1061 判断题 (15分)

1061 判断题 (15分)

判断题的评判很简单，本题就要求你写个简单的程序帮助老师判题并统计学生们判断题的得分。

输入格式:

输入在第一行给出两个不超过 100 的正整数 N 和 M，分别是学生人数和判断题数量。第二行给出 M 个不超过 5 的正整数，是每道题的满分值。第三行给出每道题对应的正确答案，0 代表“非”，1 代表“是”。随后 N 行，每行给出一个学生的解答。数字间均以空格分隔。

输出格式:

按照输入的顺序输出每个学生的得分，每个分数占一行。

输入样例:

```
3 6
2 1 3 3 4 5
0 0 1 0 1 1
0 1 1 0 0 1
1 0 1 0 1 0
1 1 0 0 1 1
```

输出样例:

```
13
11
12
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int n,m;
    cin >> n >> m;
    int score[m];
    int ans[n][m];
    for(int i = 0;i < m;i ++) cin >> score[i];
    int right[m];
    for(int i = 0;i < m;i ++) cin >> right[i];
    for(int i = 0;i < n;i ++)
    {
        int res = 0;
        for(int j = 0; j < m;j ++)
        {
            int x;
```

```
            cin >> x;
            if(x == right[j]) res += score[j];
        }
        cout << res << endl;
    }
}
```

1062 最简分数 (20分)

1062 最简分数 (20分)

一个分数一般写成两个整数相除的形式：N/M，其中 M 不为0。最简分数是指分子和分母没有公约数的分数表示形式。

现给定两个不相等的正分数 N_1/M_1 和 N_2/M_2，要求你按从小到大的顺序列出它们之间分母为 K 的最简分数。

输入格式:

输入在一行中按 N/M 的格式给出两个正分数，随后是一个正整数分母 K，其间以空格分隔。题目保证给出的所有整数都不超过 1000。

输出格式:

在一行中按 N/M 的格式列出两个给定分数之间分母为 K 的所有最简分数，按从小到大的顺序，其间以 1 个空格分隔。行首尾不得有多余空格。题目保证至少有 1 个输出。

输入样例:

```
7/18 13/20 12
```

输出样例:

```
5/12 7/12
```

1063 计算谱半径 (20分)

1063 计算谱半径 (20分)

在数学中，矩阵的“谱半径”是指其特征值的模集合的上确界。换言之，对于给定的 n 个复数空间的特征值 $\{a_1+b_1i,\cdots,a_n+b_ni\}$，它们的模为实部与虚部的平方和的开方，而“谱半径”就是最大模。

现在给定一些复数空间的特征值，请你计算并输出这些特征值的谱半径。

输入格式:

输入第一行给出正整数 N（$\leq$ 10 000）是输入的特征值的个数。随后 N 行，每行给出 1 个特征值的实部和虚部，其间以空格分隔。注意：题目保证实部和虚部均为绝对值不超过 1000 的整数。

输出格式:

在一行中输出谱半径，四舍五入保留小数点后 2 位。

输入样例:

```
5
0 1
2 0
-1 0
3 3
0 -3
```

输出样例:

```
4.24
```

```cpp
#include <iostream>
#include <cmath>

using namespace std;

int main() {
    int n;
    cin >> n;
    float max = 0;
    for(int i = 0; i < n; i++) {
        float a, b, ans;
        cin >> a >> b;
        ans = sqrt(a * a + b * b);
        max = ans > max ? ans : max;
    }
    printf("%.2f", max);
    return 0;
}
```

1064 朋友数 (20分)

1064 朋友数 (20分)

如果两个整数各位数字的和是一样的，则被称为是“朋友数”，而那个公共的和就是它们的“朋友证号”。例如 123 和 51 就是朋友数，因为 1+2+3 = 5+1 = 6，而 6 就是它们的朋友证号。给定一些整数，要求你统计一下它们中有多少个不同的朋友证号。

输入格式:

输入第一行给出正整数 N。随后一行给出 N 个正整数，数字间以空格分隔。题目保证所有数字小于 10^4。

输出格式:

首先第一行输出给定数字中不同的朋友证号的个数；随后一行按递增顺序输出这些朋友证号，数字间隔一个空格，且行末不得有多余空格。

输入样例:

```
8
123 899 51 998 27 33 36 12
```

输出样例:

```
4
3 6 9 26
```

```cpp
#include <iostream>
#include <set>

using namespace std;

int f(int num) {
    int sum = 0;
    while(num != 0) {
        sum += num % 10;
        num /= 10;
    }
    return sum;
}
int main() {
    set<int> s;
    int n, num;
    scanf("%d", &n);
    for(int i = 0; i < n; i++) {
        scanf("%d", &num);
        s.insert(f(num));
    }
    printf("%d\n", s.size());
    for(auto it = s.begin(); it != s.end(); it++) {
        if(it != s.begin()) printf(" ");
        printf("%d", *it);
    }
    return 0;
```

```
}
```

1065 单身狗 (25分)

1065 单身狗 (25分)

“单身狗”是中文对于单身人士的一种爱称。本题请你从上万人的大型派对中找出落单的客人，以便给予特殊关爱。

输入格式:

输入第一行给出一个正整数 N（≤ 50 000），是已知夫妻/伴侣的对数；随后 N 行，每行给出一对夫妻/伴侣——为方便起见，每人对应一个 ID 号，为 5 位数字（从 00000 到 99999），ID 间以空格分隔；之后给出一个正整数 M（≤ 10 000），为参加派对的总人数；随后一行给出这 M 位客人的 ID，以空格分隔。题目保证无人重婚或脚踩两条船。

输出格式:

首先第一行输出落单客人的总人数；随后第二行按 ID 递增顺序列出落单的客人。ID 间用 1 个空格分隔，行的首尾不得有多余空格。

输入样例:

```
3
11111 22222
33333 44444
55555 66666
7
55555 44444 10000 88888 22222 11111 23333
```

输出样例:

```
5
10000 23333 44444 55555 88888
```

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

using namespace std;

int main()
{
    int n;
    map<int,int> map;
    while(n --)
    {
        int a,b;
        cin >> a >> b;
        map.insert({a,b});
    }
    int t;
    cin >> t;
```

```
    while(t --)
    {

    }
}
```

1066 图像过滤 (15分)

返回

1066 图像过滤 (15分)

图像过滤是把图像中不重要的像素都染成背景色，使得重要部分被凸显出来。现给定一幅黑白图像，要求你将灰度值位于某指定区间内的所有像素颜色都用一种指定的颜色替换。

输入格式:

输入在第一行给出一幅图像的分辨率，即两个正整数 M 和 N $(0 < M, N \le 500)$，另外是待过滤的灰度值区间端点 A 和 B $(0 \le A < B \le 255)$、以及指定的替换灰度值。随后 M 行，每行给出 N 个像素点的灰度值，其间以空格分隔。所有灰度值都在 [0, 255] 区间内。

输出格式:

输出按要求过滤后的图像。即输出 M 行，每行 N 个像素灰度值，每个灰度值占 3 位（例如黑色要显示为 `000`），其间以一个空格分隔。行首尾不得有多余空格。

输入样例:

```
3 5 100 150 0
3 189 254 101 119
150 233 151 99 100
88 123 149 0 255
```

输出样例:

```
003 189 254 000 000
000 233 151 099 000
088 000 000 000 255
```

```
#include <iostream>

using namespace std;

int main()
{
    int m,n,a,b,c;
    cin >> m >> n >> a >> b >> c;
    int s[m][n];
    for(int i = 0;i < m; i ++)
    {
        for(int j = 0;j < n;j ++)
        {
            cin >> s[i][j];
            if(s[i][j] >= a && s[i][j] <= b) s[i][j] = c;
```

```
        }
    }

    for(int i = 0;i < m;i ++)
    {
        for(int j = 0;j < n;j ++)
            printf("%03d ",s[i][j]);
        cout << endl;
    }

}
```

1067 试密码 (20分)

当你试图登录某个系统却忘了密码时，系统一般只会允许你尝试有限多次，当超出允许次数时，账号就会被锁死。本题就请你实现这个小功能。

输入格式:

输入在第一行给出一个密码（长度不超过 20 的、不包含空格、Tab、回车的非空字符串）和一个正整数 N（≤ 10），分别是正确的密码和系统允许尝试的次数。随后每行给出一个以回车结束的非空字符串，是用户尝试输入的密码。输入保证至少有一次尝试。当读到一行只有单个 # 字符时，输入结束，并且这一行不是用户的输入。

输出格式:

对用户的每个输入，如果是正确的密码且尝试次数不超过 N，则在一行中输出 `Welcome in`，并结束程序；如果是错误的，则在一行中按格式输出 `Wrong password: 用户输入的错误密码`；当错误尝试达到 N 次时，再输出一行 `Account locked`，并结束程序。

输入样例 1:

```
Correct%pw 3
correct%pw
Correct@PW
whatisthepassword!
Correct%pw
#
```

输出样例 1:

```
Wrong password: correct%pw
Wrong password: Correct@PW
Wrong password: whatisthepassword!
Account locked
```

```
#include <iostream>

using namespace std;

int main()
{
```

```
    string p;
    int n;
    cin >> p >> n;
    int i;string s;
    for(i = 0;i < n;i ++)
    {
        cin >> s;
        if(s == p)
        {
            cout << "Welcome in" <<endl;;
            break;
        }

        else if(s != "#")
        {
            cout << "Wrong password: " << s << endl;
        }

        else break;
    }
    if(i == n && s != p) cout << "Account locked" <<endl;
}
```

1068 万绿丛中一点红 (20分)

1068 万绿丛中一点红 (20分)

对于计算机而言，颜色不过是像素点对应的一个 24 位的数值。现给定一幅分辨率为 $M \times N$ 的画，要求你找出万绿丛中的一点红，即有独一无二颜色的那个像素点，并且该点的颜色与其周围 8 个相邻像素的颜色差充分大。

输入格式:

输入第一行给出三个正整数，分别是 M 和 N (≤ 1000)，即图像的分辨率；以及 TOL，是所求像素点与相邻点的颜色差阈值，色差超过 TOL 的点才被考虑。随后 N 行，每行给出 M 个像素的颜色值，范围在 $[0, 2^{24})$ 内。所有同行数字间用空格或 TAB 分开。

输出格式:

在一行中按照 `(x, y): color` 的格式输出所求像素点的位置以及颜色值，其中位置 `x` 和 `y` 分别是该像素在图像矩阵中的列、行编号（从 1 开始编号）。如果这样的点不唯一，则输出 `Not Unique`；如果这样的点不存在，则输出 `Not Exist`。

输入样例 1:

```
8 6 200
0        0        0        0        0        0        0        0
65280    65280    65280    16711479 65280    65280    65280    65280
16711479 65280    65280    65280    16711680 65280    65280    65280
65280    65280    65280    65280    65280    65280    165280   165280
65280    65280    16777015 65280    65280    165280   65480    165280
16777215 16777215 16777215 16777215 16777215 16777215 16777215 16777215
```

输出样例 1:

```
(5, 3): 16711680
```

```
#include <iostream>
#include <vector>
#include <map>
using namespace std;
int m, n, tol;
vector<vector<int>> v;
int dir[8][2] = {{-1, -1}, {-1, 0}, {-1, 1}, {0, 1}, {1, 1}, {1, 0}, {1, -1},
{0, -1}};
bool judge(int i, int j) {
    for (int k = 0; k < 8; k++) {
        int tx = i + dir[k][0];
        int ty = j + dir[k][1];
        if (tx >= 0 && tx < n && ty >= 0 && ty < m && v[i][j] - v[tx][ty] >= 0 -
tol && v[i][j] - v[tx][ty] <= tol) return false;
    }
    return true;
}
int main() {
    int cnt = 0, x = 0, y = 0;
    scanf("%d%d%d", &m, &n, &tol);
    v.resize(n, vector<int>(m));
    map<int, int> mapp;
```

```
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &v[i][j]);
            mapp[v[i][j]]++;
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (mapp[v[i][j]] == 1 && judge(i, j) == true) {
                cnt++;
                x = i + 1;
                y = j + 1;
            }
        }
    }
    if (cnt == 1)
        printf("(%d, %d): %d", y, x, v[x-1][y-1]);
    else if (cnt == 0)
        printf("Not Exist");
    else
        printf("Not Unique");
    return 0;
}
```

1069 微博转发抽奖 (20分)

1069 微博转发抽奖 (20分)

小明 PAT 考了满分，高兴之余决定发起微博转发抽奖活动，从转发的网友中按顺序每隔 N 个人就发出一个红包。请你编写程序帮助他确定中奖名单。

输入格式:

输入第一行给出三个正整数 M（≤ 1000）、N 和 S，分别是转发的总量、小明决定的中奖间隔、以及第一位中奖者的序号（编号从 1 开始）。随后 M 行，顺序给出转发微博的网友的昵称（不超过 20 个字符、不包含空格回车的非空字符串）。

注意：可能有人转发多次，但不能中奖多次。所以如果处于当前中奖位置的网友已经中过奖，则跳过他顺次取下一位。

输出格式:

按照输入的顺序输出中奖名单，每个昵称占一行。如果没有人中奖，则输出 `Keep going...`。

输入样例 1:

```
9 3 2
Imgonnawin!
PickMe
PickMeMeMeee
LookHere
Imgonnawin!
TryAgainAgain
TryAgainAgain
Imgonnawin!
TryAgainAgain
```

输出样例 1:

```
PickMe
Imgonnawin!
TryAgainAgain
```

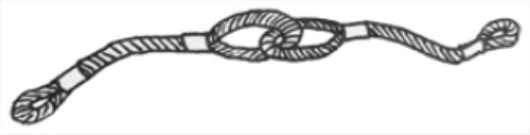
```cpp
#include <iostream>
#include <map>

using namespace std;

int main() {
    int m, n, s;
    scanf("%d%d%d", &m, &n, &s);
    string str;
    map<string, int> mapp;
    bool flag = false;
    for (int i = 1; i <= m; i++) {
        cin >> str;
        if (mapp[str] == 1) s = s + 1;
        if (i == s && mapp[str] == 0) {
            mapp[str] = 1;
            cout << str << endl;
            flag = true;
```

```
            s = s + n;
        }
    }
    if (flag == false) cout << "Keep going...";
    return 0;
}
```

1070 结绳 (25分)

1070 结绳 (25分)

给定一段一段的绳子，你需要把它们串成一条绳。每次串连的时候，是把两段绳子对折，再如下图所示套接在一起。这样得到的绳子又被当成是另一段绳子，可以再次对折去跟另一段绳子串连。每次串连后，原来两段绳子的长度就会减半。



给定 N 段绳子的长度，你需要找出它们能串成的绳子的最大长度。

输入格式:

每个输入包含 1 个测试用例。每个测试用例第 1 行给出正整数 N ($2 \le N \le 10^4$)；第 2 行给出 N 个正整数，即原始绳段的长度，数字间以空格分隔。所有整数都不超过10^4。

输出格式:

在一行中输出能够串成的绳子的最大长度。结果向下取整，即取为不超过最大长度的最近整数。

输入样例:

```
8
10 15 12 3 4 13 1 15
```

输出样例:

```
14
```

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

using namespace std;

int main() {
    int n;
    scanf("%d", &n);
    vector<int> v(n);
    for (int i = 0; i < n; i++)
        scanf("%d", &v[i]);
```

```
    sort(v.begin(), v.end());

    int result = v[0];
    for (int i = 1; i < n; i++)
        result = (result + v[i]) / 2;
    printf("%d", result);
    return 0;
}
```

1071 小赌怡情 (15分)

1071 小赌怡情 (15分)

常言道“小赌怡情”。这是一个很简单的小游戏：首先由计算机给出第一个整数；然后玩家下注赌第二个整数将会比第一个数大还是小；玩家下注 t 个筹码后，计算机给出第二个数。若玩家猜对了，则系统奖励玩家 t 个筹码；否则扣除玩家 t 个筹码。

注意：玩家下注的筹码数不能超过自己帐户上拥有的筹码数。当玩家输光了全部筹码后，游戏就结束。

输入格式:

输入在第一行给出 2 个正整数 T 和 K（≤ 100），分别是系统在初始状态下赠送给玩家的筹码数、以及需要处理的游戏次数。随后 K 行，每行对应一次游戏，顺序给出 4 个数字：

```
n1 b t n2
```

其中 `n1` 和 `n2` 是计算机先后给出的两个[0, 9]内的整数，保证两个数字不相等。 `b` 为 0 表示玩家赌 `小`，为 1 表示玩家赌 `大`。 `t` 表示玩家下注的筹码数，保证在整型范围内。

输出格式:

对每一次游戏，根据下列情况对应输出（其中 `t` 是玩家下注量， `x` 是玩家当前持有的筹码量）：

- 玩家赢，输出 `Win t!  Total = x.`；
- 玩家输，输出 `Lose t.  Total = x.`；
- 玩家下注超过持有的筹码量，输出 `Not enough tokens.  Total = x.`；
- 玩家输光后，输出 `Game Over.` 并结束程序。

输入样例 1:

```
100 4
8 0 100 2
3 1 50 1
5 1 200 6
7 0 200 8
```

输出样例 1:

```
Win 100!  Total = 200.
Lose 50.  Total = 150.
Not enough tokens.  Total = 150.
Not enough tokens.  Total = 150.
```

输入样例 2:

```
100 4
8 0 100 2
3 1 200 1
5 1 200 6
7 0 200 8
```

输出样例 2:

```
Win 100!  Total = 200.
Lose 200.  Total = 0.
Game Over.
```

```cpp
#include <iostream>

using namespace std;

int main()
{
    int t,k;
    cin >> t >> k;
    while(k -- )
    {
        int n1,b,t1,n2;
        cin >> n1 >> b >> t1 >> n2;
        if(t1 > t)
            printf("Not enough tokens.  Total = %d.\n",t);
        else
        {
            if((b && n1 < n2) || (!b && n1 > n2))
            {
                t += t1;
                printf("Win %d!  Total = %d.\n",t1,t);
            }
            else
            {
                t -= t1;
                printf("Lose %d.  Total = %d.\n",t1,t);
```

```
            }
            if(t <= 0)
            {
                cout << "Game Over." <<endl;
                break;
            }

        }

    }

}
```

1072 开学寄语 (20分)

1072 开学寄语 (20分)

下图是上海某校的新学期开学寄语：天将降大任于斯人也，必先删其微博，卸其 QQ，封其电脑，夺其手机，收其 ipad，断其 wifi，使其百无聊赖，然后，净面、理发、整衣，然后思过、读书、锻炼、明智、开悟、精进。而后必成大器也！

本题要求你写个程序帮助这所学校的老师检查所有学生的物品，以助其成大器。

输入格式:

输入第一行给出两个正整数 N（≤ 1000）和 M（≤ 6），分别是学生人数和需要被查缴的物品种类数。第二行给出 M 个需要被查缴的物品编号，其中编号为 4 位数字。随后 N 行，每行给出一位学生的姓名缩写（由 1-4 个大写英文字母组成）、个人物品数量 K（0 ≤ K ≤ 10）、以及 K 个物品的编号。

输出格式:

顺次检查每个学生携带的物品，如果有需要被查缴的物品存在，则按以下格式输出该生的信息和其需要被查缴的物品的信息（注意行末不得有多余空格）：

```
姓名缩写：物品编号1 物品编号2 ……
```

最后一行输出存在问题的学生的总人数和被查缴物品的总数。

```
#include <iostream>
#include <algorithm>
using namespace std;
```

```
bool forbid[10000];
int main() {
    int n, m, temp, k, snum = 0, fnum = 0;
    scanf("%d %d", &n, &m);
    for (int i = 0; i < m; i++) {
        scanf("%d", &temp);
        forbid[temp] = true;
    }
    for (int i = 0; i < n; i++) {
        char name[10];
        bool flag = false;
        scanf("%s %d", name, &k);
        for (int j = 0; j < k; j++) {
            scanf("%d", &temp);
            if (forbid[temp]) {
                if (!flag) {
                    printf("%s:", name);
                    flag = true;
                }
                printf(" %04d", temp);
                fnum++;
            }
        }
        if (flag) {
            printf("\n");
            snum++;
        }
    }
    printf("%d %d\n", snum, fnum);
    return 0;
}
```

1073 多选题常见计分法 (20分)

1073 多选题常见计分法 (20分)

批改多选题是比较麻烦的事情，有很多不同的计分方法。有一种最常见的计分方法是：如果考生选择了部分正确选项，并且没有选择任何错误选项，则得到 50% 分数；如果考生选择了任何一个错误的选项，则不能得分。本题就请你写个程序帮助老师批改多选题，并且指出哪道题的哪个选项错的人最多。

输入格式:

输入在第一行给出两个正整数 N（≤1000）和 M（≤100），分别是学生人数和多选题的个数。随后 M 行，每行顺次给出一道题的满分值（不超过 5 的正整数）、选项个数（不少于 2 且不超过 5 的正整数）、正确选项个数（不超过选项个数的正整数）、所有正确选项。注意每题的选项从小写英文字母 a 开始顺次排列。各项间以 1 个空格分隔。最后 N 行，每行给出一个学生的答题情况，其每题答案格式为 `(选中的选项个数 选项1 ……)`，按题目顺序给出。注意：题目保证学生的答题情况是合法的，即不存在选中的选项数超过实际选项数的情况。

输出格式:

按照输入的顺序给出每个学生的得分，每个分数占一行，输出小数点后 1 位。最后输出错得最多的题目选项的信息，格式为：`错误次数 题目编号（题目按照输入的顺序从1开始编号）-选项号`。如果有并列，则每行一个选项，按题目编号递增顺序输出；再并列则按选项号递增顺序输出。行首尾不得有多余空格。如果所有题目都没有人错，则在最后一行输出 `Too simple`。

输入样例 1:

```
3 4
3 4 2 a c
2 5 1 b
5 3 2 b c
1 5 4 a b d e
(2 a c) (3 b d e) (2 a c) (3 a b e)
(2 a c) (1 b) (2 a b) (4 a b d e)
(2 b d) (1 e) (1 c) (4 a b c d)
```

输出样例 1:

```
3.5
6.0
2.5
2 2-e
2 3-a
2 3-b
```

输入样例 2:

```
2 2
3 4 2 a c
2 5 1 b
(2 a c) (1 b)
(2 a c) (1 b)
```

输出样例 2:

```
5.0
5.0
Too simple
```

1074 宇宙无敌加法器 (20分)

1074 宇宙无敌加法器 (20分)

地球人习惯使用十进制数，并且默认一个数字的每一位都是十进制的。而在 PAT 星人开挂的世界里，每个数字的每一位都是不同进制的，这种神奇的数字称为“PAT数”。每个 PAT 星人都必须熟记各位数字的进制表，例如“……0527”就表示最低位是 7 进制数、第 2 位是 2 进制数、第 3 位是 5 进制数、第 4 位是 10 进制数，等等。每一位的进制 d 或者是 0（表示十进制）、或者是 [2，9] 区间内的整数。理论上这个进制表应该包含无穷多位数字，但从实际应用出发，PAT 星人通常只需要记住前 20 位就够用了，以后各位默认为 10 进制。

在这样的数字系统中，即使是简单的加法运算也变得不简单。例如对应进制表“0527”，该如何计算“6203 + 415”呢？我们得首先计算最低位：3 + 5 = 8；因为最低位是 7 进制的，所以我们得到 1 和 1 个进位。第 2 位是：0 + 1 + 1（进位）= 2；因为此位是 2 进制的，所以我们得到 0 和 1 个进位。第 3 位是：2 + 4 + 1（进位）= 7；因为此位是 5 进制的，所以我们得到 2 和 1 个进位。第 4 位是：6 + 1（进位）= 7；因为此位是 10 进制的，所以我们就得到 7。最后我们得到：6203 + 415 = 7201。

输入格式:

输入首先在第一行给出一个 N 位的进制表（$0 < N \leq 20$），以回车结束。随后两行，每行给出一个不超过 N 位的非负的 PAT 数。

输出格式:

在一行中输出两个 PAT 数之和。

输入样例:

```
30527
06203
415
```

输出样例:

```
7201
```

```cpp
#include <iostream>
using namespace std;
int main() {
    string s, s1, s2, ans;
    int carry = 0, flag = 0;
    cin >> s >> s1 >> s2;
    ans = s;
    string ss1(s.length() - s1.length(), '0');
    s1 = ss1 + s1;
    string ss2(s.length() - s2.length(), '0');
    s2 = ss2 + s2;
    for(int i = s.length() - 1; i >= 0; i--) {
        int mod = s[i] == '0' ? 10 : (s[i] - '0');
        ans[i] = (s1[i] - '0' + s2[i] - '0' + carry) % mod + '0';
        carry = (s1[i] - '0' + s2[i] - '0' + carry) / mod;
    }
    if (carry != 0) ans = '1' + ans;
    for(int i = 0; i < ans.size(); i++) {
        if (ans[i] != '0' || flag == 1) {
            flag = 1;
```

```
            cout << ans[i];
        }
    }
    if (flag == 0) cout << 0;
    return 0;
}
```

1075 链表元素分类 (25分)

返回

1075 链表元素分类 (25分)

给定一个单链表，请编写程序将链表元素进行分类排列，使得所有负值元素都排在非负值元素的前面，而 [0, K] 区间内的元素都排在大于 K 的元素前面。但每一类内部元素的顺序是不能改变的。例如：给定链表为 18→7→-4→0→5→-6→10→11→-2，K 为 10，则输出应该为 -4→-6→-2→7→0→5→10→18→11。

输入格式:

每个输入包含一个测试用例。每个测试用例第 1 行给出：第 1 个结点的地址；结点总个数，即正整数 N ($\leq 10^5$)；以及正整数K ($\leq 10^3$)。结点的地址是 5 位非负整数，NULL 地址用 -1 表示。

接下来有 N 行，每行格式为：

```
Address Data Next
```

其中 `Address` 是结点地址；`Data` 是该结点保存的数据，为 $[-10^5, 10^5]$ 区间内的整数；`Next` 是下一结点的地址。题目保证给出的链表不为空。

输出格式:

对每个测试用例，按链表从头到尾的顺序输出重排后的结果链表，其上每个结点占一行，格式与输入相同。

1076 Wifi密码 (15分)

返回

1076 Wifi密码 (15分)

下面是微博上流传的一张照片：“各位亲爱的同学们，鉴于大家有时需要使用 wifi，又怕耽误亲们的学习，现将 wifi 密码设置为下列数学题答案：A-1；B-2；C-3；D-4；请同学们自己作答，每两日一换。谢谢合作！！~” —— 老师们为了促进学生学习也是拼了..... 本题就要求你写程序把一系列题目的答案按照卷子上给出的对应关系翻译成 wifi 的密码。这里简单假设每道选择题都有 4 个选项，有且只有 1 个正确答案。

输入格式:

输入第一行给出一个正整数 N（≤ 100），随后 N 行，每行按照 `编号-答案` 的格式给出一道题的 4 个选项，`T` 表示正确选项，`F` 表示错误选项。选项间用空格分隔。

输出格式:

在一行中输出 wifi 密码。

输入样例:

```
8
A-T B-F C-F D-F
C-T B-F A-F D-F
A-F D-F C-F B-T
B-T A-F C-F D-F
B-F D-T A-F C-F
A-T C-F B-F D-F
D-T B-F C-F A-F
C-T A-F B-F D-F
```

输出样例:

```
13224143
```

```cpp
#include <iostream>

using namespace std;

int main() {
    string s;
    while (cin >> s)
        if(s.size() == 3 && s[2] == 'T') cout << s[0]-'A'+1;
    return 0;
}
```

1077 互评成绩计算 (20分)

1077 互评成绩计算 (20分)

在浙大的计算机专业课中，经常有互评分组报告这个环节。一个组上台介绍自己的工作，其他组在台下为其表现评分。最后这个组的互评成绩是这样计算的：所有其他组的评分中，去掉一个最高分和一个最低分，剩下的分数取平均分记为 G_1；老师给这个组的评分记为 G_2。该组得分为 $(G_1+G_2)/2$，最后结果四舍五入后保留整数分。本题就要求你写个程序帮助老师计算每个组的互评成绩。

输入格式:

输入第一行给出两个正整数 N（>3）和 M，分别是分组数和满分，均不超过 100。随后 N 行，每行给出该组得到的 N 个分数（均保证为整型范围内的整数），其中第 1 个是老师给出的评分，后面 $N-1$ 个是其他组给的评分。合法的输入应该是 $[0,M]$ 区间内的整数，若不在合法区间内，则该分数须被忽略。题目保证老师的评分都是合法的，并且每个组至少会有 3 个来自同学的合法评分。

输出格式:

为每个组输出其最终得分。每个得分占一行。

输入样例:

```
6 50
42 49 49 35 38 41
36 51 50 28 -1 30
40 36 41 33 47 49
30 250 -25 27 45 31
48 0 0 50 50 1234
43 41 36 29 42 29
```

输出样例:

```
42
33
41
31
37
39
```

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

using namespace std;

int main()
{
    int n,m;
    cin >> n >> m;
    int num = n;
    while(n --)
    {
        vector<int> score;
        int res = 0;
```

```
        for(int i = 0;i < num;i ++)
        {
            int tmp;
            cin >> tmp;
            if(tmp <= m && tmp >= 0)
                score.push_back(tmp);
        }
        sort(score.begin() + 1,score.end());
        if(score.size() == 4) res += score[2];
        else for(int i = 2;i < score.size() - 1; i++) res += score[i];

        cout << int((res * 1.0 / (score.size() - 3) + score[0]) / 2 + 0.5)
<<endl;
    }
}
```

1078 字符串压缩与解压 (20分)

返回

1078 字符串压缩与解压 (20分)

文本压缩有很多种方法，这里我们只考虑最简单的一种：把由相同字符组成的一个连续的片段用这个字符和片段中含有这个字符的个数来表示。例如 `ccccc` 就用 `5c` 来表示。如果字符没有重复，就原样输出。例如 `aba` 压缩后仍然是 `aba`。

解压方法就是反过来，把形如 `5c` 这样的表示恢复为 `ccccc`。

本题需要你根据压缩或解压的要求，对给定字符串进行处理。这里我们简单地假设原始字符串是完全由英文字母和空格组成的非空字符串。

输入格式:

输入第一行给出一个字符，如果是 `C` 就表示下面的字符串需要被压缩；如果是 `D` 就表示下面的字符串需要被解压。第二行给出需要被压缩或解压的不超过 1000 个字符的字符串，以回车结尾。题目保证字符重复个数在整型范围内，且输出文件不超过 1MB。

输出格式:

根据要求压缩或解压字符串，并在一行中输出结果。

输入样例 1:

```
C
TTTTThhiiiis isssss a   tesssst CAaaa as
```

输出样例 1:

```
5T2h4is i5s a3 te4st CA3a as
```

```
#include <iostream>
using namespace std;
int main() {
    char t;
```

```
    cin >> t;
    getchar();
    string s, num;
    getline(cin, s);
    int cnt = 1;
    if (t == 'D') {
        for (int i = 0; i < s.length(); i++) {
            if (s[i] >= '0' && s[i] <= '9') {
                num += s[i];
            } else {
                if (num.length() > 0) cnt = stoi(num);
                while(cnt--) cout << s[i];
                cnt = 1;
                num = "";
            }
        }
    } else if (s.length() != 0) {
        char pre = s[0];
        for (int i = 1; i < s.length(); i++) {
            if (s[i] == pre) {
                cnt++;
            } else {
                if (cnt >= 2) cout << cnt;
                cout << pre;
                cnt = 1;
                pre = s[i];
            }
        }
        if (cnt >= 2) cout << cnt;
        cout << pre;
    }
    return 0;
}
```

1079 延迟的回文数 (20分)

1079 延迟的回文数 (20分)

给定一个 $k+1$ 位的正整数 N，写成 $a_k \cdots a_1 a_0$ 的形式，其中对所有 i 有 $0 \le a_i < 10$ 且 $a_k > 0$。N 被称为一个**回文数**，当且仅当对所有 i 有 $a_i = a_{k-i}$。零也被定义为一个回文数。

非回文数也可以通过一系列操作变出回文数。首先将该数字逆转，再将逆转数与该数相加，如果和还不是一个回文数，就重复这个逆转再相加的操作，直到一个回文数出现。如果一个非回文数可以变出回文数，就称这个数为**延迟的回文数**。（定义翻译自 https://en.wikipedia.org/wiki/Palindromic_number ）

给定任意一个正整数，本题要求你找到其变出的那个回文数。

输入格式:

输入在一行中给出一个不超过1000位的正整数。

输出格式:

对给定的整数，一行一行输出其变出回文数的过程。每行格式如下

```
A + B = C
```

其中 `A` 是原始的数字，`B` 是 `A` 的逆转数，`C` 是它们的和。`A` 从输入的整数开始。重复操作直到 `C` 在 10 步以内变成回文数，这时在一行中输出 `C is a palindromic number.`；或者如果 10 步都没能得到回文数，最后就在一行中输出 `Not found in 10 iterations.`。

输入样例 1:

```
97152
```

输出样例 1:

```
97152 + 25179 = 122331
122331 + 133221 = 255552
255552 is a palindromic number.
```

```cpp
#include <iostream>
#include <algorithm>
using namespace std;
string rev(string s) {
    reverse(s.begin(), s.end());
    return s;
}
string add(string s1, string s2) {
    string s = s1;
    int carry = 0;
    for (int i = s1.size() - 1; i >= 0; i--) {
        s[i] = (s1[i] - '0' + s2[i] - '0' + carry) % 10 + '0';
        carry = (s1[i] - '0' + s2[i] - '0' + carry) / 10;
    }
    if (carry > 0) s = "1" + s;
    return s;
}
int main() {
```

```
    string s, sum;
    int n = 10;
    cin >> s;
    if (s == rev(s)) {
        cout << s << " is a palindromic number.\n";
        return 0;
    }
    while (n--) {
        sum = add(s, rev(s));
        cout << s << " + " << rev(s) << " = " << sum << endl;
        if (sum == rev(sum)) {
            cout << sum << " is a palindromic number.\n";
            return 0;
        }
        s = sum;
    }
    cout << "Not found in 10 iterations.\n";
    return 0;
}
```

1080 MOOC期终成绩 (25分)

1080 MOOC期终成绩 (25分)

对于在中国大学MOOC（http://www.icourse163.org/ ）学习“数据结构”课程的学生，想要获得一张合格证书，必须首先获得不少于200分的在线编程作业分，然后总评获得不少于60分（满分100）。总评成绩的计算公式为 $G = (G_{mid-term} \times 40\% + G_{final} \times 60\%)$，如果 $G_{mid-term} > G_{final}$；否则总评 G 就是 G_{final}。这里 $G_{mid-term}$ 和 G_{final} 分别为学生的期中和期末成绩。

现在的问题是，每次考试都产生一张独立的成绩单。本题就请你编写程序，把不同的成绩单合为一张。

输入格式:

输入在第一行给出3个整数，分别是 P（做了在线编程作业的学生数）、M（参加了期中考试的学生数）、N（参加了期末考试的学生数）。每个数都不超过10000。

接下来有三块输入。第一块包含 P 个在线编程成绩 G_p；第二块包含 M 个期中考试成绩 $G_{mid-term}$；第三块包含 N 个期末考试成绩 G_{final}。每个成绩占一行，格式为：`学生学号 分数`。其中`学生学号`为不超过20个字符的英文字母和数字；`分数`是非负整数（编程总分最高为900分，期中和期末的最高分为100分）。

输出格式:

打印出获得合格证书的学生名单。每个学生占一行，格式为：

`学生学号` G_p $G_{mid-term}$ G_{final} G

如果有的成绩不存在（例如某人没参加期中考试），则在相应的位置输出“-1”。输出顺序为按照总评分数（四舍五入精确到整数）递减。若有并列，则按学号递增。题目保证学号没有重复，且至少存在1个合格的学生。

输入样例：

```
6 6 7
01234 880
a1903 199
ydjh2 200
wehu8 300
dx86w 220
missing 400
ydhfu77 99
wehu8 55
ydjh2 98
dx86w 88
a1903 86
01234 39
ydhfu77 88
a1903 66
01234 58
wehu8 84
ydjh2 82
missing 99
dx86w 81
```

输出样例：

```
missing 400 -1 99 99
ydjh2 200 98 82 88
dx86w 220 88 81 84
wehu8 300 55 84 84
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
#include <map>
using namespace std;
struct node {
    string name;
    int gp, gm, gf, g;
};
bool cmp(node a, node b) {
    return a.g != b.g ? a.g > b.g : a.name < b.name;
}
map<string, int> idx;
int main() {
    int p, m, n, score, cnt = 1;
    cin >> p >> m >> n;
    vector<node> v, ans;
    string s;
    for (int i = 0; i < p; i++) {
        cin >> s >> score;
        if (score >= 200) {
            v.push_back(node{s, score, -1, -1, 0});
            idx[s] = cnt++;
        }
    }
    for (int i = 0; i < m; i++) {
```

```
        cin >> s >> score;
        if (idx[s] != 0) v[idx[s] - 1].gm = score;
    }
    for (int i = 0; i < n; i++) {
        cin >> s >> score;
        if (idx[s] != 0) {
            int temp = idx[s] - 1;
            v[temp].gf = v[temp].g = score;
            if (v[temp].gm > v[temp].gf) v[temp].g = int(v[temp].gm * 0.4 +
v[temp].gf * 0.6 + 0.5);
        }
    }
    for (int i = 0; i < v.size(); i++)
        if (v[i].g >= 60) ans.push_back(v[i]);
    sort(ans.begin(), ans.end(), cmp);
    for (int i = 0; i < ans.size(); i++)
        printf("%s %d %d %d %d\n", ans[i].name.c_str(), ans[i].gp, ans[i].gm,
ans[i].gf, ans[i].g);
    return 0;
}
```