**A.**[**老和尚的真言**](http://acm.buaa.edu.cn:8081/contest/169/problem/A/)

**题目描述**

空即是色，色即是空，阿弥陀佛……  
这一周，老和尚外出了，留下一卷真言给小和尚参悟。小和尚翻开这一卷真言，善哉，善哉……这一卷真言，居然是用英文写的！而且还加了密！幸好小和尚很快就看出了门道，他需要做的，只是先把其中全部的“esolc”删掉后，再将删除后得到的字符串中所有的“abc”换成“dnalsi” 即可。正巧，你在小和尚这里打下手，于是他来找你帮忙。

**输入**

第一行包含一个整数t(1≤t≤100)，为数据的组数。  
接下来t行，每行一个字符串（可能含空格），为加密的真言。

**输出**

对于每个字符串，输出一行，为处理后的结果。

**输入样例**

2  
abc close is a good menesolctor!  
aesolcbc

**输出样例**

dnalsi close is a good mentor!  
dnalsi

**解题思路**

**知道string中的find函数和replace函数的话，此题简单很多。**

**参考第十八章string详解。**

**具体两函数的使用看代码和备注。**

**参考代码**

#include <iostream>

#include <cstdio>

#include <string>  **//头文件**

using namespace std;

int main()

{

string s;

int t, i = 1;

cin >> t;

cin.get();  **//注意不能省略该步,否则回车也算一次**

while ( i <= t )

{

getline( cin, s );  **//输入字符串**

int l = s.length();

int esolc = s.find( "esolc" ); **//find函数 查找esolc，结果赋给变量esolc**

while ( esolc != string::npos )  **//npos是查找不到时string的一个返回值**

{

s.replace( esolc, 5, "" );

esolc = s.find( "esolc"); **//继续循环**

}

int abc = s.find( "abc" );

while ( abc != string::npos )

{

s.replace( abc, 3, "dnalsi" );

abc = s.find( "abc");

}

cout << s <<endl;

i++;

}

return 0;

}

**B.**[**盗墓笔记之秦岭神树**](http://acm.buaa.edu.cn:8081/contest/169/problem/B/)

**题目描述**

在天真和老痒还被困在棺材阵中时，Thor却阴差阳错地走了出去，走了另一条不为人知的诡异捷径，率先来到了巨树。好奇的他，独自一人，向上爬去，很快便遭遇了大波的螭蛊。  
经过观察，他发现，这棵树的螭蛊分布如下所示：

第一层：1

第二层：1 x

第三层：1 2x x^2

第四层：1 3x 3x^2 x^3

第五层：1 4x 6x^2 4x^3 x^4

.

.

.

现在Thor想知道第n层有多少只螭蛊，但是我们已经知道了，数学这个东西从来就不是Thor擅长的，因此这个任务就交给了你。

**输入**

多组测试数据，每组测试数据为一行，包含两个整数n与x（1≤n≤1000，1≤x≤1000），含义见题目描述。

**输出**

对于每组测试数据，输出一个整数，为第n层螭蛊的个数，结果对100007取模。  
输入以两个0结尾

**输入样例**

1 100  
2 4  
3 2  
0 0

**输出样例**

1  
5  
9

**解题思路**

**终于见到一个一看到题就知道怎么做的了。。**

**求乘积即可，注意每乘一次都求模。**

**参考代码**

#include <iostream>

using namespace std;

int main()

{

int n, x;

cin >> n >> x;

while ( n != 0 )

{

int sum = 1;

for ( int i = 0; i < n - 1; i++ )

sum = ( sum \* ( x + 1 ) ) % 100007;

cout << sum << endl;

cin >> n >> x;

}

return 0;

}

**C.[So 2](http://acm.buaa.edu.cn:8081/contest/169/problem/C/)**

**题目描述**

# Description

在学了计组之后，Thor终于认识到原来所有的整数在计算机里面都是用2进制储存的！  
好了，你来把一个数x拆分成2的幂的和的形式吧，要求数尽量少，且从小到大输出。

# Input

多组数据。   
每组数据仅有1行1个整数x(MAX\_INT>=x>=1)

# Output

把x拆分成2的幂的形式，数尽量少，从小到大输出。

# Sample Input

3

4

5

6

# Sample Output

1 2

4

1 4

2 4

# Hint

**解题思路**

**每个数都对应唯一的二进制数，那把所需要分解的数先转化为二进制数（最好转化完是从右往左写的，因为结果要求从小到大输出），如6转化为011（倒着看就是6的二进制数），然后输出0\*2^0 , 1\*2^1 , 1\*2^2，即为结果 1 2 4。**

**参考代码**

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

double a[ 100 ];

double t[ 100 ];

int main()

{

int x;

while ( cin >> x )

{

int c = 1, k;

while ( x / 2 != 0 ) **//转化为二进制数的数组**

{

k = x % 2;

x = x / 2;

t[ c ] = k;

// cout << t[ c ] << " "; //测试

c++;

}

t[ c ] = 1;

// cout << t[ c ] << " "; //测试

for ( int i = c, j = 1; i >= 1; i-- )

{

a[ j ] = t[ i ] \* pow( 2, static\_cast<double>( i - 1 ) );  **//2进制再如上面例子转化**

j++;

}

for ( int i = c; i >= 1; i-- )

if ( a[ i ] != 0 )

cout << a[ i ] << " ";

cout << "\n";

}

return 0;

}

**D.[Count](http://acm.buaa.edu.cn:8081/contest/169/problem/D/)**

**题目描述**

# Description

Thor老是分不清楚某种题目到底有没有出过以及出现了多少次。他实在是对数数不在行啊。  
得，你知道该怎么做了吧。

# Input

多组数据。   
每组数据有2行  
第一行1个整数n(1<=n<=1000)  
第二行n个整数x[i](x[i]在int范围内)

# Output

对于每组数据。  
以升序输出每个出现过的数及其出现过的次数。

# Sample Input

3

1 1 1

4

1 2 3 4

# Sample Output

1 3

1 1

2 1

3 1

4 1

# Hint

**解题思路**

**先把数组排序（可以用冒泡呀，不过有现成的sort函数）然后对每个数组中的数字判断是否和它下一个相等。不相等就输出相等计数器加一再判断。**

**参考代码**

#include <iostream>

#include <algorithm>

using namespace std;

int x[ 1005 ],n;

int main()

{

while ( cin >> n )

{

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

cin >> x[ i ];

sort( x, x + n ); //sort函数

int c = 0;

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

{

if ( x[ i ] != x[ i + 1 ] )

{

c++;

cout << x[ i ] << " " << c << endl;

c = 0;

}

else

c++;

}

}

return 0;

}

**E.**[**区间排序**](http://acm.buaa.edu.cn:8081/contest/169/problem/E/)

**题目描述**

# Description

有n个区间，给这n个区间从前往后排序并输出，若区间起点一样则比较终点。详情见样例。

# Input

多组数据。   
每组数据有n+1行。  
第一行有1个整数n(1<=n<=1000)  
接下来n行，每行两个整数A,B(A<=B)

# Output

输出排序之后的区间。

# Sample Input

3

1 4

1 1

2 2

2

1 2

1 2

# Sample Output

1 1

1 4

2 2

1 2

1 2

# Hint

**解题思路**

**两个数的话就用二维数组，第一维表示第几行，第二维（只需要开2）表示这一行的二个数字。可用swap函数实现冒泡排序使得程序简便。**

**参考代码**

#include <iostream>

using namespace std;

int a[ 1002 ][ 2 ];

int main()

{

int n;

while ( cin >> n )

{

for ( int i = 1; i <= n; i++ )

for ( int j = 0; j < 2; j++ )  **//用二维数组表示每行的两个数**

cin >> a[ i ][ j ];

for ( int i = 1; i <= n; i++ )  **//冒泡排序**

for ( int j = i + 1; j <= n; j++ )

{

if ( a[ i ][ 0 ] > a[ j ][ 0 ] )

{

swap( a[ i ][ 0 ], a[ j ][ 0 ] ); **//swap函数交换变量方便些**

swap( a[ i ][ 1 ], a[ j ][ 1 ] );

}

else if ( a[ i ][ 0 ] == a[ j ][ 0 ] && a[ i ][ 1 ] > a[ j ][ 1 ] )

swap( a[ i ][ 1 ], a[ j ][ 1 ] );

}

for ( int i = 1; i <= n; i++ )

{

for ( int j = 0; j < 2; j++ )

cout << a[ i ][ j ] << " ";

cout << "\n";

}

}

return 0;

}

**F.**[**苹果树**](http://acm.buaa.edu.cn:8081/contest/169/problem/F/)

**题目描述**

### Problem Description

还是苹果树•••说过两次了，晴天小猪有很多苹果树，树都种在他家的周围。现在我们用一个N\*N的图来表示晴天小猪家的状况，其中#为晴天小猪的家，0表示空地，1表示苹果树。已知晴天小猪的苹果树都种在和他家在同一斜线的位置上，你能数出他家总共有多少苹果树吗？

### Input

多组测试数据（数据量在100组以内）。每组测试数据第一行为一行为一个整数n（1<=n<=30），表示图的大小。接下来n行，每行n个字符。

### Output

对于每组输入数据，输出一行，为苹果树的总数。

### Sample Input

3

101

1#1

101

2

#1

11

### Sample Output

4

1

**解题思路**

**开始没细读题还以为图错了。。**

**先找到家，即#；**

**判断它4个方向上的所有的数字分别是否为1即可。**

**参考代码**

#include <iostream>

using namespace std;

char c[ 35 ][ 35 ];

int main()

{ int n;

while ( cin >> n )

{ int p, q, ans = 0;

for ( int i = 1; i <= n; i++ )

for ( int j = 1; j <= n; j++ )

{cin >> c[ i ][ j ];

if ( c[ i ][ j ] == '#' ) **//先找到家**

{p = i;q = j;}

}

for ( int i = p, j = q; i <= n && j <= n; i++, j++ ) **//四个方向上判断是否为1**

if ( c[ i ][ j ] == '1' )

ans++;

for ( int i = p, j = q; i <= n && j >= 1; i++, j-- )

if ( c[ i ][ j ] == '1' )

ans++;

for ( int i = p, j = q; i >= 1 && j <= n; i--, j++ )

if ( c[ i ][ j ] == '1' )

ans++;

for ( int i = p, j = q; i >= 1 && j >= 1; i--, j-- )

if ( c[ i ][ j ] == '1' )

ans++;

cout << ans << endl;

}

return 0;

}

**G.[strcmp](http://acm.buaa.edu.cn:8081/contest/169/problem/G/)**

**题目描述**

### Problem Description

依然是排序问题，不过这次排序的对象是字符串。  
设有两个序列X[0...k-1], Y[0...m-1]  
X的字典序小于Y，存在一个i<k使得, X[0..i-1] = Y[0..i-1] 且 X[i] < Y[i]  
若前k项都相等则长度长的定义为更大  
例如boat < boot < cap < card < cat < to < too< two < up  
请自己实现strcmp的功能。  
不允许使用strcmp函数及string类的小于(大于)比较符！否则此题0分！

### Input

多组数据，每组第一行一个数字n(n<=20)，接下来n行，每行一个字符串，长度小于100

### Output

按字典序输出n行，每行一个字符串。每2组输出用空行隔开。

### Sample Input

9

cap

to

cat

card

two

too

up

boat

boot

### Sample Output

boat

boot

cap

card

cat

to

too

two

up

**解题思路**

**和E题类似但是更难因为是字符串的排序。参照E题思路可做。**

**参考代码**

#include <iostream>

#include <string>

using namespace std;

string s[ 25 ];

int l[ 25 ];

int m( int, int );

int main()

{ int n;

while ( cin >> n )

{ for ( int i = 1; i <= n; i++ )

cin >> s[ i ];

for ( int i = 1; i <= n; i++ )

l[ i ] = s[ i ].length();

for ( int i = 1; i <= n; i++ )

for ( int j = i + 1; j <= n; j++ )

{if ( s[ i ][ 0 ] > s[ j ][ 0 ] )

{ swap( s[ i ], s[ j ] );

swap( l[ i ], l[ j ] );

}

else if ( s[ i ][ 0 ] == s[ j ][ 0 ] )

{int ll = m( l[ i ], l[ j ] ), c = 1, f = 0;

while ( c < ll )

{if ( s[ i ][ c ] > s[ j ][ c ] )

{swap( s[ i ], s[ j ] );

swap( l[ i ], l[ j ] );

f = 1;

break; }

else if ( s[ i ][ c ] < s[ j ][ c ] )

{ f = 1;

break;}

c++; }

if ( f == 0 )

if ( l[ i ] > l[ j ] )

{swap( s[ i ], s[ j ] );

swap( l[ i ], l[ j ] ); }

}

}

for ( int i = 1; i <= n; i++ )

cout << s[ i ] << endl;

cout << "\n";

}

return 0;

}

int m( int x, int y )

{

if ( x <= y )

return x;

else

return y;

}

**H.**[**数字填充**](http://acm.buaa.edu.cn:8081/contest/169/problem/H/)

**题目描述**

### Problem Description

晴天小猪喜欢玩数字游戏，但数独这样的游戏对他来说太难了，于是他准备玩一个容易点的游戏。游戏规则是在一个N\*N的表格里填数，规则只有一句话：总是以对角线为起点，先横着填，再竖着填。游戏给了一些样例，请在样例里面找到规律并把这个表格打印出来吧。

### Input

多组测试数据（数据量在100组以内）。每组测试数据只有一行为一个整数n（1<=n<=30），表示表格的大小。

### Output

对于每组输入数据，输出填完的表格（n行，每行n个整数，两两之间用空格隔开，注意不要在最后打印多余空格）。每两个表格之间用空行隔开，注意不要在第一行或者最后打印出多余的空行。

### Sample Input

3

5

### Sample Output

1 2 3

4 6 7

5 8 9

1 2 3 4 5

6 10 11 12 13

7 14 17 18 19

8 15 20 22 23

9 16 21 24 25

**解题思路**

**比上次上机的数字排序都简单。找到规律输出即可。**

**参考代码**

#include<iostream>

using namespace std;

main()

{

int n;

while(cin>>n)

{

for(int k=1;k<n;k++)

cout<<k<<" ";

cout<<n<<endl;

for(int i=2;i<=n;i++)

{

for(int j=1;j<=i-1;j++)

cout<<i+n-1+2\*n\*(j-1)-j\*j+1<<" ";

for(int j=i;j<n;j++)

cout<<i+n-1+2\*n\*(i-2)-(i-1)\*(i-1)+1+n+1-i+j-i<<" ";

cout<<i+n-1+2\*n\*(i-2)-(i-1)\*(i-1)+1+n+1-i+n-i<<endl;

}cout<<endl;

}

}

**I.[Ryan's ISBN](http://acm.buaa.edu.cn:8081/contest/169/problem/I/)**

**题目描述**

## Problem Description

每一本正式出版的图书都有一个ISBN号码与之对应。

ISBN码包括9位数字、1位识别码和3位分隔符，其规定格式如“x-xxx-xxxxx-x”。

其中符号“-”就是分隔符（键盘上的减号），最后一位是识别码，例如0-670-82162-4就是一个标准的ISBN码。

ISBN码的首位数字表示书籍的出版语言，例如0代表英语；

第一个分隔符“-”之后的三位数字代表出版社，例如670代表维京出版社；

第二个分隔符后的五位数字代表该书在该出版社的编号；

最后一位为识别码。

识别码的计算方法如下：

首位数字乘以1加上次位数字乘以2……以此类推，用所得的结果mod 11，所得的余数即为识别码，如果余数为10，则识别码为大写字母X。

例如ISBN号码0-670-82162-4中的识别码4是这样得到的：对067082162这9个数字，从左至右，分别乘以1，2，...,9,再求和，即0×1+6×2+……+2×9=158，然后取158 mod 11的结果4作为识别码。

你的任务是编写程序判断输入的ISBN号码中识别码是否正确，如果正确，则仅输出“Right”；

如果错误，则输出你认为是正确的ISBN号码。

## Input

多组测试数据。

对于每组测试数据，输入只有一行，是一个字符序列，表示一本书的ISBN号码（保证输入符合ISBN号码的格式要求）。

## Output

对于每组测试数据，输出共一行，假如输入的ISBN号码的识别码正确，那么输出“Right”，否则，按照规定的格式，输出正确的ISBN号码（包括分隔符“-”）。

## SampleInput

0-670-82162-4

0-670-82162-0

## SampleOutput

Right

0-670-82162-4

**解题思路**

**按照题目意思，正常计算即可。注意string的使用。**

**参考代码**

#include <iostream>

#include <string>

using namespace std;

int main()

{ string stri;

while ( cin >> stri )

{ int num = 0, ans = 0;

for ( int i = 0; i <= 10; i++ )

{if ( i != 1 && i != 5 )  **//注意除去-**

{ num++;

ans += ( stri[ i ] - '0' ) \* num;}}

ans = ans % 11;

if ( ans == 10 && stri[ 12 ] != 'X' )  **//X**

{ stri[ 12 ] = 'X';

cout << stri <<endl;}

else if ( ans == 10 && stri[ 12 ] == 'X' )

cout << "Right" << endl;

else if ( ans == stri[ 12 ] - '0' )

cout << "Right" << endl;

else

{for ( int i = 0; i <= 11; i++ )

cout << stri[ i ];

cout << ans << endl;

}

}

return 0;

}

**J.[Lost in maze](http://acm.buaa.edu.cn:8081/contest/169/problem/J/)**

## Problem Description

Ryan的梦中情人叫做小美。

有一天小美被妖怪抓走了，Ryan听闻马上动身要去上演英雄救美的壮举。

可是当他来到妖怪的领地时，发现妖怪给自己设了一个阵，只有成功破解出这个阵，Ryan才能救出他心爱的小美。

所以请你帮Ryan判断出他能不能救出小美。

## Input

多组测试数据。

对于每组数据，第一行是两个个数字m和n，表示该阵为m\*n大小（0<m,n<=10，m为行，n为列）。

接下来是一个m\*n的矩阵表示这个阵。

其中0为该阵中可走的路，1为墙（即不能走的位置），2为Ryan的位置，3为小美的位置。

Ryan只有四个可行的行走方向：即上、下、左、右。

## Output

每组数据共输出一行，"Yes"或"No"表示Ryan是否可以成功救出小美。

## Sample Input

8 7

1 1 1 1 1 1 3

1 0 0 0 0 0 0

1 0 1 1 1 1 1

1 0 0 0 1 1 1

1 1 1 0 0 0 1

1 0 0 0 1 0 1

1 0 1 1 1 0 1

2 0 1 1 1 0 0

4 4

0 3 1 2

1 1 1 0

0 0 1 0

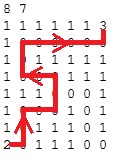
0 0 0 0

## Sample Output

Yes

No

## Hint

样例中Ryan成功解救小美的路线图：  


**解题思路**

**我不会说这是书上的练习题，不看书的同学多看看书吧。301页走迷宫问题。**

**参考代码**

#include <cstdio>

#include <cstring>

using namespace std;

const int yx[ 4 ] = { 1, -1, 0, 0 };

const int yy[ 4 ] = { 0, 0, 1, -1 };

int map[ 11 ][ 11 ];

int n,m;

int zx,zy;

bool ok = 0;

void dfs(int ,int );

int main()

{

while ( ~scanf( "%d%d", &n, &m ) )

{

ok = 0;

for (int i = 1; i <= n; i++)

for (int j = 1; j <= m; j++)

{

scanf( "%d", &map[ i ][ j ] );

if ( map[ i ][ j ] == 2 )

{

zx = i;

zy = j;

}

}

dfs( zx, zy );

if ( ok )

printf( "Yes\n" );

else

printf( "No\n" );

}

}

void dfs( int x, int y )

{

if ( ok == 1 )

return;

for ( int i = 0; i < 4; i++ )

if ( x + yx[ i ] > 0 && x + yx[ i ] <= n && y + yy[ i ] > 0 && y + yy[ i ] <= m && map[ x + yx[ i ] ][ y + yy[ i ] ] != 1 )

{

if( map[ x + yx[ i ] ][ y + yy[ i ] ] == 3 )

{

ok = 1;

break;

}

map[ x + yx[ i ] ][ y + yy[ i ] ] = 1;

dfs( x + yx[ i ], y + yy[ i ] );

map[ x + yx[ i ] ][ y + yy[ i ] ] = 0;

}

}