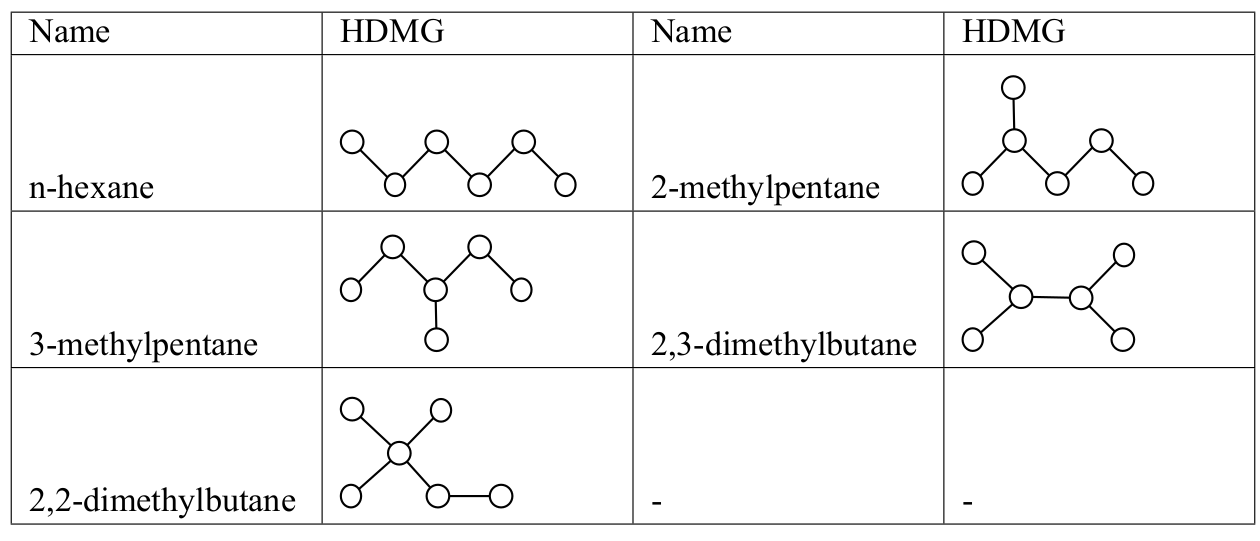
[A - 化学 (编译器选 GNU G++)](https://vjudge.net/problem/Gym-270437A)

<https://vjudge.net/contest/359621#problem/A>

化学很神奇，以下是烷烃基。



假设如上图，这个烷烃基有6个原子和5个化学键，6个原子分别标号1~6，然后用一对数字 a,b 表示原子a和原子b间有一个化学键。这样通过5行a,b可以描述一个烷烃基

你的任务是甄别烷烃基的类别。

原子没有编号方法，比如  
1 2  
2 3  
3 4  
4 5  
5 6  
和  
1 3  
2 3  
2 4  
4 5  
5 6  
是同一种，本质上就是一条链，编号其实是没有关系的，可以在纸上画画就懂了

**Input**

输入第一行为数据的组数T（1≤T≤200000）。每组数据有5行，每行是两个整数a, b(1≤a,b≤6,a ≤b)

数据保证，输入的烷烃基是以上5种之一

**Output**

每组数据，输出一行，代表烷烃基的英文名

**Example**

Input

2

1 2

2 3

3 4

4 5

5 6

1 4

2 3

3 4

4 5

5 6

**Output**

n-hexane

3-methylpentane

**解题思路：**

仔细观察五种同分异构体，发现最主要的区别在于原子的杂化方式（相邻连接的原子数）和不同类型原子的个数，其中有三个（n-hexane, 2,2-dimethylbutane, 2,3-dimethylbutane）可以通过判断具有各种分支数点的个数直接判断出来，而剩下两个则需要判断三分支点的所有邻接点中为二分支点个数，以此来区分。

每个烷烃对应一个图，图以邻接矩阵进行存储，确定每个点邻接点个数然后进行后面的判断

**完整代码；**

#include "iostream"

using namespace std;

int have3node;

int main() {

    int n, a, b;

    cin >> n;

    for (int i = 0; i < n; i++) {

        int num[6] = {0};  //存每个节点有几个连接的点

        int numOf1 = 0, numOf2 = 0, numOf3 = 0,

            numOf4 = 0;  //比如numOf3就是有三个相邻点的节点数

        bool g[6][6];    //邻接矩阵

        //初始化图

        for (int j = 0; j < 6; j++)

            for (int k = 0; k < 6; k++) g[j][k] = false;

        //连接节点

        for (int u = 0; u < 5; u++) {

            cin >> a >> b;

            g[a - 1][b - 1] = true;

            g[b - 1][a - 1] = true;

        }

        for (int x = 0; x < 6; x++) {

            for (int y = 0; y < 6; y++)

                if (g[x][y]) num[x]++;

            switch (num[x]) {

                case 1:

                    numOf1++;

                    break;

                case 2:

                    numOf2++;

                    break;

                case 3:

                    numOf3++;

                    have3node = x;

                    break;

                case 4:

                    numOf4++;

                    break;

                default:

                    break;

            }

        }

        int count = 0;

        if (numOf3 ==1) {

            for (int m = 0; m < 6; m++) {

                if (g[have3node][m]) {

                    if (num[m] == 2) count++;

                }

            }

        }

        if (numOf4 == 1)

            cout << "2,2-dimethylbutane\n";

        else if (numOf3 == 2)

            cout << "2,3-dimethylbutane\n";

        else if (numOf1 == 2)

            cout << "n-hexane\n";

        else if (count == 2)

            cout << "3-methylpentane\n";

        else

            cout << "2-methylpentane\n";

    }

    system("pause");

    return 0;

}

[B - 爆零(×)大力出奇迹(√)](https://vjudge.net/problem/HDU-2093)

<https://vjudge.net/contest/359621#problem/B>

程序设计思维作业和实验使用的实时评测系统，具有及时获得成绩排名的特点，那它的功能是怎么实现的呢？  
我们千辛万苦怼完了不忍直视的程序并提交以后，评测系统要么返回AC，要么是返回各种其他的错误，不论是怎样的错法，它总会给你记上一笔，表明你曾经在这儿被坑过，而当你历经千辛终将它AC之后，它便会和你算笔总账,表明这题共错误提交了几次。  
在岁月的长河中，你通过的题数虽然越来越多，但通过每题时你所共花去的时间（从最开始算起，直至通过题目时的这段时间）都会被记录下来，作为你曾经奋斗的痕迹。特别的，对于你通过的题目，你曾经的关于这题的每次错误提交都会被算上一定的单位时间罚时，这样一来，你在做出的题数上，可能领先别人很多，但是在做出同样题数的人中，你可能会因为罚时过高而处于排名上的劣势。  
例如某次考试一共八道题（A,B,C,D,E,F,G,H），每个人做的题都在对应的题号下有个数量标记，负数表示该学生在该题上有过的错误提交次数但到现在还没有AC，正数表示AC所耗的时间，如果正数a跟上了一对括号，里面有个正数b,则表示该学生AC了这道题，耗去了时间a，同时曾经错误提交了b次。例子可见下方的样例输入与输出部分。

**Input**

输入数据包含多行，第一行是共有的题数n（1≤n≤12）以及单位罚时m（10≤m≤20），之后的每行数据描述一个学生的信息，首先是学生的用户名（不多于10个字符的字串）其次是所有n道题的得分现状，其描述采用问题描述中的数量标记的格式，见上面的表格。

**Output**

根据这些学生的得分现状，输出一个实时排名。实时排名显然先按AC题数的多少排，多的在前，再按时间分的多少排，少的在前，如果凑巧前两者都相等，则按名字的字典序排，小的在前。每个学生占一行，输出名字（10个字符宽），做出的题数（2个字符宽，右对齐）和时间分（4个字符宽，右对齐）。名字、题数和时间分相互之间有一个空格。数据保证可按要求的输出格式进行输出。

**Sample Input**

8 20

GuGuDong 96 -3 40(3) 0 0 1 -8 0

hrz 107 67 -3 0 0 82 0 0

TT 120(3) 30 10(1) -3 0 47 21(2) -2

OMRailgun 0 -99 -8 0 -666 -10086 0 -9999996

yjq -2 37(2) 13 -1 0 113(2) 79(1) -1

Zjm 0 0 57(5) 0 0 99(3) -7 0

**Sample Output**

TT 5 348

yjq 4 342

GuGuDong 3 197

hrz 3 256

Zjm 2 316

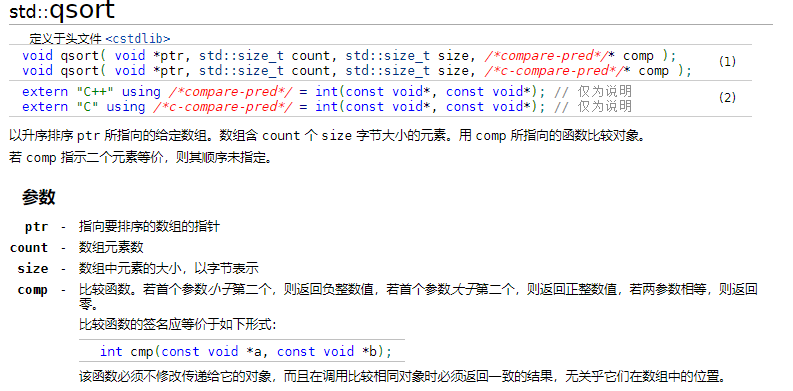
OMRailgun 0 0

OMRailgun 0 0

**解题思路：**

按照每个人的答题状况确定AC数和分数，并存储在数组中，然后排序输出，这里再次复习了格式化输出。

读取带有罚时的分数是可以用res = sscanf(str, "%d(%d)", &h, &t);来实现，使用qsort函数时，作为参数的函数cmp()的参数必须是const型的



**完整代码：**

#include <iostream>

using namespace std；

struct student {

    char name[15];

    int AC\_Count;

    int score;

};

/\*https://zh.cppreference.com/w/cpp/algorithm/qsort\*/

/\*必须是const型参数\*/

int cmp(const void \*a, const void \*b) {

    struct student \*aa = (student \*)a;

    struct student \*bb = (student \*)b;

    if (aa->AC\_Count != bb->AC\_Count)

        return bb->AC\_Count - aa->AC\_Count;

    else if (aa->score != bb->score)

        return aa->score - bb->score;

    else

        return aa->name - bb->name;

}

int n, m;

student stu[10001];

int main() {

    int  j, t,h,  res, num = 0;

    char str[20];

    scanf("%d%d", &n, &m);

    while (scanf("%s", &stu[num].name) != EOF) {

        stu[num].AC\_Count = 0;

        stu[num].score = 0;

        for (int i = 0; i < n; i++) {

            scanf("%s", str);

            res = sscanf(str, "%d(%d)", &h, &t);

            if (res == 2) {

                stu[num].AC\_Count++;

                stu[num].score += h;

                stu[num].score += m \* t;

            } else if (res == 1 && h > 0) {

                stu[num].AC\_Count++;

                stu[num].score += h;

            }

        }

        num++;

        /\*qsort(stu, num, sizeof(stu[0]), cmp);

        for (i = 0; i < num; i++) {

            printf("%-10s %2d %4d\n", stu[i].name, stu[i].AC\_Count,

        stu[i].score);

        }\*/

    }

    qsort(stu, num, sizeof(stu[0]), cmp);

    for (int i = 0; i < num; i++) {

        printf("%-10s %2d %4d\n", stu[i].name, stu[i].AC\_Count, stu[i].score);

    }

    return 0;

}

[C - 瑞神打牌 （不支持C++11，G++和C++编译器都试试）](https://vjudge.net/problem/POJ-1786)

<https://vjudge.net/contest/359621#problem/C>

 瑞神HRZ因为疫情在家闲得无聊，同时他又非常厉害，所有的课对他来说都是水一水就能拿A+，所以他无聊，找来了另外三个人：咕咕东，腾神以及zjm来打牌（天下苦瑞神久矣）。  
显然，牌局由四个人构成，围成一圈。我们称四个方向为北 东 南 西。对应的英文是North，East，South，West。游戏一共由一副扑克，也就是52张构成。开始，我们指定一位发牌员（东南西北中的一个，用英文首字母标识）开始发牌，发牌顺序为顺时针，发牌员第一个不发自己，而是发他的下一个人（顺时针的下一个人）。这样，每个人都会拿到13张牌。  
现在我们定义牌的顺序，首先，花色是（梅花）<（方片）<（黑桃）<（红桃），（输入时，我们用C,D,S,H分别表示梅花，方片，黑桃，红桃，即其单词首字母）。对于牌面的值，我们规定2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < T < J < Q < K < A。  
现在你作为上帝，你要从小到大排序每个人手中的牌，并按照给定格式输出。（具体格式见输出描述和样例输出）。

**Input**

输入包含多组数据  
每组数据的第一行包含一个大写字符，表示发牌员是谁。如果该字符为‘#’则表示输入结束。  
接下来有两行，每行有52个字符，表示了26张牌，两行加起来一共52张牌。每张牌都由两个字符组成，第一个字符表示花色，第二个字符表示数值。

**Output**

输出多组数据发牌的结果，每组数据之后需要额外多输出一个空行！！！！！  
每组数据应该由24行的组成，输出按照顺时针方向，始终先输出South Player的结果，每位玩家先输出一行即玩家名称（东南西北），接下来五行，第一行和第五行输出固定格式（见样例），第二行和第四行按顺序和格式输出数值（见样例），第三行按顺序和格式输出花色（见样例）。

**Sample Input**

N

CTCAH8CJD4C6D9SQC7S5HAD2HJH9CKD3H6D6D7H3HQH4C5DKHKS9

SJDTS3S7S4C4CQHTSAH2D8DJSTSKS2H5D5DQDAH7C9S8C8S6C2C3

#

**Sample Output**

South player:

+---+---+---+---+---+---+---+---+---+---+---+---+---+

|6 6|A A|6 6|J J|5 5|6 6|7 7|9 9|4 4|5 5|7 7|9 9|T T|

| C | C | D | D | S | S | S | S | H | H | H | H | H |

|6 6|A A|6 6|J J|5 5|6 6|7 7|9 9|4 4|5 5|7 7|9 9|T T|

+---+---+---+---+---+---+---+---+---+---+---+---+---+

West player:

+---+---+---+---+---+---+---+---+---+---+---+---+---+

|2 2|5 5|9 9|K K|5 5|7 7|9 9|4 4|T T|J J|A A|8 8|A A|

| C | C | C | C | D | D | D | S | S | S | S | H | H |

|2 2|5 5|9 9|K K|5 5|7 7|9 9|4 4|T T|J J|A A|8 8|A A|

+---+---+---+---+---+---+---+---+---+---+---+---+---+

North player:

+---+---+---+---+---+---+---+---+---+---+---+---+---+

|3 3|4 4|J J|2 2|3 3|T T|Q Q|K K|8 8|Q Q|K K|2 2|3 3|

| C | C | C | D | D | D | D | D | S | S | S | H | H |

|3 3|4 4|J J|2 2|3 3|T T|Q Q|K K|8 8|Q Q|K K|2 2|3 3|

+---+---+---+---+---+---+---+---+---+---+---+---+---+

East player:

+---+---+---+---+---+---+---+---+---+---+---+---+---+

|7 7|8 8|T T|Q Q|4 4|8 8|A A|2 2|3 3|6 6|J J|Q Q|K K|

| C | C | C | C | D | D | D | S | S | H | H | H | H |

|7 7|8 8|T T|Q Q|4 4|8 8|A A|2 2|3 3|6 6|J J|Q Q|K K|

+---+---+---+---+---+---+---+---+---+---+---+---+---+

**解题思路：**

定义一个扑克牌的结构体，包含花色和数值，，再定义一个52张牌的数组表示一副牌每次直接读取一副牌的信息然后按间隔为4分发给四个人。

struct poke  //扑克牌的牌面

{

    char num;

    char color;

};

for (int i = 0; i < 52; i++) {

    cin >> set[i].color >> set[i].num;

}

牌的排序部分由map完成，将每张牌牌面数字和花色对应具体大小的数字方便比较。这其中利用了isdigit()函数和isalpha()函数来判断数字和字母

bool cmp(poke a, poke b) {

    if (a.color == b.color) {

        if ((isdigit(a.num) && isdigit(b.num)) ||

            (isdigit(b.num) && isalpha(a.num)) ||

            (isdigit(a.num) && isalpha(b.num)))

            return a.num < b.num;

        else if (isalpha(a.num) && isalpha(b.num))

            return m[a.num] < m[b.num];

    } else {

        return mm[a.color] < mm[b.color];

    }

}

**完整代码：**

#include <algorithm>

#include <iostream>

#include <map>

#include <vector>

using namespace std;

struct poke  //扑克牌的牌面

{

    char num;

    char color;

};

//各个人的牌

vector<poke> north;

vector<poke> east;

vector<poke> south;

vector<poke> west;

map<char, int> m;

map<char, int> mm;

bool cmp(poke a, poke b) {

    if (a.color == b.color) {

        if ((isdigit(a.num) && isdigit(b.num)) ||

            (isdigit(a.num) && isalpha(b.num)) ||

            (isdigit(b.num) && isalpha(a.num)))

            return a.num < b.num;

        else if (isalpha(a.num) && isalpha(b.num))

            return m[a.num] < m[b.num];

    } else {

        return mm[a.color] < mm[b.color];

    }

}

poke set[52];

int main() {

    char dir;

    string str1, str2, str;

    m['T'] = 1, m['J'] = 2, m['Q'] = 3, m['K'] = 4, m['A'] = 5;

    mm['C'] = 1, mm['D'] = 2, mm['S'] = 3, mm['H'] = 4;

    while (cin >> dir) {

        if (dir == '#') break;

        for (int i = 0; i < 52; i++) {

            cin >> set[i].color >> set[i].num;

        }

        east.clear(), north.clear(), south.clear(), west.clear();

        /\*

        cin >> str1 >> str2;

        str = str1 + str2;\*/

        if (dir == 'N')  //如果是N发牌，则派牌的顺序是ESWN下面的则照顺时针推出

        {

            for (int i = 0; i < 52; i++) {

                if (i % 4 == 0)

                    east.push\_back(set[i]);

                else if (i % 4 == 1)

                    south.push\_back(set[i]);

                else if (i % 4 == 2)

                    west.push\_back(set[i]);

                else if (i % 4 == 3)

                    north.push\_back(set[i]);

            }

        } else if (dir == 'W') {

            for (int i = 0; i < 52; i++) {

                if (i % 4 == 0)

                    north.push\_back(set[i]);

                else if (i % 4 == 1)

                    east.push\_back(set[i]);

                else if (i % 4 == 2)

                    south.push\_back(set[i]);

                else if (i % 4 == 3)

                    west.push\_back(set[i]);

            }

        } else if (dir == 'E') {

            for (int i = 0; i < 52; i++) {

                if (i % 4 == 0)

                    south.push\_back(set[i]);

                else if (i % 4 == 1)

                    west.push\_back(set[i]);

                else if (i % 4 == 2)

                    north.push\_back(set[i]);

                else if (i % 4 == 3)

                    east.push\_back(set[i]);

            }

        } else if (dir == 'S') {

            for (int i = 0; i < 52; i++) {

                if (i % 4 == 0)

                    west.push\_back(set[i]);

                else if (i % 4 == 1)

                    north.push\_back(set[i]);

                else if (i % 4 == 2)

                    east.push\_back(set[i]);

                else if (i % 4 == 3)

                    south.push\_back(set[i]);

            }

        }

        sort(south.begin(), south.end(), cmp);

        sort(west.begin(), west.end(), cmp);

        sort(north.begin(), north.end(), cmp);

        sort(east.begin(), east.end(), cmp);

        cout << "South player:" << endl;

        cout << "+---+---+---+---+---+---+---+---+---+---+---+---+---+" << endl;

        for (int i = 0; i < 13; i++)

            cout << "|" << south[i].num << " " << south[i].num;

        cout << "|" << endl;

        for (int i = 0; i < 13; i++) cout << "| " << south[i].color << " ";

        cout << "|" << endl;

        for (int i = 0; i < 13; i++)

            cout << "|" << south[i].num << " " << south[i].num;

        cout << "|" << endl;

        cout << "+---+---+---+---+---+---+---+---+---+---+---+---+---+" << endl;

        cout << "West player:" << endl;

        cout << "+---+---+---+---+---+---+---+---+---+---+---+---+---+" << endl;

        for (int i = 0; i < 13; i++)

            cout << "|" << west[i].num << " " << west[i].num;

        cout << "|" << endl;

        for (int i = 0; i < 13; i++) cout << "| " << west[i].color << " ";

        cout << "|" << endl;

        for (int i = 0; i < 13; i++)

            cout << "|" << west[i].num << " " << west[i].num;

        cout << "|" << endl;

        cout << "+---+---+---+---+---+---+---+---+---+---+---+---+---+" << endl;

        cout << "North player:" << endl;

        cout << "+---+---+---+---+---+---+---+---+---+---+---+---+---+" << endl;

        for (int i = 0; i < 13; i++)

            cout << "|" << north[i].num << " " << north[i].num;

        cout << "|" << endl;

        for (int i = 0; i < 13; i++) cout << "| " << north[i].color << " ";

        cout << "|" << endl;

        for (int i = 0; i < 13; i++)

            cout << "|" << north[i].num << " " << north[i].num;

        cout << "|" << endl;

        cout << "+---+---+---+---+---+---+---+---+---+---+---+---+---+" << endl;

        cout << "East player:" << endl;

        cout << "+---+---+---+---+---+---+---+---+---+---+---+---+---+" << endl;

        for (int i = 0; i < 13; i++)

            cout << "|" << east[i].num << " " << east[i].num;

        cout << "|" << endl;

        for (int i = 0; i < 13; i++) cout << "| " << east[i].color << " ";

        cout << "|" << endl;

        for (int i = 0; i < 13; i++)

            cout << "|" << east[i].num << " " << east[i].num;

        cout << "|" << endl;

        cout << "+---+---+---+---+---+---+---+---+---+---+---+---+---+" << endl;

        cout << endl;

    }

    // system("pause");

}

/\*

N

CTCAH8CJD4C6D9SQC7S5HAD2HJH9CKD3H6D6D7H3HQH4C5DKHKS9

SJDTS3S7S4C4CQHTSAH2D8DJSTSKS2H5D5DQDAH7C9S8C8S6C2C3

#

\*/