1001. 害死人不偿命的(3n+1)猜想(15) [模拟]

卡拉兹(Callatz)猜想：

对任何一个自然数n，如果它是偶数，那么把它砍掉一半；如果它是奇数，那么把(3n+1)砍掉一半。这样一直反复砍下去，最后一定在某一步得到n=1。卡拉兹在1950年的世界数学家大会上公布了这个猜想，传说当时耶鲁大学师生齐动员，拼命想证明这个貌似很傻很天真的命题，结果闹得学生们无心学业，一心只证(3n+1)，以至于有人说这是一个阴谋，卡拉兹是在蓄意延缓美国数学界教学与科研的进展……

我们今天的题目不是证明卡拉兹猜想，而是对给定的任一不超过1000的正整数n，简单地数一下，需要多少步（砍几下）才能得到n=1？

输入格式：

每个测试输入包含1个测试用例，即给出自然数n的值。

输出格式：

输出从n计算到1需要的步数。

输入样例：

3

输出样例：

5

分析：count从0开始统计需要的步数，(n % 2 != 0) 表示n为奇数，当n为奇数，就令n = 3*n+1；之后将其砍掉一半，步数count+1，直到n == 1为止，最后输出count～

```
#include <iostream>
using namespace std;
int main() {
    int n, count = 0;
    cin >> n;
    while (n != 1) {
        if (n % 2 != 0) n = 3 * n + 1;
        n = n / 2;
        count++;
    }
    cout << count;
    return 0;
}
```

1002. 写出这个数 (20) [字符串处理]

读入一个自然数n，计算其各位数字之和，用汉语拼音写出和的每一位数字。

输入格式：每个测试输入包含1个测试用例，即给出自然数n的值。这里保证n小于10100。

输出格式：

在一行内输出n的各位数字之和的每一位，拼音数字间有1 空格，但一行中最后一个拼音数字后没有空格。

输入样例：

123456789098765432112345678 9

输出样例：

yi san wu

分析：用string接收输入，string的每一位数字累加到sum里面，再将sum转化为string类型的num，对num的每一位输出对应中文拼音～

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    cin >> s;
    int sum = 0;
    string str[10] =  {"ling", "yi", "er", "san", "si", "wu", "liu", "qi",
"ba", "jiu"};
    for (int i = 0; i < s.length(); i++)
        sum += (s[i] - '0');
    string num = to_string(sum);
    for (int i = 0; i < num.length(); i++) {
        if (i != 0) cout << " ";
        cout << str[num[i] - '0'];
    }
    return 0;
}
```

1003. 我要通过！(20) [数学题]

“答案正确”是自动判题系统给出的最令人欢喜的回复。本题属于PAT的“答案正确”大派送 —— 只要读入的字符串满足下列条件，系统就输出“答案正确”，否则输出“答案错误”。

得到“答案正确”的条件是：

1. 字符串中必须仅有P, A, T这三种字符，不可以包含其它字符；
2. 任意形如 xPATx 的字符串都可以获得“答案正确”，其中 x 或者是空字符串，或者是仅由字母 A 组

成的字符串；
3. 如果 aPbTc 是正确的，那么 aPbATca 也是正确的，其中 a, b, c 均或者是空字符串，或者是仅由字母 A 组成的字符串。

现在就请你为PAT写一个自动裁判程序，判定哪些字符串是可以获得“答案正确”的。

输入格式：

每个测试输入包含1个测试用例。第1行给出一个自然数n (<10)，是需要检测的字符串个数。接下来每个字符串占一行，字符串长度不超过100，且不包含空格。

输出格式：

每个字符串的检测结果占一行，如果该字符串可以获得“答案正确”，则输出YES，否则输出NO。

输入样例：

8

PAT

PAAT

AAPATAA

AAPAATAAAA

xPATx

PT

Whatever

APAAATAA

输出样例：

YES

YES

YES

YES

NO

NO

NO

NO

分析：

任意形如 xPATx 的字符串都可以获得“答案正确”，其中 x 或者是空字符串，或者是仅由字母 A 组成的字符串；

那么正确的有这些：

PAT

APATA

AAPATAA

AAAPATAAA

...不说了，就是中间一个A左右加上等量的A（不加也行）都是正确的。

如果 aPbTc 是正确的，那么 aPbATca 也是正确的，其中 a, b, c 均或者是空字符串，或者是仅由字母 A 组成的字符串。

拿上面的那几个正确的举例子，那么正确的有这些：

PAT —— 对于 aPbTc 来说ac是空，b是A。所以 PAAT 是正确的。同理PAAAAAT中间加多少个A都是正确哒～

APATA —— 对于aPbTc来说，abc都是A。所以 APAATAA 是正确的。再类推一下，那么 APAAATAAA 是正确的。

AAPATAA —— 对于aPbTc来说，a和c是AA，b是A。所以AAPAATAAAA是正确的，再类推一下，AAPAAATAAAAAA 是正确的～～

所以说规律就是，可以在P和T中间加A并且在T后面加A，要求必须是，中间加上一个A，末尾就得加上几倍的(P前面A的那个字符串)。换句话说就是，中间的A的个数如果是3，那么末尾的A的个数就得是开头A的个数的3倍。很巧，当中间A为一个的时候，末尾和开头A的个数必须相等正好是第二条的要求～

所以一句话总结字符串的要求：只能有一个P一个T，中间末尾和开头可以随便插入A。但是必须满足开头的A的个数 * 中间的A的个数 = 结尾的A的个数，而且P和T中间不能没有A～

```
#include <iostream>
#include <map>
using namespace std;
int main() {
    int n, p = 0, t = 0;
    string s;
    cin >> n;
    for(int i = 0; i < n; i++) {
        cin >> s;
        map<char, int> m;
        for(int j = 0; j < s.size(); j++) {
            m[s[j]]++;
            if (s[j] == 'P') p = j;
            if (s[j] == 'T') t = j;
        }
        if (m['P'] == 1 && m['A'] != 0 && m['T'] == 1 && m.size() == 3 && t-p
!= 1 && p * (t-p-1) == s.length()-t-1)
            printf("YES\n");
```

```
        else
            printf("NO\n");
    }
    return 0;
}
```

1004. 成绩排名 (20) [查找元素]

读入n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

分析：maxname和maxnum保存成绩最高的学生的姓名和学号，minname和minnum保存成绩最低的学生的姓名和学号，max和min保存当前的最高分和最低分，因为成绩区间为0-100，所以初始化时先令max = -1，min = 101。遍历所有数据，如果当前学生数据的分数比最大值大，那么更新max的值，并将他的姓名学号保存在maxname和maxnum中；如果当前学生数据的分数比最小值小，那么更新min的值，并将他的姓名学号保存在minname和minnum中。最后输出maxname、maxnum、minname和minnum～

```cpp
#include <iostream>
using namespace std;
int main() {
    int n, max = -1, min = 101, score;
    cin >> n;
    string maxname, minname, maxnum, minnum, name, num;
    for (int i = 0; i < n; i++) {
        cin >> name >> num >> score;
        if (max < score) {
            max = score;
            maxname = name;
            maxnum = num;
        }
        if (min > score) {
            min = score;
            minname = name;
            minnum = num;
        }
    }
    cout << maxname << " " << maxnum << endl << minname << " " << minnum;
    return 0;
}
```

1005. 继续(3n+1)猜想 (25) [Hash散列]

卡拉兹(Callatz)猜想已经在1001中给出了描述。在这个题目里，情况稍微有些复杂。

当我们验证卡拉兹猜想的时候，为了避免重复计算，可以记录下递推过程中遇到的每一个数。例如对n=3进行验证的时候，我们需要计算3、5、8、4、2、1，则当我们对n=5、8、4、2进行验证的时候，就可以直接判定卡拉兹猜想的真伪，而不需要重复计算，因为这4个数已经在验证3的时候遇到过了，我们称5、8、4、2是被3“覆盖”的数。我们称一个数列中的某个数n为“关键数”，如果n不能被数列中的其他数字所覆盖。

现在给定一系列待验证的数字，我们只需要验证其中的几个关键数，就可以不必再重复验证余下的数字。你的任务就是找出这些关键数字，并按从大到小的顺序输出它们。

输入格式：

每个测试输入包含1个测试用例，第1行给出一个正整数K(<100)，第2行给出K个互不相同的待验证的正整数n(1<n<=100)的值，数字间用空格隔开。

输出格式：

每个测试用例的输出占一行，按从大到小的顺序输出关键数字。数字间用1个空格隔开，但一行中最后一个数字后没有空格。

输入样例：

6

3 5 6 7 8 11

输出样例：

7 6

分析：对每一个输入的数字n进行验证，把验证过的数字对应的arr标记为1，然后对这些输入的数字从大到小排序，输出所有arr=0的数字即为关键数字～

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int arr[10000];
bool cmp(int a, int b) {return a > b;}
int main() {
    int k, n, flag = 0;
    cin >> k;
    vector<int> v(k);
    for (int i = 0; i < k; i++) {
        cin >> n;
        v[i] = n;
        while (n != 1) {
            if (n % 2 != 0) n = 3 * n + 1;
            n = n / 2;
            if (arr[n] == 1) break;
            arr[n] = 1;
        }
    }
    sort(v.begin(), v.end(), cmp);
    for (int i = 0; i < v.size(); i++) {
        if (arr[v[i]] == 0) {
            if (flag == 1) cout << " ";
            cout << v[i];
            flag = 1;
        }
    }
    return 0;
}
```

1006. 换个格式输出整数 (15) [字符串处理]

让我们用字母B来表示“百”、字母S表示“十”，用“12...n”来表示个位数字n（<10），换个格式来输出任一个不超过3位的正整数。例如234应该被输出为BBSSS1234，因为它有2个“百”、3个“十”、以及个位的4。

输入格式：

每个测试输入包含1个测试用例，给出正整数n（<1000）。

输出格式：

每个测试用例的输出占一行，用规定的格式输出n。

输入样例1：

234

输出样例1：

BBSSS1234

输入样例2：

23

输出样例2：

SS123

分析：因为n小于1000，所以数字不会超过百位～输入数据首先保存在a中，然后将a的每一个数字保存在int b[3]中，然后将b[2]、b[1]、b[0]中存储的数字看作输出次数依次输出B、S和12..b[0]～

```
#include <iostream>
using namespace std;
int main() {
    int a, i = 0;
    cin >> a;
    int b[3] = {0};
    while (a != 0) {
        b[i++] = a % 10;
        a = a / 10;
    }
    for (int k = 0; k < b[2]; k++)
        cout << "B";
    for (int k = 0; k < b[1]; k++)
        cout << "S";
    for (int k = 0; k < b[0]; k++)
        cout << k + 1;
    return 0;
}
```

1007. 素数对猜想 (20) [素数]

让我们定义 dn 为：dn = pn+1 – pn，其中 pi 是第i个素数。显然有 d1=1 且对于n>1有 dn 是偶数。“素数对猜想”认为“存在无穷多对相邻且差为2的素数”。

现给定任意正整数N (< 105)，请计算不超过N的满足猜想的素数对的个数。

输入格式：

每个测试输入包含1个测试用例，给出正整数N。

输出格式：

每个测试用例的输出占一行，不超过N的满足猜想的素数对的个数。

输入样例：

20

输出样例：

4

分析：判断素数的函数isprime这样写：对于数字a，i从2到根号a，如果a能够被其中一个i整除，说明i不是素数，return false，否则说明a是素数return true；对于输入数据N，for循环中的i从5到N依次判断i-2和i是否是素数，如果都是素数，则统计个数的cnt++，最后输出cnt即可～

```
#include <iostream>
using namespace std;
bool isprime(int a) {
    for (int i = 2; i * i <= a; i++)
        if (a % i == 0) return false;
    return true;
}
int main() {
    int N, cnt = 0;
    cin >> N;
    for (int i = 5; i <= N; i++)
        if (isprime(i-2) && isprime(i)) cnt++;
    cout << cnt;
    return 0;
}
```

1008. 数组元素循环右移问题 (20) [模拟]

一个数组A中存有N（N>0）个整数，在不允许使用另外数组的前提下，将每个整数循环向右移M（M>=0）个位置，即将A中的数据由（A0 A1……AN-1）变换为（AN-M …… AN-1 A0 A1……AN-M-1）（最后M个数循环移至最前面的M个位置）。如果需要考虑程序移动数据的次数尽量少，要如何设计移动的方法?

输入格式：

每个输入包含一个测试用例，第1行输入N (1<=N<=100)、M（M>=0）；第2行输入N个整数，之间用空格分隔。

输出格式：

在一行中输出循环右移M位以后的整数序列，之间用空格分隔，序列结尾不能有多余空格。

输入样例：

6 2

1 2 3 4 5 6

输出样例：

5 6 1 2 3 4

分析：数组长度为n，要想把数组循环右移m位，只需要先将整个数组a倒置，再将数组前m位倒置，最后将数组后n-m位倒置即可完成循环右移m位～reverse函数可以实现将一个数组或者vector中元素倒置，这个函数在algorithm头文件中～（如果m大于n，那么循环右移m位相当于循环右移m%n位，因为那些n倍数位的移动是多余的，所以在使用m之前，先将m = m%n）

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main() {
    int n, m;
    cin >> n >> m;
    vector<int> a(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];
    m %= n;
    if (m != 0) {
        reverse(begin(a), begin(a) + n);
        reverse(begin(a), begin(a) + m);
        reverse(begin(a) + m, begin(a) + n);
    }
    for (int i = 0; i < n - 1; i++)
        cout << a[i] << " ";
    cout << a[n - 1];
    return 0;
}
```

1009. 说反话 (20) [字符串处理]

给定一句英语，要求你编写程序，将句中所有单词的顺序颠倒输出。

输入格式：

测试输入包含一个测试用例，在一行内给出总长度不超过80的字符串。字符串由若干单词和若干空格组成，其中单词是由英文字母（大小写有区分）组成的字符串，单词之间用1个空格分开，输入保证句子末尾没有多余的空格。

输出格式：

每个测试用例的输出占一行，输出倒序后的句子。

输入样例：

Hello World Here I Come

输出样例：

Come I Here World Hello

分析：将输入的每个单词s都分别v.push(s)压入栈中，再输出栈顶v.top()，然后将栈顶元素弹出v.pop()，直到栈空为止～

```
#include <iostream>
#include <stack>
using namespace std;
int main() {
    stack<string> v;
    string s;
    while(cin >> s) v.push(s);
    cout << v.top();
    v.pop();
    while(!v.empty()) {
        cout << " " << v.top();
        v.pop();
    }
    return 0;
}
```

1010. 一元多项式求导 (25) [模拟]

设计函数求一元多项式的导数。（注：xn（n为整数）的一阶导数为n*xn-1。）

输入格式：

以指数递降方式输入多项式非零项系数和指数（绝对值均为不超过1000的整数）。数字间以空格分隔。

输出格式：

以与输入相同的格式输出导数多项式非零项的系数和指数。数字间以空格分隔，但结尾不能有多余空格。注意“零多项式”的指数和系数都是0，但是表示为“0 0”。

输入样例：

3 4 -5 2 6 1 -2 0

输出样例：

12 3 -10 1 6 0

分析：1.flag用来判断是否已经有过输出～

2.当b!=0时，因为给出的是所有非零项系数，所以必定会有输出，先判断flag是否为1，如果为1表示已经有过输出，那么在前面要先输出一个空格

3.输出 a * b 和 b – 1，然后将flag标记为1表示已经有过输出

4.最后判断当没有输出并且b==0的时候，输出“0 0”

```
#include <iostream>
using namespace std;
int main() {
    int a, b, flag = 0;
    while (cin >> a >> b) {
        if (b != 0) {
            if (flag == 1) cout << " ";
            cout << a * b << " " << b - 1;
            flag = 1;
        }
    }
    if (flag == 0) cout << "0 0";
    return 0;
}
```

1011. A+B和C (15) [模拟]

题目描述：

给定区间[-2^31, 2^31]内的3个整数A、B和C，请判断A+B是否大于C。

输入格式：

输入第1行给出正整数T(<=10)，是测试用例的个数。随后给出T组测试用例，每组占一行，顺序给出A、B和C。整数间以空格分隔。 输出格式： 对每组测试用例，在一行中输出“Case #X: true”如果A+B>C，否则输出“Case #X: false”，其中X是测试用例的编号（从1开始）。

输入样例：

4

1 2 3

2 3 4

2147483647 0 2147483646

0 -2147483648 -2147483647

输出样例：

Case #1: false

Case #2: true

Case #3: true

Case #4: false

分析：使用long long int存储a、b和c，当a + b > c的时候输出true，否则输出false～

```
#include <iostream>
using namespace std;
int main() {
    int n;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        long long int a, b, c;
        scanf("%lld%lld%lld", &a, &b, &c);
        printf("Case #%d: %s\n", i + 1, a + b > c ? "true" : "false");
    }
    return 0;
}
```

1012. 数字分类 (20) [模拟]

给定一系列正整数，请按要求对数字进行分类，并输出以下5个数字：

A1 = 能被5整除的数字中所有偶数的和；

A2 = 将被5除后余1的数字按给出顺序进行交错求和，即计算n1-n2+n3-n4...；

A3 = 被5除后余2的数字的个数；

A4 = 被5除后余3的数字的平均数，精确到小数点后1位；

A5 = 被5除后余4的数字中最大数字。

输入格式：

每个输入包含1个测试用例。每个测试用例先给出一个不超过1000的正整数N，

随后给出N个不超过1000的待分类的正整数。数字间以空格分隔。

输出格式：

对给定的N个正整数，按题目要求计算A1~A5并在一行中顺序输出。

数字间以空格分隔，但行末不得有多余空格。

若其中某一类数字不存在，则在相应位置输出“N”。

输入样例1：

13 1 2 3 4 5 6 7 8 9 10 20 16 18

输出样例1：

30 11 2 9.7 9

输入样例2：

8 1 2 4 5 6 7 9 16

输出样例2：

N 11 2 N 9

分析：将每一个数字按照取余后的结果i保存在v[i]数组中，然后对v[i]中的每一个元素按照不同i分类计算出A1 A2...A5～

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, num, A1 = 0, A2 = 0, A5 = 0;
    double A4 = 0.0;
    cin >> n;
    vector<int> v[5];
    for (int i = 0; i < n; i++) {
        cin >> num;
        v[num%5].push_back(num);
    }
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < v[i].size(); j++) {
            if (i == 0 && v[i][j] % 2 == 0) A1 += v[i][j];
            if (i == 1 && j % 2 == 0) A2 += v[i][j];
            if (i == 1 && j % 2 == 1) A2 -= v[i][j];
            if (i == 3) A4 += v[i][j];
            if (i == 4 && v[i][j] > A5) A5 = v[i][j];
        }
    }
    for (int i = 0; i < 5; i++) {
        if (i != 0) printf(" ");
        if (i == 0 && A1 == 0 || i != 0 && v[i].size() == 0) {
            printf("N"); continue;
        }
        if (i == 0) printf("%d", A1);
        if (i == 1) printf("%d", A2);
        if (i == 2) printf("%d", v[2].size());
        if (i == 3) printf("%.1f", A4 / v[3].size());
        if (i == 4) printf("%d", A5);
    }
    return 0;
}
```

1013. 数素数 (20) [素数]

令Pi表示第i个素数。现任给两个正整数M <= N <= 10^4，请输出PM到PN的所有素数。

输入格式：

输入在一行中给出M和N，其间以空格分隔。

输出格式：

输出从PM到PN的所有素数，每10个数字占1行，其间以空格分隔，但行末不得有多余空格。

输入样例：

5 27

输出样例：

11 13 17 19 23 29 31 37 41 43

47 53 59 61 67 71 73 79 83 89

97 101 103

分析：vector中保存第M到第N个素数，用cnt标记输出了多少个，如果当前已经输出的个数为10的倍数，则输出一个空行～

```
#include <iostream>
#include <vector>
using namespace std;
bool isprime(int a) {
    for (int i = 2; i * i <= a; i++)
        if(a % i == 0) return false;
    return true;
}
int main() {
    int M, N, num = 2, cnt = 0;
    cin >> M >> N;
    vector<int> v;
    while (cnt < N) {
        if (isprime(num)) {
            cnt++;
            if (cnt >= M) v.push_back(num);
        }
        num++;
    }
    cnt = 0;
    for (int i = 0; i < v.size(); i++) {
        cnt++;
        if (cnt % 10 != 1) printf(" ");
        printf("%d", v[i]);
```

```
        if (cnt % 10 == 0) printf("\n");
    }
    return 0;
}
```

1014. 福尔摩斯的约会 (20) [字符串处理]

大侦探福尔摩斯接到一张奇怪的字条：“我们约会吧！3485djDkxh4hhGE 2984akDfkkkkggEdsb s&hgsfdk d&Hyscvnm”。大侦探很快就明白了，字条上奇怪的乱码实际上就是约会的时间“星期四 14:04”，因为前面两字符串中第1对相同的大写英文字母（大小写有区分）是第4个字母‘D’，代表星期四；第2对相同的字符是‘E’，那是第5个英文字母，代表一天里的第14个钟头（于是一天的0点到23点由数字0到9、以及大写字母A到N表示）；后面两字符串第1对相同的英文字母‘s’出现在第4个位置（从0开始计数）上，代表第4分钟。现给定两对字符串，请帮助福尔摩斯解码得到约会的时间。

输入格式：

输入在4行中分别给出4个非空、不包含空格、且长度不超过60的字符串。

输出格式：

在一行中输出约会的时间，格式为“DAY HH:MM”，其中“DAY”是某星期的3字符缩写,即MON表示星期一，TUE表示星期二，WED表示星期三，THU表示星期四，FRI表示星期五，SAT表示星期六，SUN表示星期日。题目输入保证每个测试存在唯一解。

输入样例：

3485djDkxh4hhGE

2984akDfkkkkggEdsb

s&hgsfdk

d&Hyscvnm

输出样例：

THU 14:04

分析：按照题目所给的方法找到相等的字符后判断即可，如果输出的时间不足2位数要在前面添0，即用%02d输出～

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
    string a, b, c, d;
    cin >> a >> b >> c >> d;
    char t[2];
    int pos, i = 0, j = 0;
    while(i < a.length() && i < b.length()) {
        if (a[i] == b[i] && (a[i] >= 'A' && a[i] <= 'G')) {
```

```
            t[0] = a[i];
            break;
        }
        i++;
    }
    i = i + 1;
    while (i < a.length() && i < b.length()) {
        if (a[i] == b[i] && ((a[i] >= 'A' && a[i] <= 'N') || isdigit(a[i]))) {
            t[1] = a[i];
            break;
        }
        i++;
    }
    while (j < c.length() && j < d.length()) {
        if (c[j] == d[j] && isalpha(c[j])) {
            pos = j;
            break;
        }
        j++;
    }
    string week[7] = {"MON ", "TUE ", "WED ", "THU ", "FRI ", "SAT ", "SUN "};
    int m = isdigit(t[1]) ? t[1] - '0' : t[1] - 'A' + 10;
    cout << week[t[0]-'A'];
    printf("%02d:%02d", m, pos);
    return 0;
}
```

1015. 德才论 (25) [排序]

题目描述：

宋代史学家司马光在《资治通鉴》中有一段著名的“德才论”：“是故才德全尽谓之圣人，才德兼亡谓之愚人，德胜才谓之君子，才胜德谓之小人。凡取人之术，苟不得圣人，君子而与之，与其得小人，不若得愚人。”

现给出一批考生的德才分数，请根据司马光的理论给出录取排名。

输入格式：

输入第1行给出3个正整数，分别为：N（<=105），即考生总数；L（>=60），为录取最低分数线，即德分和才分均不低于L的考生才有资格被考虑录取；H（<100），为优先录取线——德分和才分均不低于此线的被定义为“才德全尽”，此类考生按德才总分从高到低排序；才分不到但德分到线的一类考生属于“德胜才”，也按总分排序，但排在第一类考生之后；德才分均低于H，但是德分不低于才分的考生属于“才德兼亡”但尚有“德胜才”者，按总分排序，但排在第二类考生之后；其他达到最低线L的考生也按总分排序，但排在第三类考生之后。

随后N行，每行给出一位考生的信息，包括：准考证号、德分、才分，其中准考证号为8位整数，德才分为区间[0, 100]内的整数。数字间以空格分隔。

输出格式：

输出第1行首先给出达到最低分数线的考生人数M，随后M行，每行按照输入格式输出一位考生的信息，考生按输入中说明的规则从高到低排序。当某类考生中有多人总分相同时，按其德分降序排列；若德分也并列，则按准考证号的升序输出。

输入样例：

14 60 80

10000001 64 90

10000002 90 60

10000011 85 80

10000003 85 80

10000004 80 85

10000005 82 77

10000006 83 76

10000007 90 78

10000008 75 79

10000009 59 90

10000010 88 45

10000012 80 100

10000013 90 99

10000014 66 60

输出样例：

12

10000013 90 99

10000012 80 100

10000003 85 80

10000011 85 80

10000004 80 85

10000007 90 78

10000006 83 76

10000005 82 77

10000002 90 60

10000014 66 60

10000008 75 79

10000001 64 90

分析：用结构体存储。写好cmp函数~结构体数组vector v[4]中v[0]保存第一类考生，v[1]保存第二类考生……以此类推。写好cmp函数很重要，cmp函数中，排序先按照总分排序，然后按照德分排序，最后按照准考证号排序……最后输出符合条件的结果～

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
struct node {
    int num, de, cai;
};
int cmp(struct node a, struct node b) {
    if ((a.de + a.cai) != (b.de + b.cai))
        return (a.de + a.cai) > (b.de + b.cai);
    else if (a.de != b.de)
        return a.de > b.de;
    else
        return a.num < b.num;
}
int main() {
    int n, low, high;
    scanf("%d %d %d", &n, &low, &high);
    vector<node> v[4];
    node temp;
    int total = n;
    for (int i = 0; i < n; i++) {
        scanf("%d %d %d", &temp.num, &temp.de, &temp.cai);
        if (temp.de < low || temp.cai < low)
            total--;
        else if (temp.de >= high && temp.cai >= high)
            v[0].push_back(temp);
        else if (temp.de >= high && temp.cai < high)
            v[1].push_back(temp);
        else if (temp.de < high && temp.cai < high && temp.de >= temp.cai)
            v[2].push_back(temp);
        else
            v[3].push_back(temp);
    }
    printf("%d\n", total);
    for (int i = 0; i < 4; i++) {
```

```
        sort(v[i].begin(), v[i].end(), cmp);
        for (int j = 0; j < v[i].size(); j++)
            printf("%d %d %d\n", v[i][j].num, v[i][j].de, v[i][j].cai);
    }
    return 0;
}
```

1016. 部分A+B (15) [模拟]

正整数A的“DA（为1位整数）部分”定义为由A中所有DA组成的新整数PA。例如：给定A = 3862767，DA = 6，则A的“6部分”PA是66，因为A中有2个6。

现给定A、DA、B、DB，请编写程序计算PA + PB。

输入格式：

输入在一行中依次给出A、DA、B、DB，中间以空格分隔，其中0 < A, B < 10^10。

输出格式：

在一行中输出PA + PB的值。

输入样例1：

3862767 6 13530293 3

输出样例1：

399

输入样例2：

3862767 1 13530293 8

输出样例2：

0

分析：将A和B保存在string a和b中，将DA和DB保存在da和db中，因为A为字符串，所以对于它的每一位a[i]，当da == (a[i]- ‘0’)时候表示da和a[i]相等，则pa = pa * 10 + da；B同理，当db == (b[i]- ‘0’)时候表示db和b[i]相等，则pb = pb * 10 + db；最后输出pa+pb的值～

```
#include <iostream>
using namespace std;
int main() {
    string a, b;
    int da, db, pa = 0, pb = 0;
    cin >> a >> da >> b >> db;
    for (int i = 0; i < a.length(); i++)
        if (da == (a[i] - '0')) pa = pa * 10 + da;
    for (int i = 0; i < b.length(); i++)
        if (db == (b[i] - '0')) pb = pb * 10 + db;
    cout << pa + pb;
    return 0;
}
```

1017. A除以B (20) [大整数运算]

本题要求计算A/B，其中A是不超过1000位的正整数，B是1位正整数。你需要输出商数Q和余数R，

使得A = B * Q + R成立。

输入格式：

输入在1行中依次给出A和B，中间以1空格分隔。

输出格式：

在1行中依次输出Q和R，中间以1空格分隔。

输入样例：

123456789050987654321 7

输出样例：

17636684150141093474 3

分析：模拟手动除法的过程，每次用第一位去除以B，如果得到的商不是0就输出，否则就*10+下一位，直到最后的数为余数～

```
#include <iostream>
using namespace std;
int main() {
    string s;
    int a, t = 0, temp = 0;
    cin >> s >> a;
    int len = s.length();
    t = (s[0] - '0') / a;
    if ((t != 0 && len > 1) || len == 1)
        cout << t;
    temp = (s[0] - '0') % a;
```

```
    for (int i = 1; i < len; i++) {
        t = (temp * 10 + s[i] - '0') / a;
        cout << t;
        temp = (temp * 10 + s[i] - '0') % a;
    }
    cout << " " << temp;
    return 0;
}
```

1018. 锤子剪刀布 (20) [模拟]

大家应该都会玩“锤子剪刀布”的游戏：两人同时给出手势，胜负规则如图所示：现给出两人的交锋记录，请统计双方的胜、平、负次数，并且给出双方分别出什么手势的胜算最大。

输入格式：

输入第1行给出正整数N（<=105），即双方交锋的次数。随后N行，每行给出一次交锋的信息，即甲、乙双方同时给出的的手势。C代表“锤子”、J代表“剪刀”、B代表“布”，第1个字母代表甲方，第2个代表乙方，中间有1个空格。

输出格式：

输出第1、2行分别给出甲、乙的胜、平、负次数，数字间以1个空格分隔。第3行给出两个字母，分别代表甲、乙获胜次数最多的手势，中间有1个空格。如果解不唯一，则输出按字母序最小的解。

输入样例：

10

C J

J B

C B

B B

B C

C C

C B

J B

B C

J J

输出样例：

5 3 2

2 3 5

B B

分析：jiawin、yiwin分别表示甲乙赢的次数，s和t分别表示每一次甲乙给出的手势，maxjia和maxyi分别表示甲乙获胜次数最多的手势所对应的下标（012分别表示BCJ），枚举每一次甲乙手势的胜负结果并累加到jiawin和yiwin中，最后根据题目要求输出结果～

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    int jiawin = 0, yiwin = 0;
    int jia[3] = {0}, yi[3] = {0};
    for (int i = 0; i < n; i++) {
        char s, t;
        cin >> s >> t;
        if (s == 'B' && t == 'C') {
            jiawin++;
            jia[0]++;
        } else if (s == 'B' && t == 'J') {
            yiwin++;
            yi[2]++;
        } else if (s == 'C' && t == 'B') {
            yiwin++;
            yi[0]++;
        } else if (s == 'C' && t == 'J') {
            jiawin++;
            jia[1]++;
        } else if (s == 'J' && t == 'B') {
```

```
            jiawin++;
            jia[2]++;
        } else if (s == 'J' && t == 'C') {
            yiwin++;
            yi[1]++;
        }
    }
    cout << jiawin << " " << n - jiawin - yiwin << " " << yiwin << endl <<
yiwin << " " << n - jiawin - yiwin << " " << jiawin << endl;
    int maxjia = jia[0] >= jia[1] ? 0 : 1;
    maxjia = jia[maxjia] >= jia[2] ? maxjia : 2;
    int maxyi = yi[0] >= yi[1] ? 0 : 1;
    maxyi = yi[maxyi] >= yi[2] ? maxyi : 2;
    char str[4] = {"BCJ"};
    cout << str[maxjia] << " " << str[maxyi];
    return 0;
}
```

1019. 数字黑洞 (20) [数学问题]

给定任一个各位数字不完全相同的4位正整数，如果我们先把4个数字按非递增排序，

再按非递减排序，然后用第1个数字减第2个数字，将得到一个新的数字。一直重复

这样做，我们很快会停在有“数字黑洞”之称的6174，这个神奇的数字也叫Kaprekar常数。

例如，我们从6767开始，将得到

7766 – 6677 = 1089

9810 – 0189 = 9621

9621 – 1269 = 8352

8532 – 2358 = 6174

7641 – 1467 = 6174

… …

现给定任意4位正整数，请编写程序演示到达黑洞的过程。

输入格式：

输入给出一个(0, 10000)区间内的正整数N。

输出格式：

如果N的4位数字全相等，则在一行内输出“N – N = 0000”；否则将计算的每一步在

一行内输出，直到6174作为差出现，输出格式见样例。注意每个数字按4位数格式输出。

输入样例1：

6767

输出样例1：

7766 – 6677 = 1089

9810 – 0189 = 9621

9621 – 1269 = 8352

8532 – 2358 = 6174

输入样例2：

2222

输出样例2：

2222 – 2222 = 0000

分析：有一个测试用例注意点，如果当输入N值为6174的时候，依旧要进行下面的步骤，直到差值为6174才可以～所以用do while语句，无论是什么值总是要执行一遍while语句，直到遇到差值是0000或6174～

s.insert(0, 4 – s.length(), '0');用来给不足4位的时候前面补0～

```
#include <iostream>
#include <algorithm>
using namespace std;
bool cmp(char a, char b) {return a > b;}
int main() {
    string s;
    cin >> s;
    s.insert(0, 4 - s.length(), '0');
    do {
        string a = s, b = s;
        sort(a.begin(), a.end(), cmp);
        sort(b.begin(), b.end());
        int result = stoi(a) - stoi(b);
        s = to_string(result);
        s.insert(0, 4 - s.length(), '0');
        cout << a << " - " << b << " = " << s << endl;
    } while (s != "6174" && s != "0000");
    return 0;
}
```

1020. 月饼 (25) [贪心算法]

月饼是中国人在中秋佳节时吃的一种传统食品，不同地区有许多不同风味的月饼。现给定所有种类月饼的库存量、总售价、以及市场的最大需求量，请你计算可以获得的最大收益是多少。

注意：销售时允许取出一部分库存。样例给出的情形是这样的：假如我们有3种月饼，其库存量分别为18、15、10万吨，总售价分别为75、72、45亿元。如果市场的最大需求量只有20万吨，那么我们最大收益策略应该是卖出全部15万吨第2种月饼、以及5万吨第3种月饼，获得 72 + 45/2 = 94.5（亿元）。

输入格式：

每个输入包含1个测试用例。每个测试用例先给出一个不超过1000的正整数N表示月饼的种类数、以及不超过500（以万吨为单位）的正整数D表示市场最大需求量。随后一行给出N个正数表示每种月饼的库存量（以万吨为单位）；最后一行给出N个正数表示每种月饼的总售价（以亿元为单位）。数字间以空格分隔。

输出格式：

对每组测试用例，在一行中输出最大收益，以亿元为单位并精确到小数点后2位。

输入样例：

3 20

18 15 10

75 72 45

输出样例：

94.50

分析：首先根据月饼的总价和数量计算出每一种月饼的单价，然后将月饼数组按照单价从大到小排序，根据需求量need的大小，从单价最大的月饼开始售卖，将销售掉这种月饼的价格累加到result中，最后输出result即可～

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
struct mooncake{
    float num, price, unit;
};
int cmp(mooncake a, mooncake b) {
    return a.unit > b.unit;
}
int main() {
    int n, need;
    cin >> n >> need;
    vector<mooncake> a(n);
```

```
    for (int i = 0; i < n; i++) scanf("%f", &a[i].num);
    for (int i = 0; i < n; i++) scanf("%f", &a[i].price);
    for (int i = 0; i < n; i++) a[i].unit = a[i].price / a[i].num;
    sort(a.begin(), a.end(), cmp);
    float result = 0.0;
    for (int i = 0; i < n; i++) {
        if (a[i].num <= need) {
            result = result + a[i].price;
        } else {
            result = result + a[i].unit * need;
            break;
        }
        need = need - a[i].num;
    }
    printf("%.2f",result);
    return 0;
}
```

1021. 个位数统计 (15) [字符串处理]

给定一个k位整数N = dk-1*10k-1 + ... + d1*101 + d0 (0<=di<=9, i=0,...,k-1, dk-1>0)，请编写程序统计每种不同的个位数字出现的次数。例如：

给定N = 100311，则有2个0，3个1，和1个3。

输入格式：

每个输入包含1个测试用例，即一个不超过1000位的正整数N。

输出格式：

对N中每一种不同的个位数字，以D:M的格式在一行中输出该位数字D及其在N中出现的次数M。

要求按D的升序输出。

输入样例：

100311

输出样例：

0:2

1:3

3:1

分析：因为N为不超过1000位的正整数，所以用字符串s来接收N，遍历字符串中的每个字符，将每个数字出现的次数保存在数组a中，a[i]表示数字i出现的次数，最后将数组a的下标0-9中所有a[i]不为0的输出即可～

```
#include <iostream>
using namespace std;
int main() {
    string s;
    cin >> s;
    int a[10] = {0};
    for (int i = 0; i < s.length(); i++)
        a[s[i] - '0']++;
    for (int i = 0; i < 10; i++) {
        if (a[i] != 0)
            printf("%d:%d\n", i, a[i]);
    }
    return 0;
}
```

1022. D进制的A+B (20) [进制转换]

输入两个非负10进制整数A和B(<=2^30-1)，输出A+B的D (1 < D <= 10)进制数。

输入格式：

输入在一行中依次给出3个整数A、B和D。

输出格式：

输出A+B的D进制数。

输入样例：

123 456 8

输出样例：

1103

分析：设t = A + B，将每一次t % d的结果保存在int类型的数组s中，然后将t / d，直到 t 等于 0为止，此时s中保存的就是 t 在 D 进制下每一位的结果的倒序，最后倒序输出s数组即可～

```
#include <iostream>
using namespace std;
int main() {
    int a, b, d;
    cin >> a >> b >> d;
    int t = a + b;
    if (t == 0) {
        cout << 0;
        return 0;
    }
    int s[100];
```

```
    int i = 0;
    while (t != 0) {
        s[i++] = t % d;
        t = t / d;
    }
    for (int j = i - 1; j >= 0; j--)
        cout << s[j];
    return 0;
}
```

1023. 组个最小数 (20) [贪心算法]

给定数字0-9各若干个。你可以以任意顺序排列这些数字，但必须全部使用。目标是使得最后得到的数尽可能小（注意0不能做首位）。例如：给定两个0，两个1，三个5，一个8，我们得到的最小的数就是10015558。现给定数字，请编写程序输出能够组成的最小的数。

输入格式：

每个输入包含1个测试用例。每个测试用例在一行中给出10个非负整数，顺序表示我们拥

有数字0、数字1、……数字9的个数。整数间用一个空格分隔。10个数字的总个数不超过50，且至少拥有1个非0的数字。

输出格式：

在一行中输出能够组成的最小的数。

输入样例：

2 2 0 0 0 3 0 0 1 0

输出样例：

10015558

分析：将数字0、数字1、……数字9的个数分别保存在数组a[i]中，因为0不能做首位，所以首先将i从1到9输出第一个a[i]不为0的数字 i ，并将这个 i 保存在变量 t 中，接着输出a[0]个0，最后输出a[t]-1个 t （因为一个 t 已经被放在首位输出过一次了～），最后 i 从 t+1 到 9 输出 a[i] 个 i ～

```
#include <iostream>
using namespace std;
int main() {
    int a[10], t;
    for (int i = 0; i < 10; i++)
        cin >> a[i];
    for (int i = 1; i < 10; i++) {
        if (a[i] != 0) {
            cout << i;
            t = i;
            break;
```

```
        }
    }
    for (int i = 0; i < a[0]; i++) cout << 0;
    for (int i = 0; i < a[t] - 1; i++) cout << t;
    for (int i = t + 1; i < 10; i++)
        for (int k = 0; k < a[i]; k++)
            cout << i;
    return 0;
}
```

1024. 科学计数法 (20) [字符串处理]

科学计数法是科学家用来表示很大或很小的数字的一种方便的方法，其满足正则表达式[+-][1-9]".”[0-9]+E[+-][0-9]+，即数字的整数部分只有1位，小数部分至少有1位，该数字及其指数部分的正负号即使对正数也必定明确给出。

现以科学计数法的格式给出实数A，请编写程序按普通数字表示法输出A，并保证所有有效位都被保留。

输入格式：

每个输入包含1个测试用例，即一个以科学计数法表示的实数A。该数字的存储长度不超过9999字节，且其指数的绝对值不超过9999。

输出格式：

对每个测试用例，在一行中按普通数字表示法输出A，并保证所有有效位都被保留，包括末尾的0。

输入样例1：

+1.23400E-03

输出样例1：

0.00123400

输入样例2：

-1.2E+10

输出样例2：

-12000000000

分析：n保存E后面的字符串所对应的数字，t保存E前面的字符串，不包括符号位。当n<0时表示向前移动，那么先输出0. 然后输出abs(n)-1个0，然后继续输出t中的所有数字；当n>0时候表示向后移动，那么先输出第一个字符，然后将t中尽可能输出n个字符，如果t已经输出到最后一个字符(j == t.length())那么就在后面补n-cnt个0，否则就补充一个小数点. 然后继续输出t剩余的没有输出的字符～

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

```
    string s;
    cin >> s;
    int i = 0;
    while (s[i] != 'E') i++;
    string t = s.substr(1, i-1);
    int n = stoi(s.substr(i+1));
    if (s[0] == '-') cout << "-";
    if (n < 0) {
        cout << "0.";
        for (int j = 0; j < abs(n) - 1; j++) cout << '0';
        for (int j = 0; j < t.length(); j++)
            if (t[j] != '.') cout << t[j];
    } else {
        cout << t[0];
        int cnt, j;
        for (j = 2, cnt = 0; j < t.length() && cnt < n; j++, cnt++) cout <<
t[j];
        if (j == t.length()) {
            for (int k = 0; k < n - cnt; k++) cout << '0';
        } else {
            cout << '.';
            for (int k = j; k < t.length(); k++) cout << t[k];
        }
    }
    return 0;
}
```

1025. 反转链表 (25) [链表]

给定一个常数K以及一个单链表L，请编写程序将L中每K个结点反转。例如：给定L为1→2→3→4→5→6，K为3，则输出应该为3→2→1→6→5→4；如果K为4，则输出应该为4→3→2→1→5→6，即最后不到K个元素不反转。

输入格式：

每个输入包含1个测试用例。每个测试用例第1行给出第1个结点的地址、结点总个数正整数N(<=10^5)、以及正整数K(<=N)，即要求反转的子链结点的个数。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是该结点保存的整数数据，Next是下一结点的地址。

输出格式：

对每个测试用例，顺序输出反转后的链表，其上每个结点占一行，格式与输入相同。

输入样例：

00100 6 4

00000 4 99999

00100 1 12309

68237 6 -1

33218 3 00000

99999 5 68237

12309 2 33218

输出样例：

00000 4 33218

33218 3 12309

12309 2 00100

00100 1 99999

99999 5 68237

68237 6 -1

分析：输入样例正确连接顺序应该是：

/*

00100 1 12309

12309 2 33218

33218 3 00000

00000 4 99999

99999 5 68237

68237 6 -1

*/

还应该考虑输入样例中有不在链表中的结点的情况。所以用个sum计数~

而且，algorithm头文件里面有reverse函数可以直接调用～

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    int first, k, n, temp;
    cin >> first >> n >> k;
    int data[100005], next[100005], list[100005];
    for (int i = 0; i < n; i++) {
```

```
        cin >> temp;
        cin >> data[temp] >> next[temp];
    }
    int sum = 0;//不一定所有的输入的结点都是有用的，加个计数器
    while (first != -1) {
        list[sum++] = first;
        first = next[first];
    }
    for (int i = 0; i < (sum - sum % k); i += k)
        reverse(begin(list) + i, begin(list) + i + k);
    for (int i = 0; i < sum - 1; i++)
        printf("%05d %d %05d\n", list[i], data[list[i]], list[i + 1]);
    printf("%05d %d -1", list[sum - 1], data[list[sum - 1]]);
    return 0;
}
```

1026. 程序运行时间(15) [模拟]

要获得一个C语言程序的运行时间，常用的方法是调用头文件time.h，其中提供了clock()函数，可以捕捉从程序开始运行到clock()被调用时所耗费的时间。这个时间单位是clock tick，即“时钟打点”。同时还有一个常数CLK_TCK，给出了机器时钟每秒所走的时钟打点数。于是为了获得一个函数f的运行时间，我们只要在调用f之前先调用clock()，获得一个时钟打点数C1；在f执行完成后再调用clock()，获得另一个时钟打点数C2；两次获得的时钟打点数之差(C2-C1)就是f运行所消耗的时钟打点数，再除以常数CLK_TCK，就得到了以秒为单位的运行时间。这里不妨简单假设常数CLK_TCK为100。现给定被测函数前后两次获得的时钟打点数，请你给出被测函数运行的时间。

输入格式：

输入在一行中顺序给出2个整数C1和C1。注意两次获得的时钟打点数肯定不相同，即C1 < C2，并且取值在[0, 10^7]。

输出格式：

在一行中输出被测函数运行的时间。运行时间必须按照“hh:mm:ss”（即2位的“时:分:秒”）

格式输出；不足1秒的时间四舍五入到秒。

输入样例：

123 4577973

输出样例：

12:42:59

分析：n表示运行的时间，n为(b-a)/100，因为常数CLK_TCK为100，题目要求不足1秒的时间四舍五入到秒，所以先给(b-a)加上50，这样如果(b-a)/100的小数位大于等于0.5则会进位，小于等于0.5则会舍去，所以n = ((b – a) + 50) / 100，因为要把秒数n化为时:分:秒的格式，一小时等于3600秒，所以hour = n / 3600，此时将n % 3600即为剩下的分钟和秒数；因为一分钟等于60秒，所以minute = n / 60，则n % 60剩下的就是秒，最后用printf的%02d格式自动为不足2位的整数在前面补上0，保证能够按照

格式输出～

```
#include <iostream>
using namespace std;
int main() {
    int a, b;
    cin >> a >> b;
    int n = ((b - a) + 50) / 100;
    int hour = n / 3600;
    n = n % 3600;
    int minute = n / 60, second = n % 60;
    printf("%02d:%02d:%02d", hour, minute, second);
    return 0;
}
```

1027. 打印沙漏(20) [图形打印]

本题要求你写个程序把给定的符号打印成沙漏的形状。

例如给定17个“*”，要求按下列格式打印

*****

 ***

  *

 ***

*****

所谓“沙漏形状”，是指每行输出奇数个符号；各行符号中心对齐；相邻两行符号数差2；符号数先从大到小顺序递减到1，再从小到大顺序递增；首尾符号数相等。

给定任意N个符号，不一定能正好组成一个沙漏。要求打印出的沙漏能用掉尽可能多的符号。

输入格式：

输入在一行给出1个正整数N（<=1000）和一个符号，中间以空格分隔。

输出格式：

首先打印出由给定符号组成的最大的沙漏形状，最后在一行中输出剩下没用掉的符号数。

输入样例：

19 *

输出样例：

*****

***

*

***

*****

2

分析：每个沙漏都是从最中间一行行向上下分别扩展一行，每次扩展行都要比之前一层多2个符号，最中间一行只有 1 个符号，假设扩展的层数为 i，则扩展出去的上边需要的所有符号个数为3 + 5 + 7 + ... + (2i+1) = (3 + 2i + 1) * i / 2 = i * (i + 2)，扩展出去的下边与上边同样多所以乘以2，加上最重要那一行1个符号，所以 总共需要2 * i * (i + 2) + 1个符号，所以i从0到N，找满足(2 * i * (i + 2) + 1) > N的最小的 i，因为符号不能超过N，所以只能扩展出去 i-1 行，用变量row表示从最中间一行需要扩展出去的行数，row = i – 1，接下来开始输出，上面的每一行，对于扩展出去的第 i 层需要输出row – i个空格，接着输出i * 2 + 1个符号c和换行符；对于最中间一行，需要输出row – 1个空格、符号c和换行符；对于下面的每一行，对于扩展出去的第i层，需要输出row-i个空格，接着输出i * 2 + 1个符号c和换行符，因为用掉的符号数为2 * row * (row + 2) + 1，所以最后输出剩下没用掉的符号数为N – (2 * row * (row + 2) + 1)~

```
#include <iostream>
using namespace std;
int main() {
    int N, row = 0;
    char c;
    cin >> N >> c;
    for (int i = 0; i < N; i++) {
        if ((2 * i * (i + 2) + 1) > N) {
            row = i - 1;
            break;
        }
    }
    for (int i = row; i >= 1; i--) {
        for (int k = row - i; k >= 1; k--) cout << " ";
        for (int j = i * 2 + 1; j >= 1; j--) cout << c;
        cout << endl;
    }
    for (int i = 0; i < row; i++) cout << " ";
    cout << c << endl;
    for (int i = 1; i <= row; i++) {
        for (int k = row - i; k >= 1; k--) cout << " ";
        for (int j = i * 2 + 1; j >= 1; j--) cout << c;
        cout << endl;
    }
    cout << (N - (2 * row * (row + 2) + 1));
    return 0;
}
```

1028. 人口普查(20) [查找元素]

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, 10^5]；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

分析：用字符串接收name和birth，如果当前birth >= “1814/09/06"且<= “2014/09/06"，则是有效生日，有效个数cnt++，如果birth >= maxbirth，则更新maxname和maxbirth的值；如果birth <= minbirth，则更新minname和minbirth的值，这里的max和min是指数值上的大小～最后输出cnt，minname和maxname，minname表示最年长的（生日的数值大小最小的），maxname表示最年轻的（生日的数值大小最大的）～

```
#include <iostream>
using namespace std;
int main() {
    int n, cnt = 0;
    cin >> n;
    string name, birth, maxname, minname, maxbirth = "1814/09/06", minbirth =
"2014/09/06";
    for (int i = 0; i < n; i++) {
        cin >> name >> birth;
        if (birth >= "1814/09/06" && birth <= "2014/09/06") {
            cnt++;
            if (birth >= maxbirth) {
```

```
                maxbirth = birth;
                maxname = name;
            }
            if (birth <= minbirth) {
                minbirth = birth;
                minname = name;
            }
        }
    }
    cout << cnt;
    if (cnt != 0) cout << " " << minname << " " << maxname;
    return 0;
}
```

1029. 旧键盘(20) [Hash散列]

旧键盘上坏了几个键，于是在敲一段文字的时候，对应的字符就不会出现。现在给出应该输入的一段文字、以及实际被输入的文字，请你列出肯定坏掉的那些键。

输入格式：

输入在2行中分别给出应该输入的文字、以及实际被输入的文字。每段文字是不超过80个字符的串，由字母A-Z（包括大、小写）、数字0-9、以及下划线“_”（代表空格）组成。题目保证2个字符串均非空。

输出格式：

按照发现顺序，在一行中输出坏掉的键。其中英文字母只输出大写，每个坏键只输出一次。题目保证至少有1个坏键。

输入样例：

7_This_is_a_test

_hs_s_a_es

输出样例：

7TI

分析：用string的find函数～遍历字符串s1，当当前字符s1[i]不在s2中，它的大写也不在ans中时，将当前字符的大写放入ans中，最后输出ans字符串即可～

ps：感谢github上的@xiaorong61给我发的pull request中strchr()函数带来的灵感～让我想到了曾经用过的string的find函数～

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
    string s1, s2, ans;
    cin >> s1 >> s2;
    for (int i = 0; i < s1.length(); i++)
        if (s2.find(s1[i]) == string::npos && ans.find(toupper(s1[i])) ==
string::npos)
            ans += toupper(s1[i]);
    cout << ans;
    return 0;
}
```

1030. 完美数列(25) [two pointers]

给定一个正整数数列，和正整数p，设这个数列中的最大值是M，最小值是m，如果M <= m * p，则称这个数列是完美数列。

现在给定参数p和一些正整数，请你从中选择尽可能多的数构成一个完美数列。

输入格式：

输入第一行给出两个正整数N和p，其中N（<= 105）是输入的正整数的个数，p（<= 109）是给定的参数。第二行给出N个正整数，每个数不超过109。

输出格式：

在一行中输出最多可以选择多少个数可以用它们组成一个完美数列。

输入样例：

10 8

2 3 20 4 5 1 6 7 8 9

输出样例：

8

分析：首先将数列从小到大排序，设当前结果为result = 0，当前最长长度为temp = 0；从i = 0～n，j从i + result到n，【因为是为了找最大的result，所以下一次j只要从i的result个后面开始找就行了】每次计算temp若大于result则更新result，最后输出result的值～

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
int main() {
    int n;
```

```
    long long p;
    scanf("%d%lld", &n, &p);
    vector<int> v(n);
    for (int i = 0; i < n; i++)
        cin >> v[i];
    sort(v.begin(), v.end());
    int result = 0, temp = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + result; j < n; j++) {
            if (v[j] <= v[i] * p) {
                temp = j - i + 1;
                if (temp > result)
                    result = temp;
            } else {
                break;
            }
        }
    }
    cout << result;
    return 0;
}
```

1031. 查验身份证(15) [字符串处理]

一个合法的身份证号码由17位地区、日期编号和顺序编号加1位校验码组成。校验码的计算规则如下：

首先对前17位数字加权求和，权重分配为：{7，9，10，5，8，4，2，1，6，3，7，9，10，5，8，4，2}；

然后将计算的和对11取模得到值Z；最后按照以下关系对应Z值与校验码M的值：

Z：0 1 2 3 4 5 6 7 8 9 10

M：1 0 X 9 8 7 6 5 4 3 2

现在给定一些身份证号码，请你验证校验码的有效性，并输出有问题的号码。

输入格式：

输入第一行给出正整数N（<= 100）是输入的身份证号码的个数。随后N行，

每行给出1个18位身份证号码。

输出格式：

按照输入的顺序每行输出1个有问题的身份证号码。这里并不检验前17位是否合理，只检查前17位是否全为数字且最后1位校验码计算准确。如果所有号码都正常，则输出“All passed”。

输入样例1：

4

320124198808240056

12010X198901011234

110108196711301866

37070419881216001X

输出样例1：

12010X198901011234

110108196711301866

37070419881216001X

输入样例2：

2

320124198808240056

110108196711301862

输出样例2：

All passed

分析：isTrue函数判断身份证号是否正常，如果不正常返回false，判断每一个给出的身份证号，如果不正常，就输出这个身份证号，并且置flag=1表示有不正常的号码，如果所有的号码都是正常，即flag依旧等于0，则输出All passed～在isTrue函数中，先判断前17位是否是数字，如果不是，直接return false，如果是，就将当前字符转化为数字并与a[i]相乘，累加在sum中，对于第18位，如果是X要转化为10～比较b[sum%11]和第18位是否相等，如果相等就返回true，不相等就返回false～

```
#include <iostream>
using namespace std;
int a[17] = {7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2};
int b[11] = {1, 0, 10, 9, 8, 7, 6, 5, 4, 3, 2};
string s;
bool isTrue() {
    int sum = 0;
    for (int i = 0; i < 17; i++) {
        if (s[i] < '0' || s[i] > '9') return false;
        sum += (s[i] - '0') * a[i];
    }
    int temp = (s[17] == 'X') ? 10 : (s[17] - '0');
    return b[sum%11] == temp;
}
int main() {
    int n, flag = 0;
    cin >> n;
    for (int i = 0; i < n; i++) {
```

```
        cin >> s;
        if (!isTrue()) {
            cout << s << endl;
            flag = 1;
        }
    }
    if (flag == 0) cout << "All passed";
    return 0;
}
```

1032. 挖掘机技术哪家强(20) [查找元素]

为了用事实说明挖掘机技术到底哪家强，PAT组织了一场挖掘机技能大赛。现请你根据比赛结果统计出技术最强的那个学校。

输入格式：

输入在第1行给出不超过10^5的正整数N，即参赛人数。随后N行，每行给出一位参赛者的信息和成绩，包括其所代表的学校的编号（从1开始连续编号）、及其比赛成绩（百分制），中间以空格分隔。

输出格式：

在一行中给出总得分最高的学校的编号、及其总分，中间以空格分隔。题目保证答案唯一，没有并列。

输入样例：

6

3 65

2 80

1 100

2 70

3 40

3 0

输出样例：

2 150

分析：用数组a保存每个学校的编号对应的总分，在输入时将每一个分数score累加到学校编号num对应的a[num]中，然后遍历整个数组，将总分最高的学校编号保存在变量 t 中，将最高总分保存在max中，最后输出t和max的值～

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

```
int main() {
    int N;
    cin >> N;
    vector<int> a(N + 1);
    int num, score;
    for (int i = 0; i < N; i++) {
        cin >> num >> score;
        a[num] += score;
    }
    int max = a[1], t = 1;
    for (int i = 2; i <= N; i++) {
        if (max < a[i]) {
            max = a[i];
            t = i;
        }
    }
    cout << t << " " << max;
    return 0;
}
```

1033. 旧键盘打字(20) [Hash散列]

旧键盘上坏了几个键，于是在敲一段文字的时候，对应的字符就不会出现。现在给出应该输入的一段文字、以及坏掉的那些键，打出的结果文字会是怎样?

输入格式:

输入在2行中分别给出坏掉的那些键、以及应该输入的文字。其中对应英文字母的坏键以大写给出；每段文字是不超过105个字符的串。可用的字符包括字母[a-z, A-Z]、数字0-9、以及下划线“_”（代表空格）、“,”、“.”、“-”、“+”（代表上档键）。题目保证第2行输入的文字串非空。

注意：如果上档键坏掉了，那么大写的英文字母无法被打出。

输出格式:

在一行中输出能够被打出的结果文字。如果没有一个字符能被打出，则输出空行。

输入样例:

7+IE.

7_This_is_a_test.

输出样例:

_hs_s_a_tst

分析：坏掉的键保存在字符串bad中，应该输入的文字保存在should中，遍历整个应该输入的字符串，因为坏键以大写给出，所以如果在bad里面找到了should[i]的大写，表示这个字符对应的键坏了，则跳过这个字符不输出，continue跳过～如果should[i]是大写并且在bad中找到了'+'，表示上档键坏了，大写无法输出，所以这个字符也不能输出，continue跳过～如果都没跳过，则要输出should[i]～

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
  string bad, should;
  getline(cin, bad);
  getline(cin, should);
  for (int i = 0, length = should.length(); i < length; i++) {
    if (bad.find(toupper(should[i])) != string::npos) continue;
    if (isupper(should[i]) && bad.find('+') != string::npos) continue;
    cout << should[i];
  }
  return 0;
}
```

1034. 有理数四则运算(20) [分数的四则运算]

本题要求编写程序，计算2个有理数的和、差、积、商。

输入格式：

输入在一行中按照“a1/b1 a2/b2”的格式给出两个分数形式的有理数，其中分子和分母全是整型范围内的整数，负号只可能出现在分子前，分母不为0。

输出格式：

分别在4行中按照“有理数1 运算符 有理数2 = 结果”的格式顺序输出2个有理数的和、差、积、商。注意输出的每个有理数必须是该有理数的最简形式“k a/b”，其中k是整数部分，a/b是最简分数部分；若为负数，则须加括号；若除法分母为0，则输出“Inf”。题目保证正确的输出中没有超过整型范围的整数。

输入样例1：

2/3 -4/2

输出样例1：

2/3 + (-2) = (-1 1/3)

2/3 – (-2) = 2 2/3

2/3 * (-2) = (-1 1/3)

2/3 / (-2) = (-1/3)

输入样例2：

5/3 0/6

输出样例2：

1 2/3 + 0 = 1 2/3

1 2/3 – 0 = 1 2/3

1 2/3 * 0 = 0

1 2/3 / 0 = Inf

分析：func(m, n)的作用是对m/n的分数进行化简，gcd(t1, t2)的作用是计算t1和t2的最大公约数～在func函数中，先看m和n里面是否有0（即m*n是否等于0），如果分母n=0，输出Inf，如果分子m=0，输出"0"～flag表示m和n是否异号，flag=true表示后面要添加负号"(-"和括号")"，再将m和n都转为abs(m)和abs(n)，即取他们的正数部分方便计算～x = m/n为m和n的可提取的整数部分，先根据flag的结果判断是否要在前面追加"(-"，然后根据x是否等于0判断要不要输出这个整数位，接着根据m%n是否等于0的结果判断后面还有没有小分数，如果m能被n整除，表示没有后面的小分数，那么就根据flag的结果判断要不要加")"，然后直接return～如果有整数位，且后面有小分数，则要先输出一个空格，接着处理剩下的小分数，先把m分子减去已经提取出的整数部分，然后求m和n的最大公约数t，让m和n都除以t进行化简～最后输出"m/n"，如果flag==true还要在末尾输出")"

注意：判断m和n是否异号千万不要写成判断m*n是否小于0，因为m*n的结果可能超过了long long int的长度，导致溢出大于0，如果这样写的话会有一个测试点无法通过～（(⊙o⊙)嗯为了这一个测试点找bug找到凌晨两三点的就是我...我好菜啊.jpg）

```
#include <iostream>
#include <cmath>
using namespace std;
long long a, b, c, d;
long long gcd(long long t1, long long t2) {
    return t2 == 0 ? t1 : gcd(t2, t1 % t2);
}
void func(long long m, long long n) {
    if (m * n == 0) {
        printf("%s", n == 0 ? "Inf" : "0");
        return ;
    }
    bool flag = ((m < 0 && n > 0) || (m > 0 && n < 0));
    m = abs(m); n = abs(n);
    long long x = m / n;
    printf("%s", flag ? "(-" : "");
    if (x != 0) printf("%lld", x);
    if (m % n == 0) {
        if(flag) printf(")");
        return ;
    }
    if (x != 0) printf(" ");
    m = m - x * n;
    long long t = gcd(m, n);
    m = m / t; n = n / t;
    printf("%lld/%lld%s", m, n, flag ? ")" : "");
}
```

```
int main() {
    scanf("%lld/%lld %lld/%lld", &a, &b, &c, &d);
    func(a, b); printf(" + "); func(c, d); printf(" = "); func(a * d + b * c,
b * d); printf("\n");
    func(a, b); printf(" - "); func(c, d); printf(" = "); func(a * d - b * c,
b * d); printf("\n");
    func(a, b); printf(" * "); func(c, d); printf(" = "); func(a * c, b * d);
printf("\n");
    func(a, b); printf(" / "); func(c, d); printf(" = "); func(a * d, b * c);
    return 0;
}
```

1035. 插入与归并(25) [two pointers]

根据维基百科的定义：

插入排序是迭代算法，逐一获得输入数据，逐步产生有序的输出序列。每步迭代中，算法从输入序列中取出一元素，将之插入有序序列中正确的位置。如此迭代直到全部元素有序。

归并排序进行如下迭代操作：首先将原始序列看成N个只包含1个元素的有序子序列，然后每次迭代归并两个相邻的有序子序列，直到最后只剩下1个有序的序列。

现给定原始序列和由某排序算法产生的中间序列，请你判断该算法究竟是哪种排序算法?

输入格式：

输入在第一行给出正整数N (<=100)；随后一行给出原始序列的N个整数；最后一行给出由某排序算法产生的中间序列。这里假设排序的目标序列是升序。数字间以空格分隔。

输出格式：

首先在第1行中输出“Insertion Sort”表示插入排序、或“Merge Sort”表示归并排序；然后在第2行中输出用该排序算法再迭代一轮的结果序列。题目保证每组测试的结果是唯一的。数字间以空格分隔，且行末不得有多余空格。

输入样例1：

10

3 1 2 8 7 5 9 4 6 0

1 2 3 7 8 5 9 4 6 0

输出样例1：

Insertion Sort

1 2 3 5 7 8 9 4 6 0

输入样例2：

10

3 1 2 8 7 5 9 4 0 6

1 3 2 8 5 7 4 9 0 6

输出样例2：

Merge Sort

1 2 3 8 4 5 7 9 0 6

分析：先将i指向中间序列中满足从左到右是从小到大顺序的最后一个下标，再将j指向从i+1开始，第一个不满足a[j] == b[j]的下标，如果j顺利到达了下标n，说明是插入排序，再下一次的序列是sort(a, a+i+2);否则说明是归并排序。归并排序就别考虑中间序列了，直接对原来的序列进行模拟归并时候的归并过程，i从0到n/k，每次一段段得sort(a + i * k, a + (i + 1) * k);最后别忘记还有最后剩余部分的sort(a + n / k * k, a + n);这样是一次归并的过程。直到有一次发现a的顺序和b的顺序相同，则再归并一次，然后退出循环～

注意：一开始第三个测试点一直不过，天真的我以为可以模拟一遍归并的过程然后在过程中判断下一步是什么。。然而真正的归并算法它是一个递归过程。。也就是先排左边一半，把左边的完全排列成正确的顺序之后，再排右边一半的。。而不是左右两边一起排列的。。后来改了自己的归并部分判断的代码就过了。。。。◕‿◕。

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    int n, a[100], b[100], i, j;
    cin >> n;
    for (int i = 0; i < n; i++)
        cin >> a[i];
    for (int i = 0; i < n; i++)
        cin >> b[i];
    for (i = 0; i < n - 1 && b[i] <= b[i + 1]; i++);
    for (j = i + 1; a[j] == b[j] && j < n; j++);
    if (j == n) {
        cout << "Insertion Sort" << endl;
        sort(a, a + i + 2);
    } else {
        cout << "Merge Sort" << endl;
        int k = 1, flag = 1;
        while(flag) {
            flag = 0;
            for (i = 0; i < n; i++) {
                if (a[i] != b[i])
                    flag = 1;
            }
            k = k * 2;
```

```
            for (i = 0; i < n / k; i++)
                sort(a + i * k, a + (i + 1) * k);
            sort(a + n / k * k, a + n);
        }
    }
    for (j = 0; j < n; j++) {
        if (j != 0) printf(" ");
        printf("%d", a[j]);
    }
    return 0;
}
```

1036. 跟奥巴马一起编程(15) [图形打印]

美国总统奥巴马不仅呼吁所有人都学习编程，甚至以身作则编写代码，成为美国历史上首位编写计算机代码的总统。2014年底，为庆祝“计算机科学教育周”正式启动，奥巴马编写了很简单的计算机代码：在屏幕上画一个正方形。现在你也跟他一起画吧！

输入格式：

输入在一行中给出正方形边长N（3<=N<=20）和组成正方形边的某种字符C，间隔一个空格。

输出格式：

输出由给定字符C画出的正方形。但是注意到行间距比列间距大，所以为了让结果看上去更像正方形，我们输出的行数实际上是列数的50%（四舍五入取整）。

输入样例：

10 a

输出样例：

aaaaaaaaaa

a   a

a   a

a   a

aaaaaaaaaa

分析：为了让结果看上去更像正方形，输出的行数实际上是列数的50%（四舍五入取整），列数是N，则行数t = N / 2 + N % 2，表示偶数等于除以2，奇数要除以2加1的意思，这样才能满足四舍五入～首先输出第一行，N个c，然后输出中间t-2行，最左边和最右边为一个c，中间为N-2个空格，最后输出最后一行，N个c～

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

```
    int N;
    char c;
    cin >> N >> c;
    int t = N / 2 + N % 2;
    for (int i = 0; i < N; i++)
        cout << c;
    cout << endl;
    for (int i = 0; i < t - 2; i++) {
        cout << c;
        for (int k = 0; k < N - 2; k++)
            cout << " ";
        cout << c << endl;
    }
    for (int i = 0; i < N; i++)
        cout << c;
    return 0;
}
```

1037. 在霍格沃茨找零钱（20) [进制转换]

如果你是哈利·波特迷，你会知道魔法世界有它自己的货币系统 —— 就如海格告诉哈利的：“十七个银西可(Sickle)兑一个加隆(Galleon)，二十九个纳特(Knut)兑一个西可，很容易。”现在，给定哈利应付的价钱P和他实付的钱A，你的任务是写一个程序来计算他应该被找的零钱。

输入格式：

输入在1行中分别给出P和A，格式为“Galleon.Sickle.Knut”，其间用1个空格分隔。这里Galleon是[0, 10^7]区间内的整数，Sickle是[0, 17)区间内的整数，Knut是[0, 29)区间内的整数。

输出格式：

在一行中用与输入同样的格式输出哈利应该被找的零钱。如果他没带够钱，那么输出的应该是负数。

输入样例1：

10.16.27 14.1.28

输出样例1：

3.2.1

输入样例2：

14.1.28 10.16.27

输出样例2：

-3.2.1

分析：abc代表应付的钱，mnt代表实付的钱，首先判断应付的钱是否大于实付的钱，如果大于，说明钱不够，为了简化计算，将a和m、b和n、c和t调换，使得计算(mnt-abc)时是大的减去小的～调换之后别忘记输出负号～xyz分别代表找的钱，从低位向高位计算，如果t < c，说明要向前借位，借一位后自己加上29，否则z = t – c即可～别忘记如果有借位，n要减去1～然后计算中间位，如果n < b，说明要借位，则y = n – b + 17，否则不用借位，y = n – b即可～最后计算最高位x，如果n < b引起了借位，则x = m – a – 1，否则x = m – a即可～最后输出x.y.z～

```
#include <iostream>
using namespace std;
int main() {
    int a, b ,c, m, n, t, x, y, z;
    scanf("%d.%d.%d %d.%d.%d",&a, &b, &c, &m, &n, &t);
    if (a > m || (a == m && b > n) || (a == m && b == n && c > t)) {
        swap(a, m); swap(b, n); swap(c, t);
        printf("-");
    }
    z = t < c ? t - c + 29 : t - c;
    n = t < c ? n - 1 : n;
    y = n < b ? n - b + 17 : n - b;
    x = n < b ? m - a - 1 : m - a;
    printf("%d.%d.%d", x, y, z);
    return 0;
}
```

1038. 统计同成绩学生(20) [Hash散列]

本题要求读入N名学生的成绩，将获得某一给定分数的学生人数输出。

输入格式：

输入在第1行给出不超过 10^5 的正整数N，即学生总人数。随后1行给出N名学生的百分制整数成绩，中间以空格分隔。最后1行给出要查询的分数个数K（不超过N的正整数），随后是K个分数，中间以空格分隔。

输出格式：

在一行中按查询顺序给出得分等于指定分数的学生人数，中间以空格分隔，但行末不得有多余空格。

输入样例：

10

60 75 90 55 75 99 82 90 75 50

3 75 90 88

输出样例：

3 2 0

分析：用b数组保存每个分数对应的学生人数，在输入的时候，对于每一个成绩temp，b[temp]++表示将数组b中对应分数的人数+1～对于m个查询，每一次都输出需要查询的temp所对应的人数b[temp]，注意i不等于0的时候要在输出人数之前输出一个空格～

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, m, temp;
    scanf("%d", &n);
    vector<int> b(101);
    for (int i = 0; i < n; i++) {
        scanf("%d", &temp);
        b[temp]++;
    }
    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d", &temp);
        if (i != 0) printf(" ");
        printf("%d", b[temp]);
    }
    return 0;
}
```

1039. 到底买不买 (20) [Hash散列]

小红想买些珠子做一串自己喜欢的珠串。卖珠子的摊主有很多串五颜六色的珠串，但是不肯把任何一串拆散了卖。于是小红要你帮忙判断一下，某串珠子里是否包含了全部自己想要的珠子？如果是，那么告诉她有多少多余的珠子；如果不是，那么告诉她缺了多少珠子。为方便起见，我们用[0-9]、[a-z]、[A-Z]范围内的字符来表示颜色.

图 1

输入格式：

每个输入包含1个测试用例。每个测试用例分别在2行中先后给出摊主的珠串和小红想做的珠串，两串都不超过1000个珠子。

输出格式：

如果可以买，则在一行中输出“Yes”以及有多少多余的珠子；如果不可以买，则在一行中输出“No”以及缺了多少珠子。其间以1个空格分隔。

输入样例1：

ppRYYGrrYBR2258

YrR8RrY

输出样例1：

Yes 8

输入样例2：

ppRYYGrrYB225

YrR8RrY

输出样例2：

No 2

分析：字符串a和b分别存储摊主的珠串和小红想做的珠串，遍历字符串a，将每一个字符出现的次数保存在book数组中，表示摊主的每个珠子的个数，遍历字符串b，如果book[b[i]]>0，表示小红要的珠子摊主有，则book[b[i]]-1，将这个珠子给小红～否则说明小红要的珠子摊主没有，则将统计缺了多少珠子的result++，如果result不等于0，说明缺珠子，则不可以买，输出No以及缺了的珠子个数result，否则说明不缺珠子，可以买，输出Yes以及摊主珠子多余的个数a.length() – b.length()～

```
#include <iostream>
using namespace std;
int book[256];
int main() {
    string a, b;
    cin >> a >> b;
    for (int i = 0; i < a.length(); i++)
        book[a[i]]++;
    int result = 0;
    for (int i = 0; i < b.length(); i++) {
        if (book[b[i]] > 0)
            book[b[i]]--;
        else
            result++;
    }
    if(result != 0)
        printf("No %d", result);
    else
        printf("Yes %d", a.length() - b.length());
    return 0;
}
```

1040. 有几个PAT（25) [逻辑题]

字符串APPAPT中包含了两个单词“PAT”，其中第一个PAT是第2位(P),第4位(A),第6位(T)；

第二个PAT是第3位(P),第4位(A),第6位(T)。

现给定字符串，问一共可以形成多少个PAT?

输入格式：

输入只有一行，包含一个字符串，长度不超过105，只包含P、A、T三种字母。

输出格式：

在一行中输出给定字符串中包含多少个PAT。由于结果可能比较大，只输出对

1000000007取余数的结果。

输入样例：

APPAPT

输出样例：

2

分析：要想知道构成多少个PAT，那么遍历字符串后对于每一A，它前面的P的个数和它后面的T的个数的乘积就是能构成的PAT的个数。然后把对于每一个A的结果相加即可～～辣么就简单啦，只需要先遍历字符串数一数有多少个T～～然后每遇到一个T呢～～countt–;每遇到一个P呢，countp++;然后一遇到字母A呢就countt * countp～～把这个结果累加到result中～～最后输出结果就好啦~~对了别忘记要对10000000007取余哦～～

PS：假设神奇的你对每次都遇到的神奇的为什么要对1000000007取模感兴趣，请戳 https://www.liuchuo.net/archives/645

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    cin >> s;
    int len = s.length(), result = 0, countp = 0, countt = 0;
    for (int i = 0; i < len; i++) {
        if (s[i] == 'T')
            countt++;
    }
    for (int i = 0; i < len; i++) {
        if (s[i] == 'P') countp++;
        if (s[i] == 'T') countt--;
        if (s[i] == 'A') result = (result + (countp * countt) % 1000000007) %
1000000007;
    }
    cout << result;
    return 0;
}
```

1041. 考试座位号(15) [查找元素]

每个PAT考生在参加考试时都会被分配两个座位号，一个是试机座位，一个是考试座位。正常情况下，考生在入场时先得到试机座位号码，入座进入试机状态后，系统会显示该考生的考试座位号码，考试时考生需要换到考试座位就座。但有些考生迟到了，试机已经结束，他们只能拿着领到的试机座位号码求助于你，从后台查出他们的考试座位号码。

输入格式：

输入第一行给出一个正整数N（<=1000），随后N行，每行给出一个考生的信息：“准考证号 试机座位号 考试座位号”。其中准考证号由14位数字组成，座位从1到N编号。输入保证每个人的准考证号都不同，并且任何时候都不会把两个人分配到同一个座位上。考生信息之后，给出一个正整数M（<=N），随后一行中给出M个待查询的试机座位号码，以空格分隔。

输出格式：

对应每个需要查询的试机座位号码，在一行中输出对应考生的准考证号和考试座位号码，中间用1个空格分隔。

输入样例：

4

10120150912233 2 4

10120150912119 4 1

10120150912126 1 3

10120150912002 3 2

2

3 4

输出样例：

10120150912002 2

10120150912119 1

分析：建立string类型的二维数组stu[1005][2]，假设试机座位号为t，将输入得到的学生准考证号s1保存在stu[t][0]中，考试座位号s2保存在stu[t][1]中，对于查询的m个学生，已知试机座位号t，则直接输出stu[t][0]和stu[t][1]即可～

```
#include <iostream>
using namespace std;
int main() {
    string stu[1005][2], s1, s2;
    int n, m, t;
    cin >> n;
    for(int i = 0; i < n; i++) {
        cin >> s1 >> t >> s2;
        stu[t][0] = s1;
        stu[t][1] = s2;
    }
    cin >> m;
    for(int i = 0; i < m; i++) {
        cin >> t;
        cout << stu[t][0] << " " << stu[t][1] << endl;
    }
    return 0;
}
```

1042. 字符统计(20) [Hash散列]

请编写程序，找出一段给定文字中出现最频繁的那个英文字母。

输入格式：

输入在一行中给出一个长度不超过1000的字符串。字符串由ASCII码表中任意可见字符及空格组成，至少包含1个英文字母，以回车结束（回车不算在内）。

输出格式：

在一行中输出出现频率最高的那个英文字母及其出现次数，其间以空格分隔。如果有并列，则输出按字母序最小的那个字母。统计时不区分大小写，输出小写字母。

输入样例：

This is a simple TEST. There ARE numbers and other symbols 1&2&3.....

输出样例：

e 7

分析：因为统计时不区分大小写，输出小写字母，所以先将string s中所有字符用tolower转为小写～然后遍历字符串s，用islower判断每一个字符是否是字母，如果是字母，就将数组a对应的下标s[i]-‘a’统计加1，接着遍历数组a，将出现次数最高的那个英文字母的数组下标i保存在t中，出现的最多次数保存在max中，最后输出下标t对应的字母t + ‘a’和max的值～

```
#include <iostream>
#include <cctype>
#include <string>
using namespace std;
int main() {
    string s;
    getline(cin, s);
    int a[26] = {0};
    for (int i = 0; i < s.length(); i++)
        s[i] = tolower(s[i]);
    for (int i = 0; i < s.length(); i++)
        if (islower(s[i])) a[s[i] - 'a']++;
    int max = a[0], t = 0;
    for (int i = 1; i < 26; i++) {
        if (a[i] > max) {
            max = a[i];
            t = i;
        }
    }
    printf("%c %d", t + 'a', max);
    return 0;
}
```

1043. 输出PATest(20) [Hash散列]

给定一个长度不超过10000的、仅由英文字母构成的字符串。请将字符重新调整顺序，按“PATestPATest....”这样的顺序输出，并忽略其它字符。当然，六种字符的个数不一定是一样多的，若某种字符已经输出完，则余下的字符仍按PATest的顺序打印，直到所有字符都被输出。

输入格式：

输入在一行中给出一个长度不超过10000的、仅由英文字母构成的非空字符串。

输出格式：

在一行中按题目要求输出排序后的字符串。题目保证输出非空。

输入样例：

redlesPayBestPATTopTeePHPereatitAPPT

输出样例：

PATestPATestPTetPTePePee

分析：将字符串的每一个字符出现的个数保存在int map[128]中，然后依次输出PATest，每次输出一次字符就将map对应的字符个数减1～

PS：感谢@xiaorong61提供的pull request～

```
#include <iostream>
using namespace std;
int main() {
  int map[128] = {0}, c;
  while ((c = cin.get()) != EOF) map[c]++;
  while (map['P'] > 0 || map['A'] > 0 || map['T'] > 0 || map['e'] > 0 ||
map['s'] > 0 || map['t'] > 0) {
    if (map['P']-- > 0) cout << 'P';
    if (map['A']-- > 0) cout << 'A';
    if (map['T']-- > 0) cout << 'T';
    if (map['e']-- > 0) cout << 'e';
    if (map['s']-- > 0) cout << 's';
    if (map['t']-- > 0) cout << 't';
  }
  return 0;
}
```

1044. 火星数字(20) [map映射，STL的使用]

火星人是以13进制计数的：地球人的0被火星人称为tret。地球人数字1到12的火星文分别为：jan, feb, mar, apr, may, jun, jly, aug, sep, oct, nov, dec。火星人将进位以后的12个高位数字分别称为：tam, hel, maa, huh, tou, kes, hei, elo, syy, lok, mer, jou。例如地球人的数字“29”翻译成火星文就是“hel mar”；而火星文“elo nov”对应地球数字“115”。为了方便交流，请你编写程序实现地球和火星数字之间

的互译。

输入格式：

输入第一行给出一个正整数N（<100），随后N行，每行给出一个[0, 169)区间内的数字 —— 或者是地球文，或者是火星文。

输出格式：

对应输入的每一行，在一行中输出翻译后的另一种语言的数字。

输入样例：

4

29

5

elo nov

tam

输出样例：

hel mar

may

115

13

分析：因为给出的可能是数字（地球文）也有可能是字母（火星文），所以用字符串s保存每一次的输入，因为如果是火星文则会出现空格，所以用getline接收一行的输入～计算string s的长度len，判断s[0]是否是数字，如果是数字，表示是地球文，则需要转为火星文，执行func1()；如果不是数字，则说明是火星文，需要转为地球文，执行func2()；

func1(int t)中，传入的值是string转int后的结果stoi(s)，因为数字最大不超过168，所以最多只会输出两位火星文，如果t / 13不等于0，说明有高位，所以输出b[t/13]；如果输出了高位（t/13不等于0）并且t % 13不等于0，说明有高位且有低位，所以此时输出空格；如果t % 13不等于0，说明有低位，此时输出a[t % 13]；注意，还有个数字0没有考虑，因为数字0取余13等于0，但是要特别输出tret，所以在func1的最后一句判断中加一句t == 0，并将a[0]位赋值成tret即可解决0的问题～

func2()中，t1和t2一开始都赋值0，s1和s2用来分离火星文单词，因为火星文字符串只可能一个单词或者两个单词，而且一个单词不会超过4，所以先将一个单词的赋值给s1，即s1 = s.substr(0, 3)；如果len > 4，就将剩下的一个单词赋值给s2，即s2 = s.substr(4, 3)；然后下标j从1到12遍历a和b两个数组，如果a数组中有和s1或者s2相等的，说明低位等于j，则将j赋值给t2；如果b数组中有和s1相等的（b数组不会和s2相等，因为如果有两个单词，s2只可能是低位），说明高位有值，将j赋值给t1，最后输出t1 * 13 + t2即可～

```
#include <iostream>
#include <string>
using namespace std;
string a[13] = {"tret", "jan", "feb", "mar", "apr", "may", "jun", "jly",
"aug", "sep", "oct", "nov", "dec"};
string b[13] = {"####", "tam", "hel", "maa", "huh", "tou", "kes", "hei",
"elo", "syy", "lok", "mer", "jou"};
string s;
int len;
void func1(int t) {
    if (t / 13) cout << b[t / 13];
    if ((t / 13) && (t % 13)) cout << " ";
    if (t % 13 || t == 0) cout << a[t % 13];
}
void func2() {
    int t1 = 0, t2 = 0;
    string s1 = s.substr(0, 3), s2;
    if (len > 4) s2 = s.substr(4, 3);
    for (int j = 1; j <= 12; j++) {
        if (s1 == a[j] || s2 == a[j]) t2 = j;
        if (s1 == b[j]) t1 = j;
    }
    cout << t1 * 13 + t2;
}
int main() {
    int n;
    cin >> n;
    getchar();
    for (int i = 0; i < n; i++) {
        getline(cin, s);
        len = s.length();
        if (s[0] >= '0' && s[0] <= '9')
            func1(stoi(s));
        else
            func2();
        cout << endl;
    }
    return 0;
}
```

1045. 快速排序(25) [快速排序]

著名的快速排序算法里有一个经典的划分过程：我们通常采用某种方法取一个元素作为主元，通过交换，把比主元小的元素放到它的左边，比主元大的元素放到它的右边。 给定划分后的N个互不相同的正整数的排列，请问有多少个元素可能是划分前选取的主元?

例如给定N = 5, 排列是1、3、2、4、5。则：

1的左边没有元素，右边的元素都比它大，所以它可能是主元；

尽管3的左边元素都比它小，但是它右边的2它小，所以它不能是主元；

尽管2的右边元素都比它大，但其左边的3比它大，所以它不能是主元；

类似原因，4和5都可能是主元。

因此，有3个元素可能是主元。

输入格式：

输入在第1行中给出一个正整数N（<= 105）； 第2行是空格分隔的N个不同的正整数，每个数不超过109。

输出格式：

在第1行中输出有可能是主元的元素个数；在第2行中按递增顺序输出这些元素，其间以1个空格分隔，行末不得有多余空格。

输入样例：

5

1 3 2 4 5

输出样例：

3

1 4 5

分析：对原序列sort排序，逐个比较，当当前元素没有变化并且它左边的所有值的最大值都比它小的时候就可以认为它一定是主元（很容易证明正确性的，毕竟无论如何当前这个数要满足左边都比他小右边都比他大，那它的排名【当前数在序列中处在第几个】一定不会变）～

如果硬编码就直接运行超时了...后来才想到这种方法～

一开始有一个测试点段错误，后来才想到因为输出时候v[0]是非法内存，改正后发现格式错误（好像可以说明那个第2个测试点是0个主元？...）然后加了最后一句printf(“\n”);才正确（难道是当没有主元的时候必须要输出空行吗...）

```
#include <iostream>
#include <algorithm>
#include <vector>
int v[100000];
using namespace std;
int main() {
    int n, max = 0, cnt = 0;
    scanf("%d", &n);
    vector<int> a(n), b(n);
```

```
    for (int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
        b[i] = a[i];
    }
    sort(a.begin(), a.end());
    for (int i = 0; i < n; i++) {
        if(a[i] == b[i] && b[i] > max)
            v[cnt++] = b[i];
        if (b[i] > max)
            max = b[i];
    }
    printf("%d\n", cnt);
    for(int i = 0; i < cnt; i++) {
        if (i != 0) printf(" ");
        printf("%d", v[i]);
    }
    printf("\n");
    return 0;
}
```

1046. 划拳(15) [模拟]

划拳是古老中国酒文化的一个有趣的组成部分。酒桌上两人划拳的方法为：每人口中喊出一个数字，同时用手比划出一个数字。如果谁比划出的数字正好等于两人喊出的数字之和，谁就赢了，输家罚一杯酒。两人同赢或两人同输则继续下一轮，直到唯一的赢家出现。

下面给出甲、乙两人的划拳记录，请你统计他们最后分别喝了多少杯酒。

输入格式：

输入第一行先给出一个正整数N（<=100），随后N行，每行给出一轮划拳的记录，格式为：

甲喊 甲划 乙喊 乙划

其中“喊”是喊出的数字，“划”是划出的数字，均为不超过100的正整数（两只手一起划）。

输出格式：

在一行中先后输出甲、乙两人喝酒的杯数，其间以一个空格分隔。

输入样例：

5

8 10 9 12

5 10 5 10

3 8 5 12

12 18 1 13

4 16 12 15

输出样例：

1 2

分析：jia和yi分别代表甲乙两人喝酒的杯数，sum表示甲喊和乙喊之和，如果sum == 甲划，并且sum != 乙划，表示乙输了，乙喝酒杯数yi++，反之jia++，最后输出jia和yi的值～

```
#include <iostream>
using namespace std;
int main() {
    int n, jia = 0, yi = 0, jiahan, jiahua, yihan, yihua, sum;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> jiahan >> jiahua >> yihan >> yihua;
        sum = jiahan + yihan;
        if (sum == jiahua && sum != yihua)
            yi++;
        if (sum != jiahua && sum == yihua)
            jia++;
    }
    cout << jia << " " << yi;
    return 0;
}
```

1047. 编程团体赛(20) [Hash散列]

编程团体赛的规则为：每个参赛队由若干队员组成；所有队员独立比赛；参赛队的成绩为所有队员的成绩和；成绩最高的队获胜。现给定所有队员的比赛成绩，请你编写程序找出冠军队。

输入格式：

输入第一行给出一个正整数N（<=10000），即所有参赛队员总数。随后N行，每行给出一位队员的成绩，格式为：“队伍编号-队员编号 成绩”，其中“队伍编号”为1到1000的正整数，“队员编号”为1到10的正整数，“成绩”为0到100的整数。

输出格式:

在一行中输出冠军队的编号和总成绩，其间以一个空格分隔。注意：题目保证冠军队是唯一的。

输入样例：

6

3-10 99

11-5 87

102-1 0

102-3 100

11-9 89

3-2 61

输出样例：

11 176

分析：用team数组保存每个队伍的总得分，输入的时候将成绩score累加到队伍编号t的team[t]中，比较team数组中所有得分，将最大值的下标保存在max中，最后输出最高分的队伍编号max和对应的总得分team[max]～

```
#include <iostream>
using namespace std;
int main() {
    int n, t, num, score;
    cin >> n;
    int team[1001] = {0};
    for (int i = 1; i <= n; i++) {
        scanf("%d-%d %d", &t, &num, &score);
        team[t] += score;
    }
    int max = 0;
    for (int i = 0; i < 1001; i++) {
        if (team[max] < team[i])
            max = i;
    }
    cout << max << " " << team[max];
    return 0;
}
```

1048. 数字加密(20) [字符串处理]

本题要求实现一种数字加密方法。首先固定一个加密用正整数A，对任一正整数B，将其每1位数字与A的对应位置上的数字进行以下运算：对奇数位，对应位的数字相加后对13取余——这里用J代表10、Q代表11、K代表12；对偶数位，用B的数字减去A的数字，若结果为负数，则再加10。这里令个位为第1位。

输入格式：

输入在一行中依次给出A和B，均为不超过100位的正整数，其间以空格分隔。

输出格式：

在一行中输出加密后的结果。

输入样例：

1234567 368782971

输出样例：

3695Q8118

分析：首先将a和b倒置，将字符串a和b中较短的那个末尾添加0直到两个字符串长度相等，然后从0开始依次处理每一位，如果当前位是奇数位（i % 2 == 0）则将a[i]的数字加上b[i]的数字再对13取余，结果添加在字符串c的末尾；如果是偶数位，计算b[i]和a[i]的差值，如果小于0就加10，然后将结果添加在字符串c的末尾，最后倒序输出字符串c～

```
#include <iostream>
#include <algorithm>
using namespace std;
int main() {
    string a, b, c;
    cin >> a >> b;
    int lena = a.length(), lenb = b.length();
    reverse(a.begin(), a.end());
    reverse(b.begin(), b.end());
    if (lena > lenb)
        b.append(lena - lenb, '0');
    else
        a.append(lenb - lena, '0');
    char str[14] = {"0123456789JQK"};
    for (int i = 0; i < a.length(); i++) {
        if (i % 2 == 0) {
            c += str[(a[i] - '0' + b[i] - '0') % 13];
        } else {
            int temp = b[i] - a[i];
            if (temp < 0) temp = temp + 10;
            c += str[temp];
        }
    }
    for (int i = c.length() - 1; i >= 0; i--)
        cout << c[i];
    return 0;
}
```

1049. 数列的片段和(20) [数学问题]

给定一个正数数列，我们可以从中截取任意的连续的几个数，称为片段。例如，给定数列{0.1, 0.2, 0.3, 0.4}，我们有(0.1) (0.1, 0.2) (0.1, 0.2, 0.3) (0.1, 0.2, 0.3, 0.4) (0.2) (0.2, 0.3) (0.2, 0.3, 0.4) (0.3) (0.3, 0.4) (0.4) 这10个片段。
给定正整数数列，求出全部片段包含的所有的数之和。如本例中10个片段总和是0.1 + 0.3 + 0.6 + 1.0 + 0.2 + 0.5 + 0.9 + 0.3 + 0.7 + 0.4 = 5.0。
输入格式：

输入第一行给出一个不超过105的正整数N，表示数列中数的个数，第二行给出N个不超过1.0的正数，是数列中的数，其间以空格分隔。
输出格式：
在一行中输出该序列所有片段包含的数之和，精确到小数点后2位。
输入样例：
4
0.1 0.2 0.3 0.4
输出样例：
5.00

分析：将数列中的每个数字读取到temp中，假设我们选取的片段中包括temp，且这个片段的首尾指针分别为p和q，那么对于p，有i种选择，即12...i，对于q，有n-i+1种选择，即i, i+1, ... n，所以p和q组合形成的首尾片段有i * (n-i+1)种，因为每个里面都会出现temp，所以temp引起的总和为temp * i * (n – i + 1)；遍历完所有数字，将每个temp引起的总和都累加到sum中，最后输出sum的值～

【Update 2020-6-18】PS：题目更新了测试样例，导致之前的代码测试点2（第3个测试点）无法通过，研究到凌晨四点找错误原因...后来找到了给PAT这道题提反馈改测试用例的这位同学写的关于这道题的博客：https://blog.zhengrh.com/post/about-double/，感谢他让我找到了错误的原因，大概意思是：N比较大时，double类型的值多次累加导致的精度误差，因为输入为十进制小数，存储到double中时，计算机内部使用二进制表示，且计算机的字长有限，有的十进制浮点数使用二进制无法精确表示只能无限接近，在字长的限制下不可避免会产生舍入误差，这些细微的误差在N较大时多次累加会产生较大误差，所以建议不要使用double类型进行多次累加的精确计算，而是转为能够精确存储的整型。尝试把输入的double类型的值扩大1000倍后转为long long整型累加，同时使用long long类型保存sum的值，输出时除以1000.0转为浮点型再输出（相当于把小数点向后移动3位后再计算，避免double类型的小数部分存储不精确，多次累加后对结果产生影响）
但我觉得乘以1000也未必严谨，可能测试样例最小只有小数点后三位，如果测试样例变成小数点后四位、五位、六位，乘以1000相当于直接在小数点后三位处截断，而原本第四五六位经过多次累加进位后依然可能会引起精度问题，但如果乘以10000就会超出long long的值，我认为最精确的应该是截取到所有小数中最大的位数的那一位。。可能我的想法有疏漏，经过测试，测试样例确实没有超过小数点后三位，虽然修改为乘以1000后代码已经AC，但如果对测试样例稍加修改，可能又会导致不AC了...所以这道题先打个问号吧，我猜可能将来题目样例还会被修改...

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cin >> n;
    long long sum = 0;
    double temp;
    for (int i = 1; i <= n; i++) {
        cin >> temp;
        sum += (long long)(temp * 1000) * i * (n - i + 1);
    }
    printf("%.2f", sum / 1000.0);
    return 0;
}
```

1050. 螺旋矩阵(25) [模拟]

本题要求将给定的N个正整数按非递增的顺序，填入“螺旋矩阵”。所谓“螺旋矩阵”，是指从左上角第1个格子开始，按顺时针螺旋方向填充。要求矩阵的规模为m行n列，满足条件：m*n等于N；m>=n；且m-n取所有可能值中的最小值。

输入格式：

输入在第1行中给出一个正整数N，第2行给出N个待填充的正整数。所有数字不超过10^4，相邻数字以空格分隔。

输出格式：

输出螺旋矩阵。每行n个数字，共m行。相邻数字以1个空格分隔，行末不得有多余空格。

输入样例：

12

37 76 20 98 76 42 53 95 60 81 58 93

输出样例：

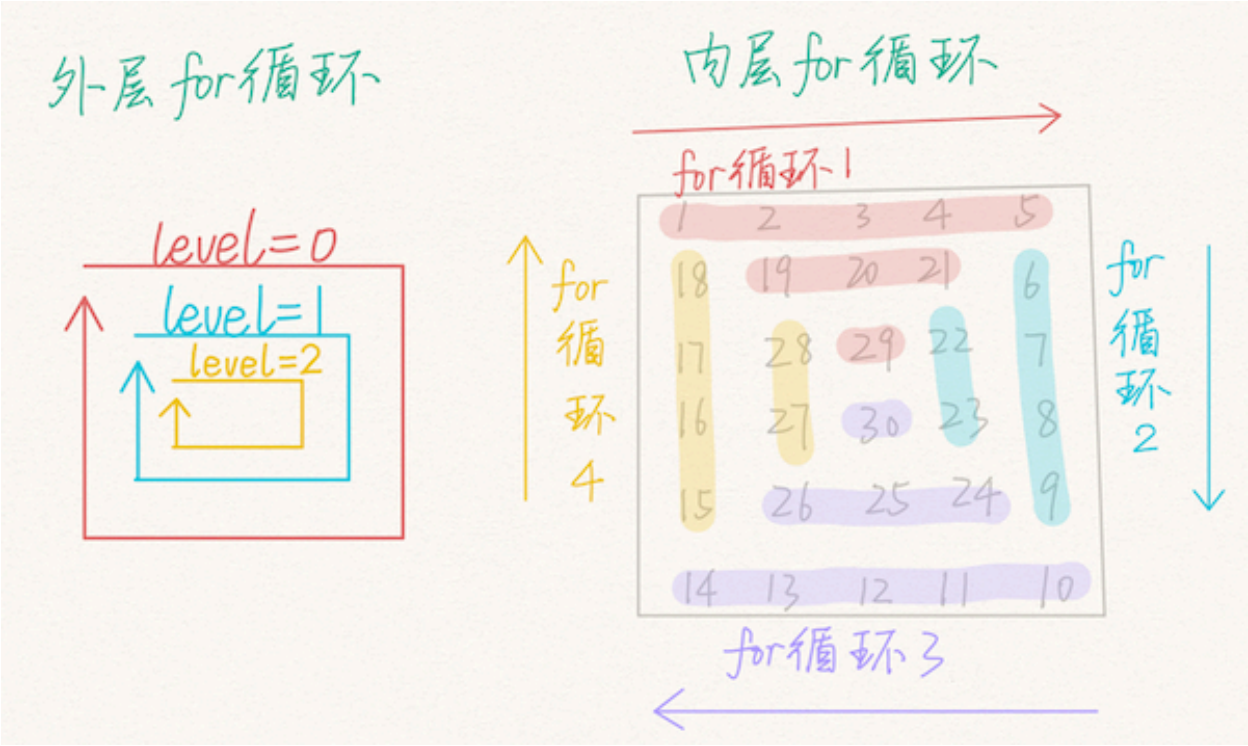
98 95 93

42 37 81

53 20 76

58 60 76

分析：首先计算行数m和列数n的值，n从根号N的整数部分开始，往前推一直到1，找到第一个满足N % n== 0的，m的值等于N/n～将N个给定的值输入数组a，并将a数组中的值按非递增排序，接着建立m行n列的数组b，填充时按层数填充，一个包裹矩阵的口字型为一层，计算螺旋矩阵的层数level，如果m的值为偶数，层数为m/2，如果m为奇数，层数为m/2+1，所以level = m / 2 + m % 2；因为是从左上角第1个格子开始，按顺时针螺旋方向填充，所以外层for循环控制层数i从0到level，内层for循环按左上到右上、右上到右下、右下到左下、左下到左上的顺序一层层填充，注意内层for循环中还要控制t <= N – 1，因为如果螺旋矩阵中所有的元素已经都填充完毕，就不能再重复填充～填充完毕后，输出整个矩阵～




```
#include <iostream>
#include <algorithm>
#include <cmath>
#include <vector>
using namespace std;
int cmp(int a, int b) {return a > b;}
int main() {
    int N, m, n, t = 0;
    scanf("%d", &N);
    for (n = sqrt((double)N); n >= 1; n--) {
        if (N % n == 0) {
            m = N / n;
            break;
        }
    }
    vector<int> a(N);
    for (int i = 0; i < N; i++)
        scanf("%d", &a[i]);
    sort(a.begin(), a.end(), cmp);
    vector<vector<int> > b(m, vector<int>(n));
    int level = m / 2 + m % 2;
    for (int i = 0; i < level; i++) {
        for (int j = i; j <= n - 1 - i && t <= N - 1; j++)
                b[i][j] = a[t++];
        for (int j = i + 1; j <= m - 2 - i && t <= N - 1; j++)
                b[j][n - 1 - i] = a[t++];
        for (int j = n - i - 1; j >= i && t <= N - 1; j--)
```

```
                b[m - 1 - i][j] = a[t++];
        for (int j = m - 2 - i; j >= i + 1 && t <= N - 1; j--)
                b[j][i] = a[t++];
    }
    for (int i = 0; i < m; i++) {
        for (int j = 0 ; j < n; j++) {
            printf("%d", b[i][j]);
            if (j != n - 1) printf(" ");
        }
        printf("\n");
    }
    return 0;
}
```

1051. 复数乘法 (15) [模拟]

复数可以写成(A + Bi)的常规形式，其中A是实部，B是虚部，i是虚数单位，满足i2 = -1；也可以写成极坐标下的指数形式(R*e(Pi))，其中R是复数模，P是辐角，i是虚数单位，其等价于三角形式(R(cos(P) + isin(P))。

现给定两个复数的R和P，要求输出两数乘积的常规形式。

输入格式：

输入在一行中依次给出两个复数的R1, P1, R2, P2，数字间以空格分隔。

输出格式：

在一行中按照“A+Bi”的格式输出两数乘积的常规形式，实部和虚部均保留2位小数。注意：如果B是负数，则应该写成“A-|B|i”的形式。

输入样例：

2.3 3.5 5.2 0.4

输出样例：

-8.68-8.23i

分析：当A或者B小于0但是大于-0.005(比如-0.00001)时候，如果按照A>=0的判断，会输出“-0.00”这样的结果,事实上应该输出“0.00”【B同理，应该输出“+0.00i”】

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
    double r1, p1, r2, p2, A, B;
    cin >> r1 >> p1 >> r2 >> p2;
    A = r1 * r2 * cos(p1) * cos(p2) - r1 * r2 * sin(p1) * sin(p2);
```

```
    B = r1 * r2 * cos(p1) * sin(p2) + r1 * r2 * sin(p1) * cos(p2);
    if (A + 0.005 >= 0 && A < 0)
        printf("0.00");
    else
        printf("%.2f", A);
    if(B >= 0)
        printf("+%.2fi", B);
    else if (B + 0.005 >= 0 && B < 0)
        printf("+0.00i");
    else
        printf("%.2fi", B);
    return 0;
}
```

1052. 卖个萌 (20) [字符串处理]

萌萌哒表情符号通常由“手”、“眼”、“口”三个主要部分组成。简单起见，我们假设一个表情符号是按下列格式输出的：

[左手]([左眼][口][右眼])[右手]

现给出可选用的符号集合，请你按用户的要求输出表情。

输入格式：

输入首先在前三行顺序对应给出手、眼、口的可选符号集。每个符号括在一对方括号[]内。题目保证每个集合都至少有一个符号，并不超过10个符号；每个符号包含1到4个非空字符。

之后一行给出一个正整数K，为用户请求的个数。随后K行，每行给出一个用户的符号选择，顺序为左手、左眼、口、右眼、右手——这里只给出符号在相应集合中的序号（从1开始），数字间以空格分隔。

输出格式：

对每个用户请求，在一行中输出生成的表情。若用户选择的序号不存在，则输出“Are you kidding me? @\/@”。

输入样例：

[╮][╭][o][~\][/~][<][>]

[╯][╰][^][-][=][>][<][@][⊙]

[Д][▽][_][ε][^]

4

1 1 2 2 2

6 8 1 5 5

3 3 4 3 3

2 10 3 9 3

输出样例：

┐(￣▽￣)┌

<(@Д=)/~

o(^ε^)o

Are you kidding me? @\/@

分析：因为不知道一行有多少个表情，所以用一个string类型的不定长二维数组vector存储表情包，以及可以调用v.size()顺便解决不知道一行有多少个表情的问题～～

不定长数组共三行，分别对应输入用例的三行~只不过在存储如数组之前把它们的方括号去掉在存储~这里可以用string的substr方法实现～然后根据输入的数字序号输出对应的表情～耶耶耶能从这道题里面复习好多知识点呢～～

注意：1.“Are you kidding me? @\/@”的‘\’是转义字符，想要输出一个反斜杠就要用两个反斜杠表示～

2.第一个测试点里面包含了空格，所以用cin会失败的，要用getline才能读入一行字符串～

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    vector<vector<string> > v;
    for(int i = 0; i < 3; i++) {
        string s;
        getline(cin, s);
        vector<string> row;
        int j = 0, k = 0;
        while(j < s.length()) {
            if(s[j] == '[') {
                while(k++ < s.length()) {
                    if(s[k] == ']') {
                        row.push_back(s.substr(j+1, k-j-1));
                        break;
                    }
                }
            }
            j++;
        }
        v.push_back(row);
    }
    int n;
    cin >> n;
    for(int i = 0; i < n; i++) {
        int a, b, c, d, e;
```

```
        cin >> a >> b >> c >> d >> e;
        if(a > v[0].size() || b > v[1].size() || c > v[2].size() || d >
v[1].size() || e > v[0].size() || a < 1 || b < 1 || c < 1 || d < 1 || e < 1) {
            cout << "Are you kidding me? @\\/@" << endl;
            continue;
        }
        cout << v[0][a-1] << "(" << v[1][b-1] << v[2][c-1] << v[1][d-1] << ")"
<< v[0][e-1] << endl;
    }
    return 0;
}
```

1053. 住房空置率 (20) [模拟]

在不打扰居民的前提下，统计住房空置率的一种方法是根据每户用电量的连续变化规律进行判断。判断方法如下：

在观察期内，若存在超过一半的日子用电量低于某给定的阈值e，则该住房为“可能空置”；

若观察期超过某给定阈值D天，且满足上一个条件，则该住房为“空置”。

现给定某居民区的住户用电量数据，请你统计“可能空置”的比率和“空置”比率，即以上两种状态的住房占居民区住房总套数的百分比。

输入格式：

输入第一行给出正整数N（<=1000），为居民区住房总套数；正实数e，即低电量阈值；正整数D，即观察期阈值。随后N行，每行按以下格式给出一套住房的用电量数据：

K E1 E2 ... EK

其中K为观察的天数，Ei为第i天的用电量。

输出格式：

在一行中输出“可能空置”的比率和“空置”比率的百分比值，其间以一个空格分隔，保留小数点后1位。

输入样例：

5 0.5 10

6 0.3 0.4 0.5 0.2 0.8 0.6

10 0.0 0.1 0.2 0.3 0.0 0.8 0.6 0.7 0.0 0.5

5 0.4 0.3 0.5 0.1 0.7

11 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1

11 2 2 2 1 1 0.1 1 0.1 0.1 0.1 0.1

输出样例：

40.0% 20.0%

分析：（样例解释：第2、3户为“可能空置”，第4户为“空置”，其他户不是空置）maybe表示可能空置，must表示空置，对于每一个用电数据，判断是否小于阈值e，累计小于e的次数sum，如果sum > (k / 2)，表示超过一半的日子低于阈值e，说明可能空置，再判断观察期k是否大于d，如果大于说明是空置，must++，否则是可能空置，maybe++，最后输出可能空置率maybe/n和空置率must/n～注意，printf中要使用两个百分号%%表示输出一个百分号～

```
#include <iostream>
using namespace std;
int main() {
    int n, d, k, maybe = 0, must = 0;
    double e, temp;
    cin >> n >> e >> d;
    for (int i = 0; i < n; i++) {
        cin >> k;
        int sum = 0;
        for (int j = 0; j < k; j++) {
            cin >> temp;
            if (temp < e) sum++;
        }
        if(sum > (k / 2)) {
            k > d ? must++ : maybe++;
        }
    }
    double mayberesult = (double)maybe / n * 100;
    double mustresult = (double)must / n * 100;
    printf("%.1f%% %.1f%%", mayberesult, mustresult);
    return 0;
}
```

1054. 求平均值 (20) [字符串处理]

本题的基本要求非常简单：给定N个实数，计算它们的平均值。但复杂的是有些输入数据可能是非法的。

一个“合法”的输入是[-1000，1000]区间内的实数，并且最多精确到小数点后2位。

当你计算平均值的时候，不能把那些非法的数据算在内。

输入格式：

输入第一行给出正整数N（<=100）。随后一行给出N个正整数，数字间以一个空格分隔。

输出格式：

对每个非法输入，在一行中输出“ERROR: X is not a legal number”，其中X是输入。最后在一行中输出结果：“The average of K numbers is Y”，其中K是合法输入的个数，Y是它们的平均值，精确到小数点后2位。如果平均值无法计算，则用“Undefined”替换Y。如果K为1，则输出“The average of 1 number is Y”。

输入样例1：

7

5 -3.2 aaa 9999 2.3.4 7.123 2.35

输出样例1：

ERROR: aaa is not a legal number

ERROR: 9999 is not a legal number

ERROR: 2.3.4 is not a legal number

ERROR: 7.123 is not a legal number

The average of 3 numbers is 1.38

输入样例2：

2

aaa -9999

输出样例2：

ERROR: aaa is not a legal number

ERROR: -9999 is not a legal number

The average of 0 numbers is Undefined

分析：使用sscanf和sprintf函数～

sscanf() – 从一个字符串中读进与指定格式相符的数据

sprintf() – 字符串格式化命令，主要功能是把格式化的数据写入某个字符串中

```
#include <iostream>
#include <cstdio>
#include <string.h>
using namespace std;
int main() {
    int n, cnt = 0;
    char a[50], b[50];
    double temp = 0.0, sum = 0.0;
    cin >> n;
    for(int i = 0; i < n; i++) {
        scanf("%s", a);
        sscanf(a, "%lf", &temp);
        sprintf(b, "%.2f",temp);
        int flag = 0;
        for(int j = 0; j < strlen(a); j++)
            if(a[j] != b[j]) flag = 1;
```

```
        if(flag || temp < -1000 || temp > 1000) {
            printf("ERROR: %s is not a legal number\n", a);
            continue;
        } else {
            sum += temp;
            cnt++;
        }
    }
    if(cnt == 1)
        printf("The average of 1 number is %.2f", sum);
    else if(cnt > 1)
        printf("The average of %d numbers is %.2f", cnt, sum / cnt);
    else
        printf("The average of 0 numbers is Undefined");
    return 0;
}
```

1055. 集体照 (25) [排序]

拍集体照时队形很重要，这里对给定的N个人K排的队形设计排队规则如下：

每排人数为N/K（向下取整），多出来的人全部站在最后一排；

后排所有人的个子都不比前排任何人矮；

每排中最高者站中间（中间位置为m/2+1，其中m为该排人数，除法向下取整）；

每排其他人以中间人为轴，按身高非增序，先右后左交替入队站在中间人的两侧（例如5人身高为190、188、186、175、170，则队形为175、188、190、186、170。这里假设你面对拍照者，所以你的左边是中间人的右边）；

若多人身高相同，则按名字的字典序升序排列。这里保证无重名。

现给定一组拍照人，请编写程序输出他们的队形。

输入格式：

每个输入包含1个测试用例。每个测试用例第1行给出两个正整数N（<=10000，总人数）和K（<=10，总排数）。随后N行，每行给出一个人的名字（不包含空格、长度不超过8个英文字母）和身高（[30, 300]区间内的整数）。

输出格式：

输出拍照的队形。即K排人名，其间以空格分隔，行末不得有多余空格。注意：假设你面对拍照者，后排的人输出在上方，前排输出在下方。

输入样例：

10 3

Tom 188

Mike 170

Eva 168

Tim 160

Joe 190

Ann 168

Bob 175

Nick 186

Amy 160

John 159

输出样例：

Bob Tom Joe Nick

Ann Mike Eva

Tim Amy John

分析：建立结构体node，里面包含string类型的姓名name和int类型的身高height～将学生的信息输入到node类型的vector数组stu中～然后对stu数组进行排序（cmp函数表示排序规则，如果身高不等，就按照身高从大到小排列；如果身高相等，就按照名字从小到大的字典序排列～）然后用while循环排列每一行，将每一行应该排列的结果的姓名保存在ans数组中～

因为是面对拍照者，后排的人输出在上方，前排输出在下方，每排人数为N/K（向下取整），多出来的人全部站在最后一排，所以第一排输出的应该是包含多出来的人，所以while循环体中，当row == k时，表示当前是在排列第一行，那么这一行的人数m应该等于总人数n减去后面的k列*(k-1)行，即m = n – n / k * (k-1)；如果不是第一行，那么m直接等于n / k；最中间一个学生应该排在m/2的下标位置，即ans[m / 2] = stu[t].name；然后排左边一列，ans数组的下标 j 从m/2-1开始，一直往左j–，而对于stu的下标 i，是从t+1开始，每次隔一个人选取（即i = i+2，因为另一些人的名字是给右边的），每次把stu[i]的name赋值给ans[j–]；排右边的队伍同理，ans数组的下标 j 从m/2 + 1开始，一直往右j++，stu的下标 i，从t+2开始，每次隔一个人选取（i = i+2），每次把stu[i]的name赋值给ans[j++]，然后输出当前已经排好的ans数组～每一次排完一列row-1，直到row等于0时退出循环表示已经排列并输出所有的行～

```
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;
struct node {
    string name;
    int height;
};
```

```
int cmp(struct node a, struct node b) {
    return a.height != b.height ? a.height > b.height : a.name < b.name;
}
int main() {
    int n, k, m;
    cin >> n >> k;
    vector<node> stu(n);
    for(int i = 0; i < n; i++) {
        cin >> stu[i].name >> stu[i].height;
    }
    sort(stu.begin(), stu.end(), cmp);
    int t = 0, row = k;
    while(row) {
        if(row == k)
            m = n - n / k * (k - 1);
        else
            m = n / k;
        vector<string> ans(m);
        ans[m / 2] = stu[t].name;
        // 左边一列
        int j = m / 2 - 1;
        for(int i = t + 1; i < t + m; i = i + 2)
            ans[j--] = stu[i].name;
        // 右边一列
        j = m / 2 + 1;
        for(int i = t + 2; i < t + m; i = i + 2)
            ans[j++] = stu[i].name;
        // 输出当前排
        cout << ans[0];
        for(int i = 1; i < m; i++)
            cout << " " << ans[i];
        cout << endl;
        t = t + m;
        row--;
    }
    return 0;
}
```

1056. 组合数的和(15) [数学问题]

给定N个非0的个位数字，用其中任意2个数字都可以组合成1个2位的数字。要求所有可能组合出来的2位数字的和。例如给定2、5、8，则可以组合出：25、28、52、58、82、85，它们的和为330。

输入格式：

输入在一行中先给出N（1<N<10），随后是N个不同的非0个位数字。数字间以空格分隔。

输出格式：

输出所有可能组合出来的2位数字的和。

输入样例：

3 2 8 5

输出样例：

330

分析：用sum统计所有可能组合出来的两位数字之和，在sum累加的过程中，对于每一个输入的数字temp，都能和其他N-1个数字组合出新的数字，temp能够放在个位也能够放在十位，所以每个数字temp都能在个位出现(N-1)次，十位出现(N-1)次，在个位产生的累加效果为temp * (N-1)，而在十位产生的累加效果为temp * (N-1) * 10，所以所有数字的累加结果sum即是所有可能组合出来的2位数字的和～

```
#include <cstdio>
int main() {
    int N, sum = 0, temp;
    scanf("%d", &N);
    for (int i = 0; i < N; i++) {
        scanf("%d", &temp);
        sum += temp * 10 * (N - 1) + temp * (N - 1);
    }
    printf("%d", sum);
    return 0;
}
```

1057. 数零壹(20) [进制转换]

给定一串长度不超过10^5的字符串，本题要求你将其中所有英文字母的序号（字母a-z对应序号1-26，不分大小写）相加，得到整数N，然后再分析一下N的二进制表示中有多少0、多少1。例如给定字符串“PAT (Basic)”，其字母序号之和为：16+1+20+2+1+19+9+3=71，而71的二进制是1000111，即有3个0、4个1。

输入格式：

输入在一行中给出长度不超过105、以回车结束的字符串。

输出格式：

在一行中先后输出0的个数和1的个数，其间以空格分隔。

输入样例：

PAT (Basic)

输出样例：

3 4

分析：用getline接收一行字符串，对于字符串的每一位，如果是字母(isalpha)，则将字母转化为大写，并累加(s[i] – ‘A’ + 1)算出n，然后将n转化为二进制，对每一位处理，如果是0则cnt0++，如果是1则cnt1++，最后输出cnt0和cnt1的值～～

```
#include <iostream>
#include <cctype>
#include <string>
using namespace std;
int main() {
    string s;
    getline(cin, s);
    int n = 0;
    for(int i = 0; i < s.length(); i++) {
        if(isalpha(s[i])) {
            s[i] = toupper(s[i]);
            n += (s[i] - 'A' + 1);
        }
    }
    int cnt0 = 0, cnt1 = 0;
    while(n != 0) {
        if(n % 2 == 0) {
            cnt0++;
        } else {
            cnt1++;
        }
        n = n / 2;
    }
    printf("%d %d", cnt0, cnt1);
    return 0;
}
```

1058. 选择题(20) [字符串处理]

批改多选题是比较麻烦的事情，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。注意判题时只有选择全部正确才能得到该题的分数。最后一行输出错得最多的题目的错误次数和编号（题目按照输入的顺序从1开始编号）。如果有并列，则按编号递增顺序输出。数字间用空格分隔，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：

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) (2 b d) (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) (2 b c) (4 a b c d)

输出样例：

3

6

5

2 2 3 4

分析：n个学生，m道题目。对于每一道题目，将题目的总分存储在total[i]数组里面，将题目的选项插入存储在right[i]（为集合类型）里面。wrongCnt[i]存储第i道题错误的人数，对于n个学生，每一个学生的答案插入一个集合st里面，比较st与right[i]是否相等，如果相等说明该生答对了，他的score += total[i]（加上当前题目的总分），如果该生答错了，wrongCnt[i]++，表示第i道题新增一个错误的人。输出每一个学生的得分；遍历wrongCnt数组，求wrongCnt的最大值maxWrongCnt。如果maxWrongCnt == 0说明没有一个人做错题目，则输出“Too simple”，否则输出maxWrongCnt的值，和wrongCnt数组中wrongCnt[i] == maxWrongCnt的那些题号～

注意：scanf中的%d和%c之间一定要有分隔符的主动scanf输入，否则可能接收成空格或者空值～

```
#include <cstdio>
#include <vector>
#include <set>
using namespace std;
int main() {
    int n, m, temp, k;
    scanf("%d%d", &n, &m);
    vector<set<char>> right(m);
    vector<int> total(m), wrongCnt(m);
```

```
    for(int i = 0; i < m; i++) {
        scanf("%d%d%d", &total[i], &temp, &k);
        for(int j = 0; j < k; j++) {
            char c;
            scanf(" %c", &c);
            right[i].insert(c);
        }
    }
    for(int i = 0; i < n; i++) {
        int score = 0;
        scanf("\n");
        for(int j = 0; j < m; j++) {
            if(j != 0) scanf(" ");
            scanf("(%d", &k);
            set<char> st;
            char c;
            for(int l = 0; l < k; l++) {
                scanf(" %c", &c);
                st.insert(c);
            }
            scanf(")");
            if(st == right[j]) {
                score += total[j];
            } else {
                wrongCnt[j]++;
            }
        }
        printf("%d\n", score);
    }
    int maxWrongCnt = 0;
    for(int i = 0; i < m; i++) {
        if(wrongCnt[i] > maxWrongCnt) {
            maxWrongCnt = wrongCnt[i];
        }
    }
    if(maxWrongCnt == 0)
        printf("Too simple");
    else {
        printf("%d", maxWrongCnt);
        for(int i = 0; i < m; i++) {
            if(wrongCnt[i] == maxWrongCnt) {
                printf(" %d", i + 1);
            }
        }
    }
    return 0;
}
```

1059. C语言竞赛(20) [逻辑题]

C语言竞赛是浙江大学计算机学院主持的一个欢乐的竞赛。既然竞赛主旨是为了好玩，颁奖规则也就制定得很滑稽：

0. 冠军将赢得一份“神秘大奖”（比如很巨大的一本学生研究论文集……）。
1. 排名为素数的学生将赢得最好的奖品 —— 小黄人玩偶！
2. 其他人将得到巧克力。

给定比赛的最终排名以及一系列参赛者的ID，你要给出这些参赛者应该获得的奖品。

输入格式：

输入第一行给出一个正整数N（<=10000），是参赛者人数。随后N行给出最终排名，每行按排名顺序给出一位参赛者的ID（4位数字组成）。接下来给出一个正整数K以及K个需要查询的ID。

输出格式：

对每个要查询的ID，在一行中输出“ID: 奖品”，其中奖品或者是“Mystery Award”（神秘大奖）、或者是“Minion”（小黄人）、或者是“Chocolate”（巧克力）。如果所查ID根本不在排名里，打印“Are you kidding?”（耍我呢？）。如果该ID已经查过了（即奖品已经领过了），打印“ID: Checked”（不能多吃多占）。

输入样例：

6

1111

6666

8888

1234

5555

0001

6

8888

0001

1111

2222

8888

2222

输出样例：

8888: Minion

0001: Chocolate

1111: Mystery Award

2222: Are you kidding?

8888: Checked

2222: Are you kidding?

分析：ran数组标记每个id对应的排名，集合ss存储所有已经询问过的id，如果发现当前id已经出现在ss中，则输出“Checked”，如果ran[id] == 0说明当前id不在排名列表中，所以输出“Are you kidding?”，如果ran[id]为1则输出“Minion”，如果ran[id]为素数则输出“Mystery Award”，否则输出“Chocolate”

```
#include <iostream>
#include <set>
#include <cmath>
using namespace std;
int ran[10000];
bool isprime(int a) {
    if(a <= 1) return false;
    int Sqrt = sqrt((double)a);
    for(int i = 2; i <= Sqrt; i++) {
        if(a % i == 0)
            return false;
    }
    return true;
}
int main() {
    int n, k;
    scanf("%d", &n);
    for(int i = 0; i < n; i++) {
        int id;
        scanf("%d", &id);
        ran[id] = i + 1;
    }
    scanf("%d", &k);
    set<int> ss;
    for(int i = 0; i < k; i++) {
        int id;
        scanf("%d", &id);
        printf("%04d: ", id);
        if(ran[id] == 0) {
            printf("Are you kidding?\n");
            continue;
        }
```

```
        if(ss.find(id) == ss.end()) {
            ss.insert(id);
        } else {
            printf("Checked\n");
            continue;
        }
        if(ran[id] == 1) {
            printf("Mystery Award\n");
        }else if(isprime(ran[id])) {
            printf("Minion\n");
        }else {
            printf("Chocolate\n");
        }
    }
    return 0;
}
```

1060. 爱丁顿数(25) [逻辑题]

英国天文学家爱丁顿很喜欢骑车。据说他为了炫耀自己的骑车功力，还定义了一个“爱丁顿数”E，即满足有E天骑车超过E英里的最大整数E。据说爱丁顿自己的E等于87。

现给定某人N天的骑车距离，请你算出对应的爱丁顿数E（<=N）。

输入格式：

输入第一行给出一个正整数N（<=105），即连续骑车的天数；第二行给出N个非负整数，代表每天的骑车距离。

输出格式：

在一行中给出N天的爱丁顿数。

输入样例：

10

6 7 6 9 3 10 8 2 7 8

输出样例：

6

分析：从下标1开始存储n天的公里数在数组a中，对n个数据从大到小排序，i表示了骑车的天数，那么满足a[i] > i的最大值即为所求～

```
#include <iostream>
#include <algorithm>
using namespace std;
int a[1000000];
bool cmp1(int a, int b) {
```

```
    return a > b;
}
int main() {
    int n;
    scanf("%d", &n);
    for(int i = 1; i <= n; i++) scanf("%d", &a[i]);
    sort(a+1, a+n+1, cmp1);
    int ans = 0, p = 1;
    while(ans <= n && a[p] > p) {
        ans++;
        p++;
    }
    printf("%d", ans);
    return 0;
}
```

1061. 判断题(15) [逻辑题]

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

分析：score数组表示每道题的分值，ans数组表示每道题的答案，对于每一个学生，如果他给出的答案temp等于正确答案ans[j]，则将这道题的分数score[j]累加到total中，最后输出total的值～

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, m, temp;
    scanf("%d%d", &n, &m);
    vector<int> score(m), ans(m);
    for(int i = 0; i < m; i++)
        scanf("%d", &score[i]);
    for(int i = 0; i < m; i++)
        scanf("%d", &ans[i]);
    for(int i = 0; i < n; i++) {
        int total = 0;
        for(int j = 0; j < m; j++) {
            scanf("%d", &temp);
            if(temp == ans[j])
                total += score[j];
        }
        printf("%d\n", total);
    }
    return 0;
}
```

1062. 最简分数(20) [分数化简]

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 N1/M1 和 N2/M2，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

分析：使用辗转相除法gcd计算a和b的最大公约数，因为要列出n1/m1和n2/m2之间的最简分数，但是n1/m1不一定小于n2/m2，所以如果n1 * m2 > n2 * m1，说明n1/m1比n2/m2大，则调换n1和n2、m1和m2的位置～假设所求的分数分母为k、分子num，先令num=1，当n1 * k >= m1 * num时，num不断++，直到num符合n1/m1 < num/k为止～然后在n1/m1和n2/m2之间找符合条件的num的值，用gcd(num, k)是否等于1判断num和k是否有最大公约数，如果等于1表示没有最大公约数，就输出num/k，然后num不断++直到退出循环～

```
#include <iostream>
using namespace std;
int gcd(int a, int b){
    return b == 0 ? a : gcd(b, a % b);
}
int main() {
    int n1, m1, n2, m2, k;
    scanf("%d/%d %d/%d %d", &n1, &m1, &n2, &m2, &k);
    if(n1 * m2 > n2 * m1) {
        swap(n1, n2);
        swap(m1, m2);
    }
    int num = 1;
    bool flag = false;
    while(n1 * k >= m1 * num) num++;
    while(n1 * k < m1 * num && m2 * num < n2 * k) {
        if(gcd(num, k) == 1) {
            printf("%s%d/%d", flag == true ? " " : "", num, k);
            flag = true;
        }
        num++;
    }
    return 0;
}
```

1063. 计算谱半径(20) [逻辑题]

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{a1+b1i, ..., an+bni}，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（<= 10000）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

分析：用max保存谱半径，对于n个特征值的实部a和虚部b，模ans = (a的平方+b的平方)的开方，所以ans = sqrt(a * a + b * b)；将ans的最大值保存在max中，最后用%.2f输出max的值～

```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
    int n;
    scanf("%d", &n);
    float max = 0;
    for(int i = 0; i < n; i++) {
        float a, b, ans;
        scanf("%f%f", &a, &b);
        ans = sqrt(a * a + b * b);
        max = ans > max ? ans : max;
    }
    printf("%.2f", max);
    return 0;
}
```

1064. 朋友数(20) [STL-set的使用]

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。注意：我们默认一个整数自己是自己的朋友。

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104。

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

分析：在接收输入数据的时候就把该数字的每一位相加，并把结果插入一个set集合中。因为set是有序的、不重复的，所以set的size值就是输出的个数，set中的每一个数字即所有答案的数字序列~

```
#include <iostream>
#include <set>
using namespace std;
int getFriendNum(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(getFriendNum(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) [逻辑题，STL-set的使用]

“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（<=50000），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（<=10000），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

分析： 设立数组couple[i] = j表示i的对象是j～一开始先设置为都是-1，设立数组isExist表示某人的对象是否来到了派对上～接收数据的时候，对于每一对a和b，将couple的a设置为b，b设置为a，表示他俩是一对～对于每一个需要判断的人，将其存储在guest数组里面，如果它不是单身的（也就是如果它的couple[guest[i]] != -1）那么就将它对象的isExist设置为1，表示他对象的对象（也就是他自己）来到了派对～这样所有isExist不为1的人，对象是没有来到派对的～把所有的人遍历后插入一个集合set里面，set的size就是所求的人数，set里面的所有数就是所求的人的递增排列～

```
#include <iostream>
#include <vector>
#include <set>
using namespace std;
int main() {
    int n, a, b, m;
    scanf("%d", &n);
    vector<int> couple(100000, -1);
```

```
    for (int i = 0; i < n; i++) {
        scanf("%d%d", &a, &b);
        couple[a] = b;
        couple[b] = a;
    }
    scanf("%d", &m);
    vector<int> guest(m), isExist(100000);
    for (int i = 0; i < m; i++) {
        scanf("%d", &guest[i]);
        if (couple[guest[i]] != -1)
            isExist[couple[guest[i]]] = 1;
    }
    set<int> s;
    for (int i = 0; i < m; i++) {
        if (!isExist[guest[i]])
            s.insert(guest[i]);
    }
    printf("%d\n", s.size());
    for (auto it = s.begin(); it != s.end(); it++) {
        if (it != s.begin()) printf(" ");
        printf("%05d", *it);
    }
    return 0;
}
```

1066. 图像过滤(15) [逻辑题]

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 <= 500），另外是待过滤的灰度值区间端点A和B（0 <= A < B <= 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

分析：不用存储到数组中，可以边输入边处理输出～假设当前输入的temp值在a～b区间就将temp替换为num～以%03d的方式输出temp～

```
#include <iostream>
using namespace std;
int main() {
    int m, n, a, b, num, temp;
    scanf("%d%d%d%d%d", &m, &n, &a, &b, &num);
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &temp);
            if (temp >= a && temp <= b)
                temp = num;
            if (j != 0) printf(" ");
            printf("%03d", temp);
        }
        printf("\n");
    }
    return 0;
}
```

1067. 试密码(20) [字符串]

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

输入样例2：

cool@gplt 3

coolman@gplt

coollady@gplt

cool@gplt

try again

#

输出样例2：

Wrong password: coolman@gplt

Wrong password: coollady@gplt

Welcome in

分析：注意3个点：1、如果已经是”#”了就不要继续下面的判断了，不然可能输出Wrong password: “#”

2、如果密码错误并且达到了尝试的次数，是先输出Wrong password那句紧接着输出Account locked那句

3、Wrong password: 后面有个空格～

```
#include <iostream>
using namespace std;
int main() {
    string password, temp;
    int n, cnt = 0;
```

```
    cin >> password >> n;
    getchar();
    while(1) {
        getline(cin, temp);
        if (temp == "#") break;
        cnt++;
        if (cnt <= n && temp == password) {
            cout << "Welcome in";
            break;
        } else if (cnt <= n && temp != password) {
            cout << "Wrong password: " << temp << endl;
            if (cnt == n) {
                cout << "Account locked";
                break;
            }
        }
    }
    return 0;
}
```

1068. 万绿丛中一点红(20) [逻辑题]

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MxN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, 224)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

输入样例2：

4 5 2

0 0 0 0

0 0 3 0

0 0 0 0

0 5 0 0

0 0 0 0

输出样例2：

Not Unique

输入样例3：

3 3 5

1 2 3

3 4 5

5 6 7

输出样例3：

Not Exist

分析：首先这个点必须是唯一的，所以用map标记如果不是唯一的点就不用考虑了～接着对于每个点，判断它的周围八个点与它的差值是否大于阈值，如果有一个点没有满足大于阈值就return false～最后记得输入的时候是列、行——m、n，输出的时候也是列、行坐标～

```
#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) [map映射]

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

输入样例2：

2 3 5

Imgonnawin!

PickMe

输出样例2：

Keep going...

分析：用mapp存储当前用户有没有已经中奖过～当输入的时候，判断当前字符串是否已经在mapp中出现过，如果出现过就将s+1。每次判断i是否等于s，如果等于s且当前用户没有中过奖，就将它的名字输出，并且s = s + n～并将mapp[str]标记为1，且flag标记为true表示有过人中奖。最后flag如果依然是false说明要输出Keep going...

```
#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) [排序，贪心]

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 <= N <= 104)；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104。

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

分析：因为所有长度都要串在一起，每次都等于(旧的绳子长度+新的绳子长度)/2，所以越是早加入绳子长度中的段，越要对折的次数多，所以既然希望绳子长度是最长的，就必须让长的段对折次数尽可能的短。所以将所有段从小到大排序，然后从头到尾从小到大分别将每一段依次加入结绳的绳子中，最后得到的结果才会是最长的结果～

```
#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) [水题]

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.

分析：ans表示n1和n2真实的结果，如果n1 > n2，ans为0，表示应该赌小，否则ans = 1，表示玩家应该赌大。T表示当前玩家有的筹码数，如果T=0，表示玩家已经输光，输出Game Over；如果t > T，表示玩家下注超过持有的筹码量，输出Not enough tokens. Total = 当前的T，如果真实结果ans等于玩家猜的结果，表示玩家赢了，筹码都归玩家，T += t；如果ans不等于b，表示玩家输了，筹码要减去t～

```
#include <iostream>
using namespace std;
int main() {
    int T, K, n1, n2, b, t;
    scanf("%d %d", &T, &K);
    for (int i = 0; i < K; i++) {
        scanf("%d %d %d %d", &n1, &b, &t, &n2);
        int ans = n1 > n2 ? 0 : 1;
        if (T == 0) {
            printf("Game Over.\n");
            return 0;
        } else if (t > T) {
            printf("Not enough tokens.  Total = %d.\n", T);
        } else if (ans == b) {
            T += t;
            printf("Win %d!  Total = %d.\n", t, T);
        } else if (ans != b) {
            T -= t;
            printf("Lose %d.  Total = %d.\n", t, T);
        }
    }
    return 0;
}
```

1072. 开学寄语(20) [简单题]

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。

输入样例:

4 2

2333 6666

CYLL 3 1234 2345 3456

U 4 9966 6666 8888 6666

GG 2 2333 7777

JJ 3 0012 6666 2333

输出样例:

U: 6666 6666

GG: 2333

JJ: 6666 2333

3 5

分析：bool类型的forbid存储禁止携带的物品，如果需要被查缴则赋值为true；flag变量表示当前学生是否已经输出过姓名，一开始flag=false，当前学生如果有需要被查缴的物品且还未输出过他的姓名，则输出name，并令flag=true；如果有需要被查缴的物品且已经输出过姓名，则输出该物品的编号，因为编号为4位数字，不满4位要在前面补0，所以用%04d输出，并将被查缴物品的总数fnum++，最后如果当前学生已经输出过姓名，则输出一个空行，并将学生的总人数snum++，最后输出snum和fnum的值～

```
#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) [逻辑题]

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

分析：错误是指错选或者漏选。用异或运算来判断一个选项和正确选项是否匹配，如果是匹配的，那么异或的结果应当是0；如果不匹配，那么这个选项就是存在错选或者漏选的情况～例如：设a为00001，b为00010，c为00100，d为01000，e为10000，如果给定的正确答案是ac，即10001，那么如果给出的选项也是10001，异或的结果就是0；如果给出的选项是a或者ab，即00001或00011，异或之后不为0，就是错选和漏选的～通过异或操作的结果，再用与运算就可以把错选和漏选的选项找出来～fullscore表示一道题满分的分值；trueopt表示正确的选项，存储的是正确选项二进制的值，二进制由hash给出分别是1,2,4,8,16；cnt是错误的次数，maxcnt是最大错误次数~

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;
```

```
int main() {
    int n, m, optnum, truenum, temp, maxcnt = 0;
    int hash[] = {1, 2, 4, 8, 16}, opt[1010][110] = {0};
    char c;
    scanf("%d %d", &n, &m);
    vector<int> fullscore(m), trueopt(m);
    vector<vector<int>> cnt(m, vector<int>(5));
    for (int i = 0; i < m; i++) {
        scanf("%d %d %d", &fullscore[i], &optnum, &truenum);
        for (int j = 0; j < truenum; j++) {
            scanf(" %c", &c);
            trueopt[i] += hash[c-'a'];
        }
    }
    for (int i = 0; i < n; i++) {
        double grade = 0;
        for (int j = 0; j < m; j++) {
            getchar();
            scanf("(%d", &temp);
            for (int k = 0; k < temp; k++) {
                scanf(" %c)", &c);
                opt[i][j] += hash[c-'a'];
            }
            int el = opt[i][j] ^ trueopt[j];
            if (el) {
                if ((opt[i][j] | trueopt[j]) == trueopt[j]) {
                    grade += fullscore[j] * 1.0 / 2;
                }
                if (el) {
                    for (int k = 0; k < 5; k++)
                        if (el & hash[k]) cnt[j][k]++;
                }
            } else {
                grade += fullscore[j];
            }
        }
        printf("%.1f\n", grade);
    }
    for (int i = 0; i < m; i++)
        for (int j = 0; j < 5; j++)
            maxcnt = maxcnt > cnt[i][j] ? maxcnt : cnt[i][j];

    if (maxcnt == 0) {
        printf("Too simple\n");
    } else {
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < cnt[i].size(); j++) {
```

```
                if (maxcnt == cnt[i][j])
                    printf("%d %d-%c\n", maxcnt, i+1, 'a'+j);
            }
        }
    }
    return 0;
}
```

1074. 宇宙无敌加法器(20) [逻辑题]

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 <=20），以回车结束。 随后两行，每行给出一个不超过N位的正的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

分析：先将要相加的两个字符串S1和S2都扩展到和S等长，然后从后往前按照进制相加到ans中，注意进位carry，最后输出字符串ans，记得不要输出字符串ans前面的0。如果一次都没有输出，最后要输出一个0～

```
#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) [链表]

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1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 (<= 105)；以及正整数K (<=1000)。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[-105, 105]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。

输入样例：

00100 9 10

23333 10 27777

00000 0 99999

00100 18 12309

68237 -6 23333

33218 -4 00000

48652 -2 -1

99999 5 68237

27777 11 48652

12309 7 33218

输出样例：

33218 -4 68237

68237 -6 48652

48652 -2 12309

12309 7 00000

00000 0 99999

99999 5 23333

23333 10 00100

00100 18 27777

27777 11 -1

分析：将结点用list[10000]保存，list为node类型，node中保存结点的值value和它的next地址。list的下标就是结点的地址。将<0、0～k、>k三部分的结点地址分别保存在v[0]、v[1]、v[2]中，最后将vector中的值依次输出即可～

```
#include <iostream>
#include <vector>
using namespace std;
struct node {
    int data, next;
}list[100000];
vector<int> v[3];
int main() {
    int start, n, k, a;
    scanf("%d%d%d", &start, &n, &k);
```

```
    for (int i = 0; i < n; i++) {
        scanf("%d", &a);
        scanf("%d%d", &list[a].data, &list[a].next);
    }
    int p = start;
    while(p != -1) {
        int data = list[p].data;
        if (data < 0)
            v[0].push_back(p);
        else if (data >= 0 && data <= k)
            v[1].push_back(p);
        else
            v[2].push_back(p);
        p = list[p].next;
    }
    int flag = 0;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < v[i].size(); j++) {
            if (flag == 0) {
                printf("%05d %d ", v[i][j], list[v[i][j]].data);
                flag = 1;
            } else {
                printf("%05d\n%05d %d ", v[i][j], v[i][j], list[v[i]
[j]].data);
            }
        }
    }
    printf("-1");
    return 0;
}
```

1076. Wifi密码(15) [简单题]

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

分析：n没什么作用～以字符串方式接收输入，只要遇到任何一个字符串s满足大小为3且s[2]为'T'，就将s[0]字母对应的wifi密码输出～

```
#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) [水题]

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 G1；老师给这个组的评分记为 G2。该组得分为 (G1+G2)/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

分析：maxn和minn分别保存最高分和最低分~将所有其他组评分中的有效分数累加到g1，最后减去minn和maxn并求平均分，最后求得最终得分~

```
#include <iostream>
using namespace std;
int main() {
    int n, m;
    cin >> n >> m;
    for (int i = 0; i < n; i++) {
        int g2, g1 = 0, cnt = -2, temp, maxn = -1, minn = m + 1;
        cin >> g2;
        for (int j = 0; j < n-1; j++) {
            cin >> temp;
            if (temp >= 0 && temp <= m) {
                if (temp > maxn) maxn = temp;
                if (temp < minn) minn = temp;
                g1 += temp;
                cnt++;
            }
        }
```

```
            cout << int((((g1 - minn - maxn) * 1.0 / cnt) + g2) / 2 + 0.5) <<
endl;
        }
        return 0;
}
```

1078. 字符串压缩与解压(20) [逻辑题]

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

输入样例 2：

D

5T2h4is i5s a3 te4st CA3a as10Z

输出样例 2：

TTTTThhiiiis isssss a tesssst CAaaa asZZZZZZZZZZ

分析：1、解压时，cnt初始值为1，若遇到数字则将字符串数字转化为int型数字cnt，之后输出cnt次当前字母。

2、压缩时，用pre保存前一个字母，当当前字母与pre相同时，cnt++；否则输出cnt(cnt>=2时才需输出cnt)和pre，然后令pre=当前字母。最后别忘记结尾处理：当cnt>=2时输出cnt，然后输出pre

```cpp
#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) [逻辑题]

给定一个 k+1 位的正整数 N，写成 ak...a1a0 的形式，其中对所有 i 有 0 <= ai < 10 且 ak > 0。N 被称为一个回文数，当且仅当对所有 i 有 ai = ak-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.

输入样例 2：

196

输出样例 2：

196 + 691 = 887

887 + 788 = 1675

1675 + 5761 = 7436

7436 + 6347 = 13783

13783 + 38731 = 52514

52514 + 41525 = 94039

94039 + 93049 = 187088

187088 + 880781 = 1067869

1067869 + 9687601 = 10755470

10755470 + 07455701 = 18211171

Not found in 10 iterations.

分析：1、将字符串倒置与原字符串比较看是否相等可知s是否为回文串

2、字符串s和它的倒置t相加，只需从头到尾相加然后再倒置（记得要处理最后一个进位carry，如果有进位要在末尾+'1'）

3、倒置可采用algorithm头文件里面的函数reverse(s.begin(), s.end())直接对s进行倒置

```
#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) [map映射，排序]

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期中x 40% + G期末x 60%)，如果 G期中 > G期末；否则总评 G 就是 G期末。这里 G期中 和 G期末 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编程；第二块包含 M 个期中考试成绩 G期中；第三块包含 N 个期末考试成绩 G期末。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编程 G期中 G期末 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

分析：1、因为所有人必须要G编程>=200分，所以用v数组保存所有G编程>=200的人，（一开始gm和gf都为-1），用map映射保存名字所对应v中的下标（为了避免与“不存在”混淆，保存下标+1，当为0时表示该学生的姓名在v中不存在）

2、G期中中出现的名字，如果对应的map并不存在（==0），说明该学生编程成绩不满足条件，则无须保存该学生信息。将存在的人的期中考试成绩更新

3、G期末中出现的名字，也必须保证在map中存在，先更新G期末和G总为新的成绩，当G期末<G期中时再将G总更新为(G期中x 40% + G期末x 60%)

4、将v数组中所有G总满足条件的放入ans数组中，对ans排序（总分递减，总分相同则姓名递增），最后输出ans中的学生信息～

```
#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;
}
```

1081. 检查密码(15) [string字符串]

本题要求你帮助某网站的用户注册模块写一个密码合法性检查的小功能。该网站要求用户设置的密码必须由不少于6个字符组成，并且只能有英文字母、数字和小数点“.”，还必须既有字母也有数字。

输入格式：

输入第一行给出一个正整数 N（<=100），随后 N 行，每行给出一个用户设置的密码，为不超过80个字符的非空字符串，以回车结束。

输出格式：

对每个用户的密码，在一行中输出系统反馈信息，分以下5种：

如果密码合法，输出“Your password is wan mei.”；

如果密码太短，不论合法与否，都输出“Your password is tai duan le.”；

如果密码长度合法，但存在不合法字符，则输出“Your password is tai luan le.”；

如果密码长度合法，但只有字母没有数字，则输出“Your password needs shu zi.”；

如果密码长度合法，但只有数字没有字母，则输出“Your password needs zi mu.”。

输入样例：

5

123s

zheshi.wodepw

1234.5678

WanMei23333

pass*word.6

输出样例：

Your password is tai duan le.

Your password needs shu zi.

Your password needs zi mu.

Your password is wan mei.

Your password is tai luan le.

分析：非空字符串，每个字符串以回车结束，但是字符串里面可能会有空格，所以不能直接用cin，要用getline接收一行字符。在接收完n后要getchar()读取一下换行符才能用getline，否则换行符会被读进getline中～

```
#include <iostream>
#include <cctype>
using namespace std;
int main() {
    int n;
    cin >> n; getchar();
    for (int i = 0; i < n; i++) {
        string s;
        getline(cin, s);
        if (s.length() >= 6) {
            int invalid = 0, hasAlpha = 0, hasNum = 0;
            for (int j = 0; j < s.length(); j++) {
                if (s[j] != '.' && !isalnum(s[j])) invalid = 1;
                else if (isalpha(s[j])) hasAlpha = 1;
                else if (isdigit(s[j])) hasNum = 1;
```

```
            }
            if (invalid == 1) cout << "Your password is tai luan le.\n";
            else if (hasNum == 0) cout << "Your password needs shu zi.\n";
            else if (hasAlpha == 0) cout << "Your password needs zi mu.\n";
            else cout << "Your password is wan mei.\n";
        } else
            cout << "Your password is tai duan le.\n";
    }
    return 0;
}
```

1082. 射击比赛 (20) [逻辑题]

本题目给出的射击比赛的规则非常简单，谁打的弹洞距离靶心最近，谁就是冠军；谁差得最远，谁就是菜鸟。本题给出一系列弹洞的平面坐标(x,y)，请你编写程序找出冠军和菜鸟。我们假设靶心在原点(0,0)。

输入格式：

输入在第一行中给出一个正整数 N（<= 10 000）。随后 N 行，每行按下列格式给出：

ID x y

其中 ID 是运动员的编号（由4位数字组成）；x 和 y 是其打出的弹洞的平面坐标(x,y)，均为整数，且 0 <= |x|, |y| <= 100。题目保证每个运动员的编号不重复，且每人只打 1 枪。

输出格式：

输出冠军和菜鸟的编号，中间空 1 格。题目保证他们是唯一的。

输入样例：

3

0001 5 7

1020 -1 3

0233 0 -1

输出样例：

0233 0001

分析：

1、注意n=1的情况，即冠军和菜鸟都是同一个人的情况（第二个测试点）

2.、注意距离越大的越菜～

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

```
    int n, id, x, y, maxid, maxdis = -1, minid, mindis = 99999;
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> id >> x >> y;
        int dis = x * x + y * y;
        if (dis > maxdis) maxid = id;
        if (dis < mindis) minid = id;
        maxdis = max(maxdis, dis);
        mindis = min(mindis, dis);
    }
    printf("%04d %04d", minid, maxid);
    return 0;
}
```

1083. 是否存在相等的差 (20) [hash映射，map STL]

给定 N 张卡片，正面分别写上 1、2、……、N，然后全部翻面，洗牌，在背面分别写上 1、2、……、N。将每张牌的正反两面数字相减（大减小），得到 N 个非负差值，其中是否存在相等的差?

输入格式:

输入第一行给出一个正整数 N（2 <= N <= 10000），随后一行给出 1 到 N 的一个洗牌后的排列，第 i 个数表示正面写了 i 的那张卡片背面的数字。

输出格式:

按照“差值 重复次数”的格式从大到小输出重复的差值及其重复的次数，每行输出一个结果。

输入样例:

8

3 5 8 6 2 1 4 7

输出样例:

5 2

3 3

2 2

分析：所有差值出现的次数保存在a数组中，从后往前输出所有出现的次数>=2的值～

```
#include <iostream>
using namespace std;
int main() {
    int n, t, a[10000] = {0};
    cin >> n;
    for (int i = 1; i <= n; i++) {
        cin >> t;
        a[abs(t-i)]++;
    }
    for (int i = 9999; i >= 0; i--)
        if (a[i] >= 2) cout << i << " " << a[i] << endl;
    return 0;
}
```

1084. 外观数列 (20) [String字符串处理]

外观数列是指具有以下特点的整数序列：

d, d1, d111, d113, d11231, d112213111, ...

它从不等于 1 的数字 d 开始，序列的第 n+1 项是对第 n 项的描述。比如第 2 项表示第 1 项有 1 个 d，所以就是 d1；第 2 项是 1 个 d（对应 d1）和 1 个 1（对应 11），所以第 3 项就是 d111。又比如第 4 项是 d113，其描述就是 1 个 d，2 个 1，1 个 3，所以下一项就是 d11231。当然这个定义对 d = 1 也成立。本题要求你推算任意给定数字 d 的外观数列的第 N 项。

输入格式：

输入第一行给出[0,9]范围内的一个整数 d、以及一个正整数 N（<=40），用空格分隔。

输出格式：

在一行中给出数字 d 的外观数列的第 N 项。

输入样例：

1 8

输出样例：

1123123111

分析：用string s接收所需变幻的数字，每次遍历s，从当前位置i开始，看后面有多少个与s[i]相同，设j处开始不相同，那么临时字符串 t += s[i] + to_string(j – i);然后再将t赋值给s，cnt只要没达到n次就继续加油循环下一次，最后输出s的值～

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

```
    string s;
    int n, j;
    cin >> s >> n;
    for (int cnt = 1; cnt < n; cnt++) {
        string t;
        for (int i = 0; i < s.length(); i = j) {
            for (j = i; j < s.length() && s[j] == s[i]; j++);
            t += s[i] + to_string(j - i);
        }
        s = t;
    }
    cout << s;
    return 0;
}
```

1085. PAT单位排行 (25) [排序，map STL]

每次 PAT 考试结束后，考试中心都会发布一个考生单位排行榜。本题就请你实现这个功能。

输入格式：

输入第一行给出一个正整数N（<=105），即考生人数。随后N行，每行按下列格式给出一个考生的信息：

准考证号 得分 学校

其中“准考证号”是由6个字符组成的字符串，其首字母表示考试的级别：“B”代表乙级，“A”代表甲级，“T”代表顶级；“得分”是 [0,100] 区间内的整数；“学校”是由不超过6个英文字母组成的单位码（大小写无关）。注意：题目保证每个考生的准考证号是不同的。

输出格式：

首先在一行中输出单位个数。随后按以下格式非降序输出单位的排行榜：

排名 学校 加权总分 考生人数

其中“排名”是该单位的排名（从1开始）；“学校”是全部按小写字母输出的单位码；“加权总分”定义为“乙级总分/1.5 + 甲级总分 + 顶级总分*1.5”的整数部分；“考生人数”是该属于单位的考生的总人数。

学校首先按加权总分排行。如有并列，则应对应相同的排名，并按考生人数升序输出。如果仍然并列，则按单位码的字典序输出。

输入样例：

10

A57908 85 Au

B57908 54 LanX

A37487 60 au

T28374 67 CMU

T32486 24 hypu

A66734 92 cmu

B76378 71 AU

A47780 45 lanx

A72809 100 pku

A03274 45 hypu

输出样例：

5

1 cmu 192 2

1 au 192 3

3 pku 100 1

4 hypu 81 2

4 lanx 81 2

分析：两个map，一个cnt用来存储某学校名称对应的参赛人数，另一个sum计算某学校名称对应的总加权成绩。每次学校名称string school都要转化为全小写，将map中所有学校都保存在vector ans中，类型为node，node中包括学校姓名、加权总分、参赛人数。对ans数组排序，根据题目要求写好cmp函数，最后按要求输出。对于排名的处理：设立pres表示前一个学校的加权总分，如果pres和当前学校的加权总分不同，说明rank等于数组下标+1，否则rank不变～

注意：总加权分数取整数部分是要对最后的总和取整数部分，不能每次都直接用int存储，不然会有一个3分测试点不通过～

PS: 更新后的pat系统会导致之前使用map的代码最后一个测试点超时，更改为unordered_map即可AC～

```
#include <iostream>
#include <algorithm>
#include <cctype>
#include <vector>
#include <unordered_map>
using namespace std;
struct node {
    string school;
    int tws, ns;
};
bool cmp(node a, node b) {
    if (a.tws != b.tws)
```

```
        return a.tws > b.tws;
    else if (a.ns != b.ns)
        return a.ns < b.ns;
    else
        return a.school < b.school;
}
int main() {
    int n;
    scanf("%d", &n);
    unordered_map<string, int> cnt;
    unordered_map<string, double> sum;
    for (int i = 0; i < n; i++) {
        string id, school;
        cin >> id;
        double score;
        scanf("%lf", &score);
        cin >> school;
        for (int j = 0; j < school.length(); j++)
            school[j] = tolower(school[j]);
        if (id[0] == 'B')
            score = score / 1.5;
        else if (id[0] == 'T')
            score = score * 1.5;
        sum[school] += score;
        cnt[school]++;
    }
    vector<node> ans;
    for (auto it = cnt.begin(); it != cnt.end(); it++)
        ans.push_back(node{it->first, (int)sum[it->first], cnt[it->first]});
    sort(ans.begin(), ans.end(), cmp);
    int rank = 0, pres = -1;
    printf("%d\n", (int)ans.size());
    for (int i = 0; i < ans.size(); i++) {
        if (pres != ans[i].tws) rank = i + 1;
        pres = ans[i].tws;
        printf("%d ", rank);
        cout << ans[i].school;
        printf(" %d %d\n", ans[i].tws, ans[i].ns);
    }
    return 0;
}
```

1086. 就不告诉你 (15) [字符串string]

做作业的时候，邻座的小盆友问你：“五乘以七等于多少？”你应该不失礼貌地围笑着告诉他：“五十三。”本题就要求你，对任何一对给定的正整数，倒着输出它们的乘积。

输入格式：

输入在第一行给出两个不超过 1000 的正整数 A 和 B，其间以空格分隔。

输出格式：

在一行中倒着输出 A 和 B 的乘积。

输入样例：

5 7

输出样例：

53

分析：a 和 b的乘积转换成字符串，再将字符串反转，最后将反转过的字符串转换成数字～

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main() {
    int a, b;
    scanf("%d %d", &a, &b);
    string s = to_string(a * b);
    reverse(s.begin(), s.end());
    printf("%d", stoi(s));
    return 0;
}
```

1087. 有多少不同的值 (20) [逻辑题]

当自然数 n 依次取 1、2、3、……、N 时，算式 ⌊n/2⌋+⌊n/3⌋+⌊n/5⌋ 有多少个不同的值？（注：⌊x⌋ 为取整函数，表示不超过 x 的最大自然数，即 x 的整数部分。）

输入格式：

输入给出一个正整数 N（2≤N≤104）。

输出格式：

在一行中输出题面中算式取到的不同值的个数。

输入样例：

2017

输出样例：

1480

分析：把i/2 + i/3 + i/n的值插入到set中，输出set的size就是算式中不同值的个数～

```
#include <iostream>
#include <set>
using namespace std;
int main() {
    int n;
    scanf("%d", &n);
    set<int> s;
    for (int i = 1; i <= n; i++)
        s.insert(i / 2 + i / 3 + i / 5);
    printf("%d", s.size());
    return 0;
}
```

1088. 三人行 (20) [逻辑题]

子曰：“三人行，必有我师焉。择其善者而从之，其不善者而改之。”

本题给定甲、乙、丙三个人的能力值关系为：甲的能力值确定是 2 位正整数；把甲的能力值的 2 个数字调换位置就是乙的能力值；甲乙两人能力差是丙的能力值的 X 倍；乙的能力值是丙的 Y 倍。请你指出谁比你强应“从之”，谁比你弱应“改之”。

输入格式：

输入在一行中给出三个数，依次为：M（你自己的能力值）、X 和 Y。三个数字均为不超过 1000 的正整数。

输出格式：

在一行中首先输出甲的能力值，随后依次输出甲、乙、丙三人与你的关系：如果其比你强，输出 Cong；平等则输出 Ping；比你弱则输出 Gai。其间以 1 个空格分隔，行首尾不得有多余空格。

注意：如果解不唯一，则以甲的最大解为准进行判断；如果解不存在，则输出 No Solution。

输入样例 1：

48 3 7

输出样例 1：

48 Ping Cong Gai

输入样例 2：

48 11 6

输出样例 2：

No Solution

分析：丙不一定是int值，可能是4.5这样的数字～所以要用double存储丙～i、j、k分别代表甲乙丙～i从99遍历到10找到符合题意的那个数字即可～

```
#include <iostream>
#include <cmath>
using namespace std;
int m, x, y;
void print(double t) {
    if (m == t) printf(" Ping");
    else if (m < t) printf(" Cong");
    else printf(" Gai");
}
int main() {
    scanf("%d %d %d", &m, &x, &y);
    for (int i = 99; i >= 10; i--) {
        int j = i % 10 * 10 + i / 10;
        double k = abs(j - i) * 1.0 / x;
        if (j == k * y) {
            cout << i;
            print(i); print(j); print(k);
            return 0;
        }
    }
    cout << "No Solution";
    return 0;
}
```

1089. 狼人杀-简单版 (20) [逻辑题]

以下文字摘自《灵机一动·好玩的数学》：“狼人杀”游戏分为狼人、好人两大阵营。在一局“狼人杀”游戏中，1 号玩家说：“2 号是狼人”，2 号玩家说：“3 号是好人”，3 号玩家说：“4 号是狼人”，4 号玩家说：“5 号是好人”，5 号玩家说：“4 号是好人”。已知这 5 名玩家中有 2 人扮演狼人角色，有 2 人说的不是实话，有狼人撒谎但并不是所有狼人都在撒谎。扮演狼人角色的是哪两号玩家?

本题是这个问题的升级版：已知 N 名玩家中有 2 人扮演狼人角色，有 2 人说的不是实话，有狼人撒谎但并不是所有狼人都在撒谎。要求你找出扮演狼人角色的是哪几号玩家?

输入格式：

输入在第一行中给出一个正整数 N（5≤N≤100）。随后 N 行，第 i 行给出第 i 号玩家说的话（1≤i≤N），即一个玩家编号，用正号表示好人，负号表示狼人。

输出格式：

如果有解，在一行中按递增顺序输出 2 个狼人的编号，其间以空格分隔，行首尾不得有多余空格。如果解不唯一，则输出最小序列解 —— 即对于两个序列 A=a[1],...,a[M] 和 B=b[1],...,b[M]，若存在 0≤k[k+1]<b[k+1]，则称序列 A 小于序列 B。若无解则输出 No Solution。

输入样例 1：

5

-2

+3

-4

+5

+4

输出样例 1：

1 4

输入样例 2：

6

+6

+3

+1

-5

-2

+4

输出样例 2（解不唯一）：

1 5

输入样例 3：

5

-2

-3

-4

-5

-1

输出样例 3：

No Solution

分析：每个人说的数字保存在v数组中，i从1～n、j从i+1～n遍历，分别假设i和j是狼人，a数组表示该人是狼人还是好人，等于1表示是好人，等于-1表示是狼人。k从1～n分别判断k所说的话是真是假，k说的话和真实情况不同（即v[k] * a[abs(v[k])] < 0）则表示k在说谎，则将k放在lie数组中；遍历完成后判断lie数组，如果说谎人数等于2并且这两个说谎的人一个是好人一个是狼人（即a[lie[0]] + a[lie[1]] == 0）表示满足题意，此时输出i和j并return，否则最后的时候输出No Solution～

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;
int main() {
    int n;
    cin >> n;
    vector<int> v(n+1);
    for (int i = 1; i <= n; i++) cin >> v[i];
    for (int i = 1; i <= n; i++) {
        for (int j = i + 1; j <= n; j++) {
            vector<int> lie, a(n + 1, 1);
            a[i] = a[j] = -1;
            for (int k = 1; k <= n; k++)
                if (v[k] * a[abs(v[k])] < 0) lie.push_back(k);
            if (lie.size() == 2 && a[lie[0]] + a[lie[1]] == 0) {
                cout << i << " " << j;
                return 0;
            }
        }
    }
    cout << "No Solution";
    return 0;
}
```

1090. 危险品装箱 (25) [STL-map的应用]

集装箱运输货物时，我们必须特别小心，不能把不相容的货物装在一只箱子里。比如氧化剂绝对不能跟易燃液体同箱，否则很容易造成爆炸。

本题给定一张不相容物品的清单，需要你检查每一张集装箱货品清单，判断它们是否能装在同一只箱子里。

输入格式：

输入第一行给出两个正整数：N (≤10^4) 是成对的不相容物品的对数；M (≤100) 是集装箱货品清单的单数。

随后数据分两大块给出。第一块有 N 行，每行给出一对不相容的物品。第二块有 M 行，每行给出一箱货物的清单，格式如下：

K G[1] G[2] ... G[K]

其中 K (≤1000) 是物品件数，G[i] 是物品的编号。简单起见，每件物品用一个 5 位数的编号代表。两个数字之间用空格分隔。

输出格式：

对每箱货物清单，判断是否可以安全运输。如果没有不相容物品，则在一行中输出 Yes，否则输出 No。

输入样例：

6 3

20001 20002

20003 20004

20005 20006

20003 20001

20005 20004

20004 20006

4 00001 20004 00002 20003

5 98823 20002 20003 20006 10010

3 12345 67890 23333

输出样例：

No

Yes

Yes

分析：用map存储每一个货物的所有不兼容货物～在判断给出的一堆货物是否是相容的时候，判断任一货物的不兼容货物是否在这堆货物中～如果存在不兼容的货物，则这堆货物不能相容～如果遍历完所有的货物，都找不到不兼容的两个货物，则这堆货物就是兼容的～

```
#include <iostream>
#include <vector>
#include <map>
using namespace std;
int main() {
    int n, k, t1, t2;
    map<int,vector<int>> m;
    scanf("%d%d", &n, &k);
    for (int i = 0; i < n; i++) {
        scanf("%d%d", &t1, &t2);
```

```
        m[t1].push_back(t2);
        m[t2].push_back(t1);
    }
    while (k--) {
        int cnt, flag = 0, a[100000] = {0};
        scanf("%d", &cnt);
        vector<int> v(cnt);
        for (int i = 0; i < cnt; i++) {
            scanf("%d", &v[i]);
            a[v[i]] = 1;
        }
        for (int i = 0; i < v.size(); i++)
            for (int j = 0; j < m[v[i]].size(); j++)
                if (a[m[v[i]][j]] == 1) flag = 1;
        printf("%s\n",flag ? "No" :"Yes");
    }
    return 0;
}
```

1091. N-自守数 (15) [逻辑题]

如果某个数 K 的平方乘以 N 以后，结果的末尾几位数等于 K，那么就称这个数为“N-自守数”。例如 3×922 = 25392，而 25392 的末尾两位正好是 92，所以 92 是一个 3-自守数。

本题就请你编写程序判断一个给定的数字是否关于某个 N 是 N-自守数。

输入格式：

输入在第一行中给出正整数 M（≤20），随后一行给出 M 个待检测的、不超过 1000 的正整数。

输出格式：

对每个需要检测的数字，如果它是 N-自守数就在一行中输出最小的 N 和 NK2 的值，以一个空格隔开；否则输出 No。注意题目保证 N<10。

输入样例：

3

92 5 233

输出样例：

3 25392

1 25

No

分析：从1-9枚举，判断是否存在N。通过to_string把乘积转成字符串，再通过substr取末尾子串比较即可～

```cpp
#include <iostream>
#include <string>
using namespace std;
int main() {
    int m;
    cin >> m;
    while (m--) {
        int k, flag = 0;
        cin >> k;
        for (int n = 1; n < 10; n++) {
            int mul = n * k * k;
            string smul = to_string(mul), sk = to_string(k);
            string smulend = smul.substr(smul.length() - sk.length());
            if (smulend == sk) {
                printf("%d %d\n", n, mul);
                flag = 1;
                break;
            }
        }
        if (flag == 0) printf("No\n");
    }
    return 0;
}
```

1092. 最好吃的月饼 (20) [二维数组]

月饼是久负盛名的中国传统糕点之一，自唐朝以来，已经发展出几百品种。

若想评比出一种“最好吃”的月饼，那势必在吃货界引发一场腥风血雨…… 在这里我们用数字说话，给出全国各地各种月饼的销量，要求你从中找出销量冠军，认定为最好吃的月饼。

输入格式：
输入首先给出两个正整数 N（≤1000）和 M（≤100），分别为月饼的种类数（于是默认月饼种类从 1 到 N 编号）和参与统计的城市数量。

接下来 M 行，每行给出 N 个非负整数（均不超过 1 百万），其中第 i 个整数为第 i 种月饼的销量（块）。数字间以空格分隔。

输出格式：
在第一行中输出最大销量，第二行输出销量最大的月饼的种类编号。如果冠军不唯一，则按编号递增顺序输出并列冠军。数字间以 1 个空格分隔，行首尾不得有多余空格。

输入样例：
5 3
1001 992 0 233 6
8 0 2018 0 2008
36 18 0 1024 4
输出样例：
2018
3 5

分析：用sum数组统计每种月饼的总销量，并且求出最大销量。然后遍历每种月饼的销量，等于最大值的就保存输出～

```
#include <iostream>
#include <vector>
using namespace std;
int sum[1005];
int main() {
    int m, n, temp, maxn = 0;
    vector<int> ans;
    cin >> m >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            cin >> temp;
            sum[j] += temp;
            maxn = max(maxn, sum[j]);
        }
    }
    cout << maxn << endl;
    for (int i = 1; i <= m; i++)
        if (sum[i] == maxn) ans.push_back(i);
    for (int i = 0; i < ans.size(); i++) {
        if (i != 0) cout << " ";
```

```
        cout << ans[i];
    }
    return 0;
}
```

1093. 字符串A+B (20) [字符串处理]

给定两个字符串 A 和 B，本题要求你输出 A+B，即两个字符串的并集。要求先输出 A，再输出 B，但重复的字符必须被剔除。

输入格式：

输入在两行中分别给出 A 和 B，均为长度不超过 106 的、由可见 ASCII 字符 (即码32~126)和空格组成的、由回车标识结束的非空字符串。

输出格式：

在一行中输出题面要求的 A 和 B 的和。

输入样例：

This is a sample test

to show you_How it works

输出样例：

This ampletowyu_Hrk

题目大意：求出两个字符串的并集，按照原有顺序输出。意思是按顺序输出，第一次出现的所有字符。

分析：用数组hash标记一下是否是第一次输出即可（字符本身是一个ASCII码值，可当作整数用）

```
#include <iostream>
using namespace std;
int main() {
    string s1, s2, s;
    int hash[200] = {0};
    getline(cin, s1);
    getline(cin, s2);
    s = s1 + s2;
    for (int i = 0; i < s.size(); i++) {
        if (hash[s[i]] == 0) cout << s[i];
        hash[s[i]] = 1;
    }
    return 0;
}
```

1094. 谷歌的招聘 (20) [字符串处理]

2004 年 7 月，谷歌在硅谷的 101 号公路边竖立了一块巨大的广告牌（如下图）用于招聘。内容超级简单，就是一个以 .com 结尾的网址，而前面的网址是一个 10 位素数，这个素数是自然常数 e 中最早出现的 10 位连续数字。能找出这个素数的人，就可以通过访问谷歌的这个网站进入招聘流程的下一步。

自然常数 e 是一个著名的超越数，前面若干位写出来是这样的：e = 2.71828182845904523536028747135266249775724709369995957496696762772407663035354759457138217852516642742746639193200305992 1... 其中粗体标出的 10 位数就是答案。

本题要求你编程解决一个更通用的问题：从任一给定的长度为 L 的数字中，找出最早出现的 K 位连续数字所组成的素数。

输入格式：

输入在第一行给出 2 个正整数，分别是 L（不超过 1000 的正整数，为数字长度）和 K（小于 10 的正整数）。接下来一行给出一个长度为 L 的正整数 N。

输出格式：

在一行中输出 N 中最早出现的 K 位连续数字所组成的素数。如果这样的素数不存在，则输出 404。注意，原始数字中的前导零也计算在位数之内。例如在 200236 中找 4 位素数，0023 算是解；但第一位 2 不能被当成 0002 输出，因为在原始数字中不存在这个 2 的前导零。

输入样例 1：

20 5

23654987725541023819

输出样例 1：

49877

输入样例 2：

10 3

2468024680

输出样例 2：

404

题目大意：给出一个l长度的字符串，求出其中第一个k位的素数

分析: 枚举每个k位的子串，转换成整数，判断是否是素数（判断素数的时候要把0和1也考虑进去）～

```
#include <iostream>
#include <string>
using namespace std;
bool isPrime(int n) {
    if (n == 0 || n == 1) return false;
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0) return false;
    return true;
}
int main() {
    int l, k;
    string s;
    cin >> l >> k >> s;
    for (int i = 0; i <= l - k; i++) {
        string t = s.substr(i, k);
        int num = stoi(t);
        if (isPrime(num)) {
            cout << t;
            return 0;
        }
    }
    cout << "404\n";
    return 0;
}
```

1095. 解码PAT准考证 (25) [模拟，排序，map]

PAT 准考证号由 4 部分组成：

第 1 位是级别，即 T 代表顶级；A 代表甲级；B 代表乙级；

第 2~4 位是考场编号，范围从 101 到 999；

第 5~10 位是考试日期，格式为年、月、日顺次各占 2 位；

最后 11~13 位是考生编号，范围从 000 到 999。

现给定一系列考生的准考证号和他们的成绩，请你按照要求输出各种统计信息。

输入格式：

输入首先在一行中给出两个正整数 N（≤104 ）和 M（≤100），分别为考生人数和统计要求的个数。

接下来 N 行，每行给出一个考生的准考证号和其分数（在区间 [0,100] 内的整数），其间以空格分隔。

考生信息之后，再给出 M 行，每行给出一个统计要求，格式为：类型 指令，其中

类型 为 1 表示要求按分数非升序输出某个指定级别的考生的成绩，对应的 指令 则给出代表指定级别的字母；

类型 为 2 表示要求将某指定考场的考生人数和总分统计输出，对应的 指令 则给出指定考场的编号；

类型 为 3 表示要求将某指定日期的考生人数分考场统计输出，对应的 指令 则给出指定日期，格式与准考证上日期相同。

输出格式：

对每项统计要求，首先在一行中输出 Case #: 要求，其中 # 是该项要求的编号，从 1 开始；要求 即复制输入给出的要求。随后输出相应的统计结果：

类型 为 1 的指令，输出格式与输入的考生信息格式相同，即 准考证号 成绩。对于分数并列的考生，按其准考证号的字典序递增输出（题目保证无重复准考证号）；

类型 为 2 的指令，按 人数 总分 的格式输出；

类型 为 3 的指令，输出按人数非递增顺序，格式为 考场编号 总人数。若人数并列则按考场编号递增顺序输出。

如果查询结果为空，则输出 NA。

输入样例：

8 4

B123180908127 99

B102180908003 86

A112180318002 98

T107150310127 62

A107180908108 100

T123180908010 78

B112160918035 88

A107180908021 98

1 A

2 107

3 180908

2 999

输出样例：

Case 1: 1 A

A107180908108 100

A107180908021 98

A112180318002 98

Case 2: 2 107

3 260

Case 3: 3 180908

107 2

123 2

102 1

Case 4: 2 999

NA

题目大意：给出一组学生的准考证号和成绩，准考证号包含了等级(乙甲顶)，考场号，日期，和个人编号信息，并有三种查询方式

查询一：给出考试等级，找出该等级的考生，按照成绩降序，准考证升序排序

查询二：给出考场号，统计该考场的考生数量和总得分

查询三：给出考试日期，查询改日期下所有考场的考试人数，按照人数降序，考场号升序排序

分析：先把所有考生的准考证和分数记录下来～

1、按照等级查询，枚举选取匹配的学生，然后排序即可

2、按照考场查询，枚举选取匹配的学生，然后计数、求和

3、按日期查询每个考场人数，用unordered_map存储，最后排序汇总～

注意：1、第三个用map存储会超时，用unordered_map就不会超时啦～

2、排序传参建议用引用传参，这样更快，虽然有时候不用引用传参也能通过，但还是尽量用，养成好习惯～

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

```cpp
struct node {
    string t;
    int value;
};
bool cmp(const node &a, const node &b) {
    return a.value != b.value ? a.value > b.value : a.t < b.t;
}
int main() {
    int n, k, num;
    string s;
    cin >> n >> k;
    vector<node> v(n);
    for (int i = 0; i < n; i++)
        cin >> v[i].t >> v[i].value;
    for (int i = 1; i <= k; i++) {
        cin >> num >> s;
        printf("Case %d: %d %s\n", i, num, s.c_str());
        vector<node> ans;
        int cnt = 0, sum = 0;
        if (num == 1) {
            for (int j = 0; j < n; j++)
                if (v[j].t[0] == s[0]) ans.push_back(v[j]);
        } else if (num == 2) {
            for (int j = 0; j < n; j++) {
                if (v[j].t.substr(1, 3) == s) {
                    cnt++;
                    sum += v[j].value;
                }
            }
            if (cnt != 0) printf("%d %d\n", cnt, sum);
        } else if (num == 3) {
            unordered_map<string, int> m;
            for (int j = 0; j < n; j++)
                if (v[j].t.substr(4, 6) == s) m[v[j].t.substr(1, 3)]++;
            for (auto it : m) ans.push_back({it.first, it.second});
        }
        sort(ans.begin(), ans.end(),cmp);
        for (int j = 0; j < ans.size(); j++)
            printf("%s %d\n", ans[j].t.c_str(), ans[j].value);
        if (((num == 1 || num == 3) && ans.size() == 0) || (num == 2 && cnt ==
0)) printf("NA\n");
    }
    return 0;
}
```