谢字希

乐于总结，乐于开源

https://github.com/XieZixiUSTC/code1024

C printf getchar a
注意牛客网的输入和输出 和力扣的不一样

1.输入整数，位运算(每次都往右移动，比较n&1)

题目描述

输入一个 int 型的正整数，计算出该 int 型数据在内存中存**储时 1 的个数。**

输入描述:
输入一个整数（int 类型）
输出描述:
这个数转换成 2 进制后，输出 1 的个数

示例 1

输入

复制

```
5
```

输出

复制

2

```
#include<iostream>
using namespace std;
int main(){
    int n;
    while(cin>>n){
        int m=0;
        while(n!=0){
            if(n&1==1){
                m++;
            }
            n=n>>1;
        }
        cout<<m<<endl;
    }
}
```

2.1cin>>的用法

cin 可以连续从键盘读取想要的数据，**以空格、tab 或换行作为分隔符。**实例程序如下。

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

int main()
{
	char a;
	int b;
	float c;
	string
	cin>>a>>b>>c;
	cout<<a<<" "<<b<<" "<<c<<" "<<endl;

	system("pause");
	return 0;
}
```

2. 浮点数(直接每个加 0.5)

题目描述

写出一个程序，接受一个正浮点数值，输出该数值的近似整数值。如果小数点后数值大于等于 5,向上取整；小于 5，则向下取整。

输入描述:
输入一个正浮点数值
输出描述:
输出该数值的近似整数值

示例 1

输入

复制

```
5.5
```

输出

复制

```
6
```

```
#include<iostream>
#include<math.h>
using namespace std;
int main(){
	float f;
```

```
    while(cin>>f){
        cout<<int(f+0.5)<<endl;//一直向下取整
         //cout<<ceil(f)<<endl;
    }
    return 0;
}
```

3. 最小公倍数（每次找公约数，并且 ab 要整除 l,c 要乘以 i，最后 c*a*b）

题目描述

正整数 A 和正整数 B 的最小公倍数是指 能被 A 和 B 整除的最小的正整数值，设计一个算法，

求输入 A 和 B 的最小公倍数。

输入描述：
输入两个正整数 A 和 B。
输出描述：
输出 A 和 B 的最小公倍数。

示例 1

输入

复制

```
5 7
```

输出

复制

35

```
#include<iostream>
using namespace std;
int main()
{
   int a,b,c=1;
   cin>>a>>b;
   for(int i=2;i<=a&&i<=b;i++)
   {    if(a%i==0&&b%i==0)
       {
          a/=i;//往下除以一个两个的公约数
          b/=i;
          c*=i;
       }
   }
     c=c*a*b;
   cout<<c;
}
```

4. 数字颠倒(%10 每个/10 每次变化)

题目描述

输入一个整数，将这个整数以字符串的形式逆序输出

程序不考虑负数的情况，若数字含有 0，则逆序形式也含有 0，如输入为 100，则输出为 001

输入描述：
输入一个 `int` 整数
输出描述：
将这个整数以字符串的形式逆序输出

示例 1

输入

复制

```
1516000
```

输出

复制

```
0006151
```

```
#include<iostream>
#include<algorithm>
#include<string>
using namespace std;
int main()
{
	int a,i,k;
	cin>>a;
	while(a)
	{
		cout<<a%10;
		a=a/10;
	}
	return 0;
}
```

5. 字符串反转（直接 reverse）

题目描述

接受一个只包含小写字母的字符串，然后输出该字符串反转后的字符串。（字符串长度不超过 1000）

输入描述：
输入一行，为一个只包含小写字母的字符串。
输出描述：
输出该字符串反转后的字符串。

示例 1

输入

复制

```
abcd
```

输出

复制

```
Dcba
```

```
#include<iostream>
#include<algorithm>
using namespace std;
int main(){
        string str;
    while(cin>>str){
            reverse(str.begin(),str.end());
                    cout<<str<<endl;
    }
     return 0;
}
```

6.汽水瓶（喝几瓶，还有几个瓶子，总共有几个瓶子，最后有两个要+1）

题目描述

有这样一道智力题：“某商店规定：三个空汽水瓶可以换一瓶汽水。小张手上有十个空汽水瓶，她最多可以换多少瓶汽水喝？”答案是 5 瓶，方法如下：先用 9 个空瓶子换 3 瓶汽水，喝掉 3 瓶满的，喝完以后 4 个空瓶子，用 3 个再换一瓶，喝掉这瓶满的，这时候剩 2 个空瓶子。然后你让老板先借给你一瓶汽水，喝掉这瓶满的，喝完以后用 3 个空瓶子换一瓶满的还给老板。如果小张手上有 n 个空汽水瓶，最多可以换多少瓶汽水喝？

输入描述:
输入文件最多包含 10 组测试数据，每个数据占一行，仅包含一个正整数 n（1<=n<=100），表示小张手上的空汽水瓶数。n=0 表示输入结束，你的程序不应当处理这一行。
输出描述:
对于每组测试数据，输出一行，表示最多可以喝的汽水瓶数。如果一瓶也喝不到，输出 0。

示例 1

输入

复制

```
3

10

81
```

```
0
```

输出

复制

```
1

5

40
```

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



int main()
{
	int n;
	while (cin>>n)
	{
		if (n == 0)break;
		int ans=0;
		while (n >= 3)
		{
			int a = n / 3;
			int b = n % 3;
			ans += a;
			n = a + b;
		}
		if (n == 2) ans += 1;//还可以借一瓶
		cout << ans<<endl;
	}
}
```

7. 统计每个月的兔子总数（动态规划，熟兔子，两个月兔子，一个月兔子，--month，先减在判断）

题目描述

有一只兔子，从出生后第 **3 个月起每个月都生一只兔子**，小兔子长到第**三个月后每个月又生一只兔子，假如兔子都不死，问每个月的兔子总数为多少?**

本题有多组数据。

输入描述:
输入 int 型表示 **month**
输出描述:
输出兔子总数 int 型

示例 1

输入

复制

```
9
```

输出

复制

34

```
#include <iostream>
using namespace std;
int main()
{
	int month;
	while(cin >> month)
	{
		int shu3 = 0;//成熟了的可以生兔子的兔子数量，即成熟度是 3 及以上的
		int shu1 = 1;//新生的成熟度为 1 的兔子数量
		int shu2 = 0;//差一个月就成熟的成熟度为 2 的兔子数量

		//这里一定是--month
		//因为初始三个值已经是第一个月的数了，所以循环少一个月
		while(--month)//先减在判断
		{
			shu3 += shu2;//之前熟了的兔子加上两个月熟的兔子就是所有熟兔子
			shu2 = shu1;//两个月的成熟度的兔子都是新生兔子变的
			shu1 = shu3;//新生兔子都是成熟了的兔子生的
		}
		cout << shu1 + shu2 + shu3 << endl;
	}
	return 0;
}
```

8. 四则运算（用一个栈，来表示，flag，就是括号，数字之后的。注意递归）

题目描述

输入一个表达式（用字符串表示），求这个表达式的值。

保证字符串中的有效字符包括

[‘0’ - ‘9’], ‘+’ , ‘-’ , ‘*’ , ‘/’ , ‘(’ , ‘)’ , ‘[’ , ‘]’ , ‘{’ , ‘}’ 。且表达式

一定合法。

输入描述:
输入一个算术表达式
输出描述:
得到计算结果

示例 1

输入

复制

```
3+2*{1+2*[-4/(8-6)+7]}
```

输出

复制

```
25
```

四则运算，递归解法分析，最简洁的代码

大佬的递归解法，也可以称为消消乐解法，这是我见过最简洁的表达式求值代码。看到这种解法之后我都不想去看什么逆波兰了。。。首先声明，我只是个搬运工。

第一步，先**考虑无括号的情况，先乘除后加减，这个用栈很容易解**决，遇到数字先压栈，如果下一个是乘号或除号，先出栈，和下一个数进行乘除运算，再入栈，最后就能保证栈内所有数字都是加数，最后对所有加数求和即可。

第二步，遇到左括号，直接递归执行第一步即可，最后检测到右括号，**返回括号内的计算结果，退出函数，这个结果作为一个加数，返回上一层入栈。**

```
#include <iostream>
#include <stack>

using namespace std;

int pos;//全局变量

int compute(string & data)
{
	int len = data.length();
	int num = 0;
	char flag = '+';
	stack<int> st;

	while (pos < len) {
		if (data[pos] == '{' || data[pos] == '[' || data[pos] == '(') {
			pos++;
			num = compute(data);
		}

		while (pos < len && isdigit(data[pos])) {//将数字拿出来
			num = num * 10 + data[pos] -'0';
			pos++;
		}
```

```
            switch (flag) {
            case '+':
                st.push(num);
                break;
            case '-':
                st.push(-num);
                break;记得有个 break
            case '*':
                {
                    int temp = st.top();
                    temp *= num;
                    st.pop();
                    st.push(temp);
                    break;
                }
            case '/':
                {
                    int temp = st.top();
                    temp /= num;
                    st.pop();
                    st.push(temp);
                    break;
                }
            }

            num = 0;//复原
            flag = data[pos];//数字之后是符号啊。
            if (data[pos] == '}' || data[pos] == ']'|| data[pos] == ')') {
                pos++;
                break;
            }
            pos++;//下一个啊
        }

        int sum = 0;
        while (st.size()) {
            sum += st.top();
            st.pop();
        }
        return sum;
}

int main()
{
```

```
    string data;

    while (cin >> data) {
        pos = 0;
        cout << compute(data) << endl;
    }
    return 0;
}
```

9. 杨辉三角变形（找规律，奇数和偶数）

题目描述

1

1 1 1

1 2 3 2 1

1 3 6 7 6 3 1

1 4 10 16 19 16 10 4 1

以上三角形的数阵，第一行只有一个数 1，以下每行的每个数，是恰好是它上面的数，左上角数到右上角的数，3 个数之和（如果不存在某个数，认为该数就是 0）。

求第 n 行第一个偶数出现的位置。如果没有偶数，则输出-1。例如输入 3,则输出 2，输入 4 则输出 3。

输入 n(n <= 1000000000)

本题有多组输入数据，输入到文件末尾，请使用 while(cin>>)等方式读入

输入描述:
输入一个 `int` 整数
输出描述:
输出返回的 `int` 值

示例 1

输入

复制

```
4

2
```

输出

复制

```
3

-1
```

题目描述

1

1 1 1

1 2 3 2 1

1 3 6 7 6 3 1

1 4 10 16 19 16 10 4 1

以上三角形的数阵，第一行只有一个数 1，以下每行的每个数，是恰好是它上面的数，左上角数到右上角的数，3 个数之和（如果不存在某个数，认为该数就是 0）。

请在这里输入引用内容

求第 n 行第一个偶数出现的位置。如果没有偶数，则输出-1。例如输入 3，则输出 2，输入 4 则输出 3。

当 n<3 时，没有偶数，输出-1；

当 n 为奇数时，**第一个偶数位置在第二，输出 2；**

当 n 为偶数且能被 4 整除时，**第一个偶数位置在第三，输出 3；**

当 n 为偶数但不能被 4 整除时，**偶数位置在第四，输出 4**

```
#include<iostream>
using namespace std;
int main(){
    int num;
    while(cin>>num){
        if(!num||num==1||num==2) cout<<-1<<endl;
        else if(num&1) cout<<2<<endl;//奇数第二个
        else if(num%4) cout<<4<<endl;//偶数，不被 4 整除，第四个
        else cout<<3<<endl;//偶数被 4 整除，第三个
    }
    return 0;
}
```

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

int main(){
    int a;
    while(cin>>a){
        if(a==1||a==2){
            cout<<-1<<endl;
        }
        else{
            if(a%2==1){
            cout<<2<<endl;
            }
            else if(a%4==0){
                cout<<3<<endl;
            }
            else{
                cout<<4<<endl;
            }
        }
    }
    return 0;
}
```

10. 表达式求值

题目描述

给定一个字符串描述的算术表达式，计算出结果值。

输入字符串长度不超过 100，合法的字符包括”+, -, *, /, (,)”，”0-9”，字符串内容的合法性及表达式语法的合法性由做题者检查。本题目只涉及整型计算。

输入描述:
输入算术表达式
输出描述:
计算出结果值

示例 1

输入

复制

```
400+5
```

输出

复制

```
405
```

```
#include <iostream>
#include <stack>

using namespace std;

int pos;

int compute(string & data)
{
	int len = data.length();
	int num = 0;
	char flag = '+';
	stack<int> st;

	while (pos < len) {
		if (data[pos] == '{' || data[pos] == '[' || data[pos] == '(') {
			pos++;
			num = compute(data);
		}

		while (pos < len && isdigit(data[pos])) {//将数字拿出来
			num = num * 10 + data[pos] -'0';
			pos++;
		}

		switch (flag) {
		case '+':
			st.push(num);
			break;
		case '-':
			st.push(-num);
			break;
		case '*':
			{
				int temp = st.top();
				temp *= num;
				st.pop();
				st.push(temp);
				break;
			}
		case '/':
			{
				int temp = st.top();
				temp /= num;
```

```
                st.pop();
                st.push(temp);
                break;
            }
        }

        num = 0;
        flag = data[pos];//数字之后是符号，如果是后括号呢，就返回了，会继续寻找下一个符号。
        if (data[pos] == '}' || data[pos] == ']'|| data[pos] == ')') {
            pos++;
            break;
        }
        pos++;//下一个啊
    }

    int sum = 0;
    while (st.size()) {
        sum += st.top();
        st.pop();
    }
    return sum;
}

int main()
{
    string data;

    while (cin >> data) {
        pos = 0;
        cout << compute(data) << endl;
    }
    return 0;
}
```

和上面一样，只要知道模板就行了。

11. 完全数计算（6,28,496,8128）

题目描述

完全数（Perfect number），又称完美数或完备数，是一些特殊的自然数。

它所有的真因子（即除了自身以外**的约数）的和（即因子函数）**，恰好等于它本身。

例如：28，它有约数 1、2、4、7、14、28，除去它本身 28 外，其余 5 个数相加，

1+2+4+7+14=28。s

输入 n，请输出 n 以内(含 n)完全数的个数。计算范围, 0 < n <= 500000

本题输入含有多组样例。

输入描述:
输入一个数字 n
输出描述:
输出不超过 **n** 的完全数的个数

示例 1

输入

复制

```
1000

7

100
```

输出

复制

```
3

1

2
```

```
#include <iostream>
using namespace std;
int main(){
    int n;
    while(cin >> n){
        if(n < 6)
            cout << 0 << endl;
        else if(n < 28)
            cout << 1 << endl;
        else if(n < 496)
            cout << 2 << endl;
        else if(n < 8128)
            cout << 3 << endl;
        else
            cout << 4 << endl;
    }
    return 0;
}
```

如何找到未知的完全数

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    while(cin>>n)
    {
        int count=0;
        for(int i=6;i<=n;i++)
        {
            int temp=0;
            for(int j=1;j<i;j++)
```

```
            {//注意 j<n，因为求和时不算这个数的自身
                if(i%j==0)
                    temp+=j;
            }
            if(temp==i)
                count++;
        }
        cout << count << endl;
    }
}
```

12. 字符逆序（getline cin str 直接 reverse）

题目描述

将一个字符串 str 的内容颠倒过来，并输出。str 的长度不超过 100 个字符。

输入描述:
输入一个字符串，可以有空格
输出描述:
输出逆序的字符串

示例 1

输入

复制

```
I am a student
```

输出

复制

```
tneduts a ma I
```

```
#include<iostream>
#include<string>
#include<algorithm>

using namespace std;

#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <stack>
#include <vector>

using namespace std;

int main()
{
	string s;

	while (getline(cin, s))//直接逆序
	{
		reverse(s.begin(), s.end());
		cout << s << endl;
	}
	return 0;
}


#include <iostream>
#include <string>
using namespace std;
int main(){
	string str;
	getline(cin,str);
	for(int i = str.size()-1;i>=0;i--)
		cout << str[i];
}
```

13. 等差数列 a1+a1+(n-1)*d)）

题目描述

功能:等差数列 2，5，8，11，14。。。。

输入:正整数 N >0

输出:**求等差数列前 N 项和**

本题为多组输入，请使用 while(cin>>)等形式读取数据

输入描述:
输入一个正整数。
输出描述:
输出一个相加后的整数。

示例 1

输入

复制

```
2
```

输出

复制

```
7
```

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

void sum(int n)
{
	int Sn=(3*n+1)*n/2;
	cout<<Sn<<endl;
}

int main()
{
	int n;
	while(cin>>n)
	{
		sum(n);
	}
}
```

14. 走方格的方案数（注意是格子和线,用 dfs，每次 cnt=0）

题目描述

请计算 n*m 的棋盘格子（n 为横向的格子数，m 为竖向的格子数）沿着各自**边缘线从左上角走到右下角**，总共有多少种走法，要求不能走回头路，即：**只能往右和往下走**，不能往左和往上走。

本题含有多组样例输入。

输入描述:
每组样例输入两个正整数 n 和 m，用空格隔开。(1≤n,m≤8)
输出描述:

每组样例输出一行结果

示例 1

输入

复制

```
2 2

1 2
```

输出

复制

```
6

3
```

```
#include<iostream>
using namespace std;
int sum(int m,int n){
	int k;
	if(m==0||n==0){
		k=1;
	}else{
		k=sum(m-1,n)+sum(m,n-1);//用递归的解法
	}
	return k;
}
int main(){
	int m,n;
```

```
    while(cin>>m>>n){
        int num=sum(m,n);
        cout<<num<<endl;
    }
}
```

用 dfs 递归

```
#include<iostream>
using namespace std;
int cnt;
int m, n;//直接就是全局了呗
void dfs(int row, int col)
{
    if (row == m && col == n )/*终止条件，找到设定点*/
    {
        cnt++;
        return;
    }
    if (row <m+1 && col <n+1)/*递归条件，不能超出边界，注意 m*n 其实划线来看是有（m+1）*(n+1)个数*/索引可以取到 m/n
    {
        dfs(row + 1, col);
        dfs(row , col+1);
    }
}

int main()
{
    while (cin >> m >> n)
    {
        dfs(0, 0);
        cout << cnt << endl;
        cnt = 0;//还原
    }
}
```

15. 密码强度等级(拿到大小写，字符个数，数字个数)

题目描述

密码按如下规则进行计分，并根据不同的得分为密码进行安全等级划分。

一、密码长度:

5 分: 小于等于 4 个字符

10 分: 5 到 7 字符

25 分: 大于等于 8 个字符

二、字母:

0 分: 没有字母

10 分: 全都是小（大）写字母

20 分: 大小写混合字母

三、数字:

0 分: 没有数字

10 分: 1 个数字

20 分: 大于 1 个数字

四、符号:

0 分: 没有符号

10 分: 1 个符号

25 分: 大于 1 个符号

五、奖励:

2 分: 字母和数字

3 分: 字母、数字和符号

5 分: 大小写字母、数字和符号

最后的评分标准:

>= 90: 非常安全

>= 80: 安全（Secure）

>= 70: 非常强

>= 60: 强（Strong）

>= 50: 一般（Average）

>= 25: 弱（Weak）

>= 0: 非常弱

对应输出为：

VERY_SECURE

SECURE,

VERY_STRONG,

STRONG,

AVERAGE,

WEAK,

VERY_WEAK,

请根据输入的密码字符串，进行安全评定。

注：

字母：a-z, A-Z

数字：-9

符号包含如下： (ASCII 码表可以在 UltraEdit 的菜单 view->ASCII Table 查看)

!"#$%&'()*+,-./ (ASCII 码：x21~0x2F)

:;<=>?@ (ASCII<=><=><=><=><=>码：x3A~0x40)

[\]^_` (ASCII 码：x5B~0x60)

{|}~ (ASCII 码：x7B~0x7E)

输入描述：
本题含有多组输入样例。
每组样例输入一个 `string` 的密码
输出描述：
每组样例输出密码等级

示例 1

输入

复制

```
38$@NoNoNo
```

```
123
```

输出

复制

```
VERY_SECURE
```

```
WEAK
```

说明

```
第一组样例密码强度为 95 分。
```

```
第二组样例密码强度为 25 分。
```

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string password;
    while(getline(cin,password))
    {
        int score=0;
```

```
//密码长度得分
if(password.size()<=4){
	score+=5;
}else if(password.size()>=8){
	score+=25;
}else{
	score+=10;
}
//字母、数字、符号统计
int Alpha=0,alpha=0,number=0,number_count=0,ch=0,ch_count=0;
for(int i=0;i<password.size();i++)
{
	if(password[i]>='a' && password[i]<='z'){
		alpha=1;
	}else if(password[i]>='A' && password[i]<='Z'){
		Alpha=1;
	}else if(isdigit(password[i])){
		number=1;//有数字
		number_count++;
	}else{
		ch=1;//有字符
		ch_count++;//有几个字符
	}
}
//字母得分
if((alpha==1&&Alpha==0) || (alpha==0&&Alpha==1)){
		score+=10;
}else if(alpha==1 && Alpha==1){//有大小写
		score+=20;
}
//数字得分
if(number_count>1){
	score+=20;
}else if(number){
	score+=10;
}
//符号得分
if(ch_count>1){
	score+=25;
}else if(ch){
	score+=10;
}
//奖励得分
if(Alpha && alpha && number && ch){//奖励
```

```
            score+=5;
        }else if((Alpha||alpha)&& number && ch){
            score+=3;
        }else if((Alpha||alpha)&& number){
            score+=2;
        }
        if(score>=90){
            cout<<"VERY_SECURE"<<endl;
        }else if(score>=80){
            cout<<"SECURE"<<endl;
        }else if(score>=70){
            cout<<"VERY_STRONG"<<endl;
        }else if(score>=60){
            cout<<"STRONG"<<endl;
        }else if(score>=50){
            cout<<"AVERAGE"<<endl;
        }else if(score>=25){
            cout<<"WEAK"<<endl;
        }else{
            cout<<"VERY_WEAK"<<endl;
        }
    }
    return 0;
}
```

16. 放苹果（动态规划，苹果全部放，或者一个空碗，注意递归条件）

题目描述

题目描述

把 m 个同样的苹果放在 n 个同样的盘子里，允许有的盘子空着不放，问共有多少种不同的分法？（用 K 表示）5，1，1 和 1，5，1 是同一种分法。

数据范围：0<=m<=10，1<=n<=10。

本题含有多组样例输入。

输入描述:
输入两个 int 整数
输出描述:
输出结果，int 型

示例 1

输入

复制

```
7 3
```

输出

复制

```
8
```

放苹果分为两种情况，一种是有**盘子为空**，一种是**每个盘子上都有苹果。**
令(m,n)表示将 m 个苹果放入 n 个盘子中的摆放方法总数。
1.假设有一个盘子为空，则(m,n)问题转化为将 m 个苹果放在 n-1 个盘子上，即求得(m,n-1)即可
2.假设所有盘子都装有苹果，则每个盘子上至少有一个苹果，即**最多剩下 m-n 个苹果**，问题转化为将 m-n 个苹果放到 n 个盘子上
即求(m-n，n)

```
#include <iostream>

using namespace std;

int putApples(int M, int N){
    if (M < 0)  // M 的值在递归时为(M-N)决定，因此 M < 0 时为 0
        return 0;
    if (M == 0 || M == 1) // 在 0 个苹果或者 1 个苹果的条件下，都只有一种放法
```

```
        return 1;
    if (N == 1) // N从一个>=1的数递减，因此N == 1可以作为递归出口
        return 1;
    return putApples(M-N, N) + putApples(M, N-1);
}

int main(){
    int N, M;
    while(cin >> M >> N){
        if((0 <= M && M <= 10) && (1 <= N && N <= 10))
            cout << putApples(M, N) << endl;
        else
            cout << "-1" << endl;
    }
    return 0;
}
'''
```

17. 二进制下 1 的个数

题目描述

输入一个正整数，计算它在二进制下的 1 的个数。

注意多组输入输出！！！！！！

输入描述:
输入一个整数
输出描述:
计算整数二进制中 1 的个数

示例 1

输入

复制

```
5
```

输出

复制

```
2
```

说明

```
5 的二进制表示是 101，有 2 个 1
```

```
#include<iostream>
using namespace std;
int main()
{
	int num;
	while (cin >> num)
	{
		int count = 0;
		while (num)
		{
			num = (num - 1)&num;//每次都减去 1，右移一位。每次都减 1
			count++;
		}
		cout << count<<endl;
	}
	return 0;
}
```

```
#include<iostream>
```

```
using namespace std;
int main(){
    int n;
    while(cin>>n){
        int m=0;
        while(n!=0){
            if(n&1==1){
                m++;
            }
            n=n>>1;
        }
        cout<<m<<endl;
    }
}
```

18. 配置文献恢复（一个 find 在 0 中找）

题目描述

有 6 条配置命令，它们执行的结果分别是：

命　　令	执　　行
reset	reset what
reset board	board fault
board add	where to add
board delete	no board at all
reboot backplane	impossible
backplane abort	install first

he he unknown command

注意：he he 不是命令。

为了简化输入，方便用户，以“最短唯一匹配原则”匹配：

1、若只输入一字串，则只匹配一个关键字的命令行。例如输入：r，根据该规则，匹配命令 reset，执行结果为：**reset what；输入：res，根据该规则，匹配命令 reset**，执行结果为：reset what;

2、若只输入一字串，但本条命令有两个关键字，则匹配失败。例如输入：reb，可以找到命令 reboot backpalne，但是该命令有两个关键词，**所有匹配失败，执行结果为：**unknown command

3、若输入两字串，则先匹配第一关键字，如果有匹配但不唯一，继续匹配第二关键字，如果仍不唯一，匹配失败。例如输入：r b，找到匹配命令 **reset board 和 reboot backplan**e，执行结果为：unknown command。

4、若输入两字串，则先匹配第一关键字，如果有匹配但不唯一，继续匹配第二关键字，如果唯一，匹配成功。例如输入：b a，无法确定是命令 **board add** 还是 **backplane abort**，匹配失败。

5、若输入两字串，第一关键字匹配成功，则匹配第二关键字，若无匹配，失败。例如输入：bo a，确定是命令 **board add**，匹配成功。

6、若匹配失败，打印“unknown command”

输入描述：
多行字符串，每行字符串一条命令
输出描述：

执行结果，每条命令输出一行

示例 1

输入

复制

```
reset

reset board

board add

board delet

reboot backplane

backplane abort
```

输出

复制

```
reset what

board fault

where to add

no board at all

impossible

install first
```

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;
bool match(string str, string s) {
    return str.find(s) == 0;
}
int main() {
    string str;
    string cmd[6] = {"reset",          "reset board", "board add",   "board delete",
"reboot backplane", "backplane abort" };
    string res[7] = {"reset what",  "board fault", "where to add",    "no   board   at   all",
"impossible",       "install first",   "unknown command" };
    while (getline(cin, str)) {
        string s1, s2, temp;
        stringstream ss(str);
        ss >> s1 >> s2;//第二个没有就是空
        int resi = 6;
        if (s2.empty())
            resi = match(cmd[0], s1) ? 0 : 6;//第一个只能 res 匹配到，reset,没有就是
unknow
        else {
            bool flag = false;
            for (int i = 1; i < 6; i++) {
                stringstream allcmd(cmd[i]);
                allcmd >> temp >> temp;//第二个 cmd 是 tmp
                if (match(cmd[i], s1) && match(temp, s2)) {//第一个匹配是总的
```

```
                            if (!flag) {
                                flag = true;
                                resi = i;//就匹配一次，如果两个后面还有匹配，就是
unknow，歧义
                            } else {
                                resi = 6;
                                break;
                            }
                        }
                    }
                }
                cout << res[resi] << endl;
            }
            return 0;
        }//大家的都好长，我来个短一点的吧。。。。
```

19. 百钱买鸡问题（百钱，百鸡，两个循环，剩余的钱，得到鸡的数量是否 100）

题目描述

公元前五世纪，我国古代数学家张丘建在《算经》一书中提出了“百鸡问题”：鸡翁一值钱五，鸡母一值钱三，鸡雏三值钱一。百钱买百鸡，**问鸡翁、鸡母、鸡雏各几何**？

详细描述：

接口说明

原型：

int GetResult(vector &list)

输入参数：

无

输出参数（指针指向的内存区域保证有效）：

list 鸡翁、鸡母、鸡雏组合的列表

返回值：

-1 失败

0 成功

输入描述：
输入任何一个整数，即可运行程序。
输出描述：

示例 1

输入

复制

```
1
```

输出

复制

```
0 25 75

4 18 78

8 11 81

12 4 84
```

利用数学思维解决问题 虽然直接三层 for 循环也可解决问题，但效率太低，因为里面有很多答案是不符合实际情况的。找到他们之间的绝对数学关系即可快速拿到准确的答案。

5 31

复制代码

```
#include <iostream>
/*
 鸡翁 a 只(0≤a≤20) 鸡母 b 只(0≤b≤33) 鸡雏(100-a-b)只 由 5a+3b+(100-a-b)/3=100
 得到关系式 b=25-7a/4，也就是说鸡翁数量是 4 的倍数，最少 0 只 做多 20 只
 设一个变量 num(0≤num≤3，≤3 是要保证鸡母数不为负)
 则鸡翁 4num 只，鸡母为 25-7num 只，鸡雏为 75+3num 只
*/
int main()
{
    for(int num=0;num<=3;num++) // 只需要执行 num=0 1 2 3 时的 4 次 for 循环
        std::cout<<4*num<<" "<<25-7*num<<" "<<75+3*num<<std::endl;
}
```

收起

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

```
int main() {
	int a,b,c;
	int surplus = 0;
	for(a = 0;a <= 25;a++)
	{
		for(b = 0;b <= 33;b++)
		{
			surplus = 100 - a*5 - b*3;
			if(surplus <= 0)
				break;
			c = surplus*3;//小鸡三个才值一钱
			if(a+b+c == 100)//数量等于 100
			{
				cout<< a <<" "<< b <<" "<< c <<endl;
				break;
			}

		}
	}
}
```

20. 日期到天数的替换（100 的倍数，能否被 4 整除，被 4 整除，闰年）

题目描述

根据输入的日期，计算是这一年的第几天。。

测试用例有多组，注意循环输入

输入描述:
输入多行，每行空格分割，分别是年，月，日
输出描述:
成功:返回 outDay 输出计算后的第几天;
失败:返回-1

示例 1

输入

复制

```
2012 12 31
```

输出

复制

```
366
```

```
#include <iostream>
#include <vector>
#include <sstream>
#include <algorithm>
using namespace std;
const int days[12] = {
    31,28,31,30,31,30,31,31,30,31,30,31
};
int main()
{
    int year, month, day;
    bool flag;
    while(cin >> year >> month >> day)
    {
        if(year%100==0)
        {
            flag = (year%400==0);//是 100 的倍数，并且是 400
        }
        else
        {
            flag = (year%4==0);//或者被 4 整除
        }

        int n = 0;
```

```
            for(int i=1; i<month; i++)
            {
                n += days[i-1];
                if(i==2 && flag)//如果是二月，还要加 1,
                {
                    n++;
                }
            }
            n+=day;//加上几天,输入年月日表示过了几天
            cout << n << endl;
        }
        return 0;
    }
```

21. 参数解析（是空，并且 temp 是否为空）

题目描述

在命令行输入如下命令：

xcopy /s c:\ d:\,

各个参数如下：

参数 1：命令字 xcopy

参数 2：字符串/s

参数 3：字符串 c:\

参数 4: 字符串 d:\

请编写一个参数解析程序，实现将命令行各个参数解析出来。

解析规则：

1.参数分隔符为空格

2.对于用""包含起来的参数，如果中间有空格，不能解析为多个参数。比如在命令行输入

xcopy /s "C:\program files" "d:\"时，参数仍然是 4 个，第 3 个参数应该是字符串

C:\program files，而不是 C:\program，注意输出参数时，需要将""去掉，引号不存在嵌

套情况。

3.参数不定长

4.输入由用例保证，不会出现不符合要求的输入

输入描述：
输入一行字符串，可以有空格
输出描述：
输出参数个数，分解后的参数，每个参数都独占一行

示例 1

输入

复制

```
xcopy /s c:\\ d:\\
```

输出

复制

```
4

xcopy

/s

c:\\

d:\\
```

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

int main ()
{
	string input;
	while( getline(cin,input))//cin 给了 input
	{
		vector<string> ans;
		string tmp = "";
		string order = "";
		for(int i = 0; i< input.size(); i++)
		{
			if(input[i] == ' ')//遇到空格处理之前保存的
			{
				if(!tmp.empty())
					ans.push_back(tmp);//将这个字符丢进来
				tmp.clear();//有个 clear
			}
			else if(input[i] == '"')//遇到引号，处理这部分
			{
				i++;
				while(input[i]!='"')
				{
					tmp.push_back(input[i]);//这个字符
					i++;
				}
			}
			else//命令部分，都保存起来，不是空格，不是引号，就是元素
				tmp.push_back(input[i]);//其他东西
```

```
            }
            if(!tmp.empty())
                ans.push_back(tmp);//最后有一个

            cout<<ans.size()<<endl;
            for(int i = 0; i<ans.size(); i++)
                cout<<ans[i]<<endl;
        }

    return 0;
}
```

22. 公共子串计算（）

dp[i][j] = dp[i - 1][j - 1] + 1;

题目描述

给定两个**只包含小写字母的字符串，计算两个字符串的最大公共子串的长度。**

注：子串的定义指一个字符串删掉其部分前缀和后缀（也可以不删）后形成的**字符串。**

输入描述：
输入两个只包含小写字母的字符串
输出描述：
输出一个整数，代表最大公共子串的长度

示例 1

输入

复制

```
asdfas
```

```
werasdfaswer
```

输出

```
复制
```

```
6
```

```
#include<iostream>
#include<memory.h>
#include<vector>
using namespace std;
int main() {
	string str1;
	string str2;
	while (cin >> str1 >> str2) {
		int m = str1.length();
		int n = str2.length();
		//int dp[m+1][n+1];
		vector<vector<int>> dp(m+1,vector<int>(n+1,0));
		int max = 0;
		for (int i = 1; i <= m; i++)
		{
			for (int j = 1; j <= n; j++)
			{
				if (tolower(str1[i - 1]) == tolower(str2[j - 1]))//用 tolower
					dp[i][j] = dp[i - 1][j - 1] + 1;
				if (dp[i][j] > max)
					max = dp[i][j];
			}
		}
		cout << max << endl;
	}
	return 0;
}
```

23. 尼科彻斯定理（用公式求出第一项，之后往后加）

题目描述

验证尼科彻斯定理，即：任何一个整数 m 的立方都可以写成 m 个连续奇数之和。

例如：

1^3=1

2^3=3+5

3^3=7+9+11

4^3=13+15+17+19

输入一个正整数 m（m≤100），将 m 的立方写成 m 个连续奇数之和的形式输出。

本题含有多组输入数据。

输入描述：
输入一个 `int` 整数
输出描述：
输出分解后的 `string`

示例 1

输入

复制

```
6
```

输出

复制

```
31+33+35+37+39+41
```

尼科斯彻定理

高中的我们看到这个题的一瞬间，一定会觉得这是个弱智题。然而现在的我，想了好久。。

题目的意思是已知等差数列和 为 ，项数 n 为 m，公差 d 为 2，求首项：

复制代码

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        while (in.hasNextInt()) {
            int n = in.nextInt();
            long sum = (long)Math.pow(n,3);
            int a1 = (int)sum/n - (n - 1);
            StringBuilder sb = new StringBuilder(Integer.toString(a1));
            for(int i = 1; i < n; i++){
                a1 = a1 + 2;
                sb.append("+");
                sb.append(a1);
            }
            System.out.println(sb);
        }
```

```
    }
}
```

收起

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

int main() {
    int a,b,i;
    while (cin >> a ) {
        b=a*(a-1)+1;//第一个数
        for(i=0;i<a-1;i++)
            cout<<b+2*i<<'+';//首项等于这个
        cout<<b+2*i<<endl;
    }
}
```

24. 二维数组的操作（就是每次操作看是否越界）

题目描述

有一个大小的数据表，你会依次进行以下 5 种操作：

1.输入和，初始化大小的表格。

2.输入 x_1x_1、y_1y_1、x_2x_2、y_2y_2，交换坐标在(x_1,y_1)(x_1,y_1)和(x_2,y_2)(x_2,y_2)的两个数。

3.输入，在第行上方添加一行。

4.输入，在第列左边添加一列。

5.输入、，查找坐标为的单元格的值。

请编写程序，判断对表格的各种操作是否合法。

详细要求:

1.数据表的最大规格为 9 行*9 列，对表格进行操作时遇到超出规格应该返回错误。

2.对于插入操作，如果插入后行数或列数超过 9 了则应返回错误。如果插入成功了则将数据表恢复至初始化的大小，多出的数据则应舍弃。

3.所有输入坐标操作，对大小的表格，行号坐标只允许 0~m-1，列号坐标只允许 0~n-1。超出范围应该返回错误。

本题含有多组样例输入！

输入描述:

输入数据按下列顺序输入：

```
1 表格的行列值
2 要交换的两个单元格的行列值
3 输入要插入的行的数值
4 输入要插入的列的数值
5 输入要查询的单元格的坐标
```

输出描述:

输出按下列顺序输出：

```
1 初始化表格是否成功，若成功则返回 0， 否则返回-1
2 输出交换单元格是否成功
3 输出插入行是否成功
4 输出插入列是否成功
5 输出查询单元格数据是否成功
```

示例 1

输入

复制

```
4 9

5 1 2 6

0

8

2 3

4 7

4 2 3 2

3

3

4 7
```

输出

复制

```
0

-1

0

-1

0

0

-1

0

0

-1
```

说明

本组样例共有 2 组样例输入。

第一组样例：

1.初始化数据表为 4 行 9 列，成功

2.交换第 5 行 1 列和第 2 行 6 列的数据，失败。因为行的范围应该是(0,3)，不存在第 5 行。

3.在第 0 行上方添加一行，成功。

4.在第 8 列左边添加一列，失败。因为列的总数已经达到了 9 的上限。

5.查询第 2 行第 3 列的值，成功。

第二组样例：

1.初始化数据表为 4 行 7 列，成功

2.交换第 4 行 2 列和第 3 行 2 列的数据，失败。因为行的范围应该是(0,3)，不存在第 4 行。

3.在第 3 行上方添加一行，成功。

4.在第 3 列左边添加一列，成功。

5.查询第 4 行 7 列的值，失败。因为虽然添加了一行一列，但数据表会在添加后恢复成 4 行 7 列的形态，所以行的区间仍然在[0,3]，列的区间仍然在[0,6]，无法查询到(4,7)坐标。

```
#include <bits/stdc++.h>
using namespace std;

int main() {
	ios::sync_with_stdio(false);
	cin.tie(0), cout.tie(0);
	int m, n, x1, y1,x2, y2, x, y, qx, qy;
	while(cin >> m >> n >> x1 >> y1 >> x2 >> y2 >> x >> y >> qx >> qy) {//交换，插入行列，查询
		cout << ((m>=0 && m<=9 && n>=0 && n<=9) ? 0 : -1) << endl;
		cout << ((x1>=0 && x1<m && y1>=0 && y1<n && x2>=0 && x2<m && y2>=0 && y2<n) ? 0 : -1) << endl;
		cout << ((x>=0 && x<m && m<9) ? 0 : -1) << endl;
		cout << ((y>=0 && y<n && n<9) ? 0 : -1) << endl;
		cout << ((qx>=0 && qx<m && qy>=0 && qy<n) ? 0 : -1) << endl;
	}
	return 0;
}
```

25. 统计大写字母个数（>=A　<=Z）

题目描述

找出给定字符串中大写字符(即'A'-'Z')的个数。

输入描述:
本题含有多组样例输入
对于每组样例，输入一行，代表待统计的字符串
输出描述:
对于每组样例，输出一个整数，代表字符串中大写字母的个数

示例 1

输入

复制

```
add123#$%#%#O


150175017(&^%&$vabovbao
```

输出

复制

```
1

0
```

```
#include<iostream>

using namespace std;

int main()
{

	string s;

	while(getline(cin, s))
	{
		int cnt = 0;
		for(size_t i=0; i <s.size(); i++)
		{
			if(s[i]>='A'&&s[i]<='Z')
			{
				cnt++;
			}
		}
		cout << cnt <<endl;
	}
```

```
}
```

一行行

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

int main() {
    string str;
    while(getline(cin, str)) {
        int count = 0;
        for(int i = 0; i < str.size(); ++i) {
            if(str[i]-'A'>=0 && 'Z'-str[i]>=0) {
                ++count;
            }
        }
        cout << count << endl;
    }
}
```

26. 最长回文子串(先前后和自身，在斜着遍历，在斜着遍历 i+2 开始 n-3 开始)

校招时部分企业笔试将禁止编程题跳出页面，为提前适应，练习时请使用在线自测，而非本地 IDE。

题目描述

给定一个仅包含小写字母的字符串，**求它的最长回文子串的长度。**

所谓回文串，指左右对称的字符串。

所谓子串，指一个字符串删掉其部分前缀和后缀（也可以不删）的字符串

（注意：记得加上 while 处理多个测试用例）

输入描述：
输入一个仅包含小写字母的字符串
输出描述：
返回最长回文子串的长度

示例 1

输入

复制

```
cdabbacc
```

输出

复制

```
4
```

说明

```
abba 为最长的回文子串
```

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

using namespace std;

void longestPalindrome(string s){
        int n = (int)s.length();
```

```
    vector<vector<bool>> dp(n, vector<bool>(n,false));

    for(int i = 0; i < n; i++){//自身是 true
        dp[i][i] = true;
    }

    int begin = 0;
    int maxlen = 1;

    for(int i = 0; i < n-1; i++){//先判定前后，有些直接是 j-i<3 就是 ABA AA
        if(s[i] == s[i+1]){
            dp[i][i+1] = true;
            begin = i;
            maxlen = 2;
        }
    }

    for(int i = n - 3; i >=0 ; i--){
        for(int j = i + 2; j < n; j++){//i+2 i+1 已经有了
            if(s[i] == s[j] && dp[i+1][j-1]){
                dp[i][j] = true;
                if(maxlen < j - i + 1){
                    begin = i;
                    maxlen = j - i +1;
                }
            }
        }
    }

    cout<<maxlen<<endl;
}

int main(int argc, const char * argv[]) {
    string str;
    while(cin>>str){
        longestPalindrome(str);

    }
    return 0;
}
```

27. 最大连续 bit 数

题目描述

求一个 byte 数字对应的二进制数字中 1 的最大连续数，例如 3 的二进制为 00000011，最大连续 2 个 1

本题含有多组样例输入。

输入描述:
输入一个 byte 数字
输出描述:
输出转成二进制之后连续 1 的个数

示例 1

输入

复制

```
3

5
```

输出

复制

```
2



1
```

说明

```
3 的二进制表示是 11，最多有 2 个连续的 1。



5 的二进制表示是 101，最多只有 1 个连续的 1。
```

```
#include<iostream>
using namespace std;
int main()
{
	int n;
	while(cin>>n)
	{
		int max1=0,count=0,a=1;
		while(a)//死循环 A 不动啊
		{
			if(a&n)//第一个数
			{
				count++;
				max1=max(max1,count);
			}else
				count=0;
			a=a<<1;//左移动一个
		}
		cout<<max1<<endl;
	}
	return 0;
}
```

这个 ok

```
#include <iostream>
#include <string>
```

```
using namespace std;

int main()
{
	int num, cnt = 0;
	while (cin >> num)
	{
		int max_1 = 0;
		while (num)
		{
			int cnt = 0;
			while (num & 1)//只要有 1，没有 1 就重新开始
			{
				cnt++;
				num = num >> 1;//每次右移
			}
			max_1 = max(max_1, cnt);
			num = num >> 1;//自身右移
		}
		cout << max_1 << endl;
	}

	return 0;
}
```

28. 进制转换

题目描述

写出一个程序，接受一个十六进制的数，输出该数值的十进制表示。

输入描述:
输入一个十六进制的数值字符串。注意：一个用例会同时有多组输入数据，请参考帖子 `https://www.nowcoder.com/discuss/276` 处理多组输入的问题。
输出描述:
输出该数值的十进制字符串。不同组的测试用例用`\n`隔开。

示例 1

输入

复制

```
0xA


0xAA
```

输出

复制

```
10


170
```

另外在编程中十六进制数也用"0x"作为开头。

```
#include<iostream>
#include<string>
#include<cmath>
using namespace std;
 int main()
 {
        string s;
        while(cin>>s)
        {
              int bit=0;
              int ans =0;
              for(int i=s.length()-1;i>1;i--)//所以要大于 1    ox 开头  倒序遍历
              {
                    if(s[i]>='0'&&s[i]<='9')
                          ans+=(s[i]-'0')*pow(16,bit++);
                    else if(s[i]>='A'&&s[i]<='F')
                          ans+=(s[i]-'A'+10)*pow(16,bit++);
              }
              cout<<ans<<endl;
        }
```

```
        return 0;
}
```

1. 数学上对进制转换

0xabcd = a*16^3 + b*16^2 +c*16^1 +d*16^0

如果看成一个数组来存储，则在数组里对顺序与幂的大小成相反关系。

index = len-1 时，幂=0；

index = 2 时，幂 = len -3

2. 求字符串长度 len.

3. 考虑字符可能的三种情况，分别计算出对应的十进制数 digit

这里通过字符串想减，将十六进制位转换位十进制

4.计算对应的 sum = pow(16,j)*digit

29. 质数因子

题目描述

功能:输入一个正整数，按照从小到大的顺序输出它的所有质因子（重复的也要列举）（如 180 的质因子为 2 2 3 3 5 ）

最后一个数后面也要有空格

输入描述：
输入一个 long 型整数
输出描述：
按照从小到大的顺序输出它的所有质数的因子，以空格隔开。最后一个数后面也要有空格。

示例 1

输入

复制

```
180
```

输出

复制

```
2 2 3 3 5
```

30. 解题思路

1.质数知识

1.1 定义

质因数（素因数或质因子）在数论里是指能整除给定正整数的质数。除了 1 以外，两个没有其他**共同质因子的正整数称为互质**。因为 1 没有质因子，1 与任何正整数（包括 1 本身）都是互质。正整数的因数分解可将正整数表示为一连串的质因子相乘，质因子如重复可以用指数表示。根据算术基本定理，任何正整数皆有独一无二的质因子分解式。只有一个质因子的正整数为质数。

请在这里输入引用内容

每个合数都可以写成几个质数（也可称为素数）相乘的形式，这几个质数就都叫做这个合数的质因数。如果一个质数是某个数的因数，那么就说这个质数是这个数的质因数；而这个因数一定是一个质数。

1.2 例子

- 1 没有质因子。
- 5 只有 1 个质因子，5 本身。（5 是质数。）
- 6 的质因子是 2 和 3。(6 = 2 × 3)
- 2、4、8、16 等只有 1 个质因子：2（2 是质数，4 = 2，8 = 2，如此类推。）
- 10 有 2 个质因子：2 和 5。(10 = 2 × 5)

就是一个数的约数，并且是质数，比如 8=2×2×2，2 就是 8 的质因数。12=2×2×3，2 和 3 就是 12 的质因数。把一个式子以 12=2×2×3 的形式表示，叫做分解质因数。

16=2×2×2×2,2 就是 16 的质因数，把一个合数写成几个质数相乘的形式表示，这也是分解质因数。

分解质因数的方法是先用一个合数的最小质因数去除这个合数，得出的数若是一个质数，就写成这个合数相乘形式；若是一个合数就继续按原来的方法，直至最后是一个质数 。

分解质因数的有两种表示方法，除了大家最常用知道的“短除分解法”之外，还有一种方法就是“塔形分解法”。

分解质因数对解决一些自然数和乘积的问题有很大的帮助，同时又为求最大公约数和最小公倍数做了重要的铺垫

1.4 计算方法

短除法

求一个数分解质因数，要从**最小的质数除起，一直除到结果为质数为止。**分解质因数的算式的叫短除法，和除法的性质差不多，还可以用来求多个个数的公因式：

求最大公因数的一种方法，也可用来求最小公倍数。

求几个数最大公因数的方法，开始时用观察比较的方法，即：先把每个数的因数找出来，然后再找出公因数，最后在公因数中找出最大公因数。

2 程序思路

分解质因数代码：

将一个正整数分解质因数。例如：输入 90,打印出 90=2*33*5。

程序分析：对 n 进行分解质因数，应先找到一个最小的质数 k，然后按下述步骤完成：

(1)如果这个质数恰等于 n，则说明分解质因数的过程已经结束，打印出即可。

(2)如果 n>k，但 n 能被 k 整除，则应打印出 k 的值（k 为质因子）；并用 n 除以 k 的商，作为新的正整数你 n；同时 k++

重复执行第一步。

(3)如果 n 不能被 k 整除，则用 k++,重复执行第一步。

```
#include<iostream>
using namespace std;
#include<math.h>

int main()
{
	int num;
	string st;
	cin>>num;
	for (int i=2;i<=sqrt(num);i++)
	{
		while (num%i==0)//一直除以这个数
		{
			num/=i;
			cout<< i <<" ";
		}
	}
	if (num!=1)//自身除不尽，只能加上自身
		cout<< num <<" ";

	return 0;
}
```

题解 | #质数因子#(python 循环解法)

首先我们得知道一个整数 x 的质因子怎么求：

用 x 从小到大去除在 2 到根号 范围之间的整数。即从 2 开始除，如果能整除，记录下这个除数，然后用商去继续进行上述的操作，直到商为 1；如果除不进，除数加一。如果一直加一，除数大于，则说明 x 的质因子只有它本身。

（注：

1、范围是[2：]）的原因是，再往后就重复了（ **因为如果它不是质数，那么它一定可以表示成两个数（除了 1 和它本身）相乘，这两个数必然有一个小于等于它的平方根。只要找**

到小于或等于的那个就行了）

2、除法各个数的叫法：被除数 除数=商...余数

3、还有一个规律就是，求质因子时，用商继续去除的时候可以发现，后面能整出的数都比前面能整除的数大或等于。（eg.180 的质因子是 2 2 3 3 5，后面的都>=前面的）

）

代码:

```
import sys
if name=='main':
        line=sys.stdin.readline().strip('\n') #标准输入流
        x=int(line)
        res='' #用来存储输出的字符串*
        i=2
        while x!=1: #商不为 1
            if x%i==0: #如果能整除
                res+=str(i) #将当前的除数（当前的一个质因子）加到 res 中
                res+=' '
                x=x/i #将 a 变成商，继续上述操作
            else: #不能能整除
                if i>int(x**0.5): #如果除数比根号 x 大，则说明 x 的质因子只有它本身，故将 x 加 res,结束循环
                    res+=str(int(x))
                    res+=' '
                    break
                else:#不能能整除，但除数又还没大于根号 x
                    i+=1
        print(res)
```

收起

31. 合并表记录

题目描述

数据表记录包含表索引和数值（int 范围的正整数），请对表**索引相同的记录进行合并**，即

将相同索引的数值进行求和运算，输出按照 **key 值升序进行输出。**

输入描述:
先输入键值对的个数
然后输入成对的 index 和 value 值，以空格隔开
输出描述:
输出合并后的键值对（多行）

示例 1

输入

复制

```
4

0 1

0 2

1 2
```

```
3 4
```

输出

复制

```
0 3

1 2

3 4
```

```
#include<iostream>
#include<map>
using namespace std;
int main()
{
	int n;
	map<int,int> m;
	cin>>n;//有几组
	for(int i=0;i<n;i++)
	{
		pair<int,int> tmp;
		cin>>tmp.first;
		cin>>tmp.second;
		if((m.find(tmp.first))!=m.end())
			m[tmp.first]+=tmp.second;//如果存在就加起来
		else
			m[tmp.first]=tmp.second;
	}
	for(auto it=m.begin();it!=m.end();it++)
		cout<<it->first<<" "<<it->second<<endl;//按照已经排好序
	return 0;
}
```

32. 提取不重复的数

题目描述

输入一个 int 型整数，按照从**右向左的阅读顺序**，返回一个不含重复数字的新的整数。

保证**输入的整数最后一位不是 0**。

输入描述:
输入一个 `int` 型整数
输出描述:
按照从右向左的阅读顺序，返回一个**不含重复数字**的新的整数

示例 1

输入

复制

```
9876673
```

输出

复制

```
37689
```

```
#include<iostream>
#include<unordered_map>
#include<stdlib.h>
using namespace std;
```

```
int main(){
    string s;
    cin >> s;
    unordered_map<char, int> map;
    string ss;
    for(int i=s.size()-1; i>=0; i--){
        char ch = s[i];
        if(map.count(ch) <= 0)ss.push_back(s[i]);//如果没有加入
        map[ch]++;//有的话+1
    }

    for(char ch:ss){
        cout << ch;
    }

    cout << endl;
    return 0;
}
```

33. 字符个数统计

题目描述

编写一个函数，计算字符串中含有的不同字符的个数。字符在 ACSII 码范围内(0~127)，换

行**表示结束符**，不算在字符里。不在范围内的不作统计。多个相同的字符只计算一次

例如，对于字符串 abaca 而言，有 a、b、c 三种不同的字符，因此输出 3。

输入描述:
输入一行没有空格的字符串。
输出描述:
输出范围在(0~127)字符的个数。

示例 1

输入

复制

```
abc
```

输出

复制

```
3
```

```
#include<iostream>
#include<set>
#include<string>
using namespace std;

int main(){
	set<char> iset;
	string str;
	cin>>str;
	for(auto s:str){
		iset.insert(s);
	}
	cout<<iset.size();
	return 0;
}
```

```
#include<bits/stdc++.h>
using namespace std;

int main() {
	string str; cin >> str;
	unordered_set<char> set;
	for(char c : str)
		if(c >= 0 && c <= 127) set.insert(c);
	cout << set.size() << endl;
	return 0;
}
```

34. 字符串排序

题目描述

给定 n 个字符串，请对 n 个字符串按照**字典序排列。**

输入描述:

输入第一行为一个正整数 n(1≤n≤1000),下面 n 行为 n 个字符串(字符串长度≤100),字符串中只含有大小写字母。

输出描述:

数据输出 n 行，输出结果为按照字典序排列的字符串。

示例 1

输入

复制

```
9
cap
to
cat
card
```

```
two

too

up

boat

boot
```

输出

复制

```
boat

boot

cap

card

cat

to

too

two
```

up

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

int main()
{
	int cout1;
	cin>>cout1;
	vector<string> s1;
	for(int j = 0;j<cout1;j++)
	{
		string temp;
		cin >> temp;
		s1.push_back(temp);
	}
	sort(s1.begin(), s1.end());//里面就是字典序
	vector<string>::iterator it;
	for(it=s1.begin();it!=s1.end();it++)
		cout<<*it<<endl;
	return 0;

}
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;
bool cmp(string a,string b)
{
	//return a.compare(b)<0;	//升序
	return a<b;	//两个效果一样
}
int main()
{
	int n;
	cin>>n;
	string str[1001];
    for(int i=0;i<n;i++)
    {
        cin>>str[i];
    }
	sort(str,str+n,cmp);//自定义的 cmp 函数
```

```
    for(int i=0;i<n;i++)
    {
        cout<<str[i]<<endl;
    }
    return 0;
}
```

35. 购物单

题目描述

王强今天很开心，公司发给 N 元的年终奖。王强决定把年终奖用于购物，他把想买的物品分为两类：主件与附件，附件是从属于某个主件的，下表就是一些主件与附件的例子：

主件	附件
电脑	打印机，扫描仪
书柜	图书
书桌	台灯，文具
工作椅	无

如果要买归类为附件的物品，必须先买该附件所属的主件。每个主件可以有 0 个、 1 个或 2 个附件。附件不再有从属于自己的附件。王强想买的东西很多，为了不超出预算，他把每件物品规定了一个重要度，分为 5 等：用整数 1 ~ 5 表示，第 5 等最重要。他还从因特网上查到了每件物品的价格（都是 10 元的整数倍）。他希望在不超过 N 元（可以等于 N 元）的前提下，使每件物品的价格与重要度的乘积的总和最大。

设第 j 件物品的价格为 v[j]，重要度为 w[j]，共选中了 k 件物品，编号依次为 j 1， j 2，……，j k，则所求的总和为：

v[j 1]*w[j 1]+v[j 2]*w[j 2]+ … +v[j k]*w[j k]。（其中 * 为乘号）

请你帮助王强设计一个满足要求的购物单。

输入描述：

输入的第 1 行，为两个正整数，用一个空格隔开：N m

（其中 N （ <32000 ）表示总钱数， m （ <60 ）为希望购买物品的个数。）

从第 2 行到第 m+1 行，第 j 行给出了编号为 j-1 的物品的基本数据，每行有 3 个非负整数 v p q

（其中 v 表示该物品的价格（ v<10000 ），p 表示该物品的重要度（ 1 ~ 5 ），q 表示该物品是主件还是附件。如果 q=0 ，表示该物品为主件，如果 q>0 ，表示该物品为附件， q 是所属主件的编号）

输出描述：

输出文件只有一个正整数，为不超过总钱数的物品的价格与重要度乘积的总和的最大值（ <200000 ）。

示例 1

输入

复制

```
1000 5
800 2 0
400 5 1
300 5 1
400 3 0
500 2 0
```

输出

复制

```
2200
```

C++面向对象解决 01 背包变种问题——购物车

纯 C++，面向对象解决 01 背包变种问题——购物车

基本思想：

每一个主件视其附件的个数，可分为不同的情况，分别为：**（不放入），主件，主件+附件**

1，主件+附件 2，主件+附件 1+附件 2 这几种情况的组合。将所有的主件看作 01 背包中的石头，在外层 i 和 j 的循环内再比较上述情况的组合得到主件和附件一起考虑的最大值。如果没有附件，则只考虑主件。

程序步骤:

- 1 读入所有数据到 map 中;
- 2 将附件**连接到主件上**;
- 3 计算出每个主件的不同情况（基本思想中所述），并将此主件加入到一个新的 vector 内;
- 4 初始化二维 dp 表的第一行;
- 5 进行类似于多重背包的动态规划（带有附件的主件有多种选择，不带附件的主件只能选择放入或者不放入）；
- 6 记录路径（可选，此题不需要输出，以注释表示）；

源代码

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

int main()
{
    int N, m;
    while (std::cin >> N >> m) {
        int val[60][3] = {0}, cos[60][3] = {0}, total[3200] = {0};
        for (int i = 1; i <= m; ++i) {//从 1 开始
            int cost, p, q;
            std::cin >> cost >> p >> q;//价格  重要度，是否是主见，不等于 0 就是附件编号
            if (q != 0) {//就是附件
                if (val[q][1] == 0) {
                    val[q][1] = cost * p;//附件 1
                    cos[q][1] = cost;//q 就是属于哪个，属于 i
```

```
                } else {
                    val[q][2] = cost * p;//附件 2
                    cos[q][2] = cost;
                }
            } else {
                val[i][0] = cost * p;//等于 0 是主件，重要值乘以 p 主件
                cos[i][0] = cost;
            }
        }

        for (int i = 1; i <= m; ++i) { // 遍历所有物件
            for (int j = N / 10; j >= cos[i][0] / 10; --j) { // 每个钱数，都是 10 的倍数/10
                // 只选主件
                if (j >= cos[i][0] / 10) {
                    total[j] = std::max(total[j], val[i][0] + total[j - cos[i][0] / 10]);
                }

                // 选主件和一个附件//存在
                int tmp = (cos[i][0] + cos[i][1]) / 10;
                if (cos[i][1] != 0 && j >= tmp) {//i 代表主件
                    total[j] = std::max(total[j], val[i][0] + val[i][1] + total[j - tmp]);
                }

                // 选主件和两个附件
                tmp = (cos[i][0] + cos[i][1] + cos[i][2]) / 10;
                if (cos[i][2] != 0 && j >= tmp) {
                    total[j] = std::max(total[j], val[i][0] + val[i][1] + val[i][2] + total[j - tmp]);
                }
            }
        }

        std::cout << total[N/10] << std::endl;
    }

    return 0;
}
```

36. 简单密码

题目描述

密码是我们生活中非常重要的东东，我们的那么一点不能说的秘密就全靠它了。哇哈哈. 接下来渊子要在密**码之上再加一套密码，虽然简单但也**安全。

假设渊子原来一个 BBS 上的密码为 zvbo9441987,为了方便记忆，他通过一种算法把这个密码变换成 YUANzhi1987，这个密码是他的名字和出生年份，怎么忘都忘不了，而且可以明目张胆地放在**显眼的地方而不被别人知道真正的密码**。

他是这么变换的，大家都知道手机上的字

母： 1--1， abc--2, def--3, ghi--4, jkl--5, mno--6, pqrs--7, tuv--8 wxyz--9, 0--0,就这么简单，渊子把密码中出现的**小写字母都变成对应的数字**，数字和其他的符号都不做变换，

声明：密码中没有空格，而密码中出现的**大写字母则变成小写之后往后移一位**，如：X，先变成小写，再往后移一位，不就是 y 了嘛，简单吧。记住，z 往后移是 a 哦。

输入描述:
输入包括多个测试数据。输入是一个明文，密码长度不超过 100 个字符，输入直到文件结尾
输出描述:

输出渊子真正的密文

示例 1

输入

复制

```
YUANzhi1987
```

输出

复制

```
zvbo9441987
```

```
#include<iostream>
#include<string>
using namespace std;
const string dict1="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
const string dict2="bcdefghijklmnopqrstuvwxyza22233344455566677778889999";

char Char_Change(char a){
	for(int i=0;i<dict1.size();i++)
		if(dict1[i]==a) return dict2[i];
	return a;
}

int main(){
	//string data="YUANzhi1987";
	string data;
	while(getline(cin,data)){
		for(int i=0;i<data.size();i++)
			data[i] = Char_Change(data[i]);
		cout<<data<<endl;
```

```
    }
    return 0;
}

#include <iostream>

using namespace std;

int main() {
    string s;
    string m = "22233344455566677778889999";

    cin >> s;

    for (auto & e : s) {
        if (e >= 'A' && e <= 'Z') {//大写
            if (e == 'Z')
                e = 'a';
            else e = tolower(e + 1);
        }
        else if (e >= 'a' && e <= 'z') {//小写
            e = m[e - 'a'];
        }
    }

    cout << s << endl;

    return 0;
}
```

37. 字符串排序

题目描述

编写一个程序，将输入字符串中的字符按如下规则排序。

规则 1 ： 英文字母从 **A 到 Z 排列，不区分大小写。**

如，输入： Type 输出： epTy

规则 2 ：同一个英文字母的大小写同时存在时，按照**输入顺序排列。**

如，输入： BabA 输出： aABb

规则 3 ：非英文字母的**其它字符保持原来的位置。**

如，输入： By?e 输出： Be?y

注意有多组测试数据，即输入有多行，每一行单独处理（换行符隔开的表示不同行）

输入描述：

输入字符串

输出描述：

输出字符串

示例 1

输入

复制

```
A Famous Saying: Much Ado About Nothing (2012/8).
```

输出

复制

```
A aaAAbc dFgghh: iimM nNn oooos Sttuuuy (2012/8).
```

#
本文系「人工智能安全」（微信公众号）原创，转载请联系本文作者（同博客作者）。
欢迎你转发分享至朋友圈，并给予「关注、星标、点赞」三连支持。互相欣赏，互相批判。
我是一名有诗人气质的网络安全工程师
期待与你的思想交流碰撞出智慧的花火
水木清华
2020-03-16
字符串排序
*/

```
#include <iostream>
#include <vector>
using namespace std;
string String_Sorting(string str)
{
	int len = str.size(); //获取字符串长度
	vector <char> vec; //用一个 char 型的向量存储按规则排序后的字符串中的字母字符
	//规则一：英文字母从 A 到 Z 排列，不区分大小写。
	//规则二：同一个英文字母的大小写同时存在时，按照输入顺序排列。
	for (int j = 0; j < 26; j++)
	{
		for (int i = 0; i < len; i++)
		{
			if ((str[i] - 'a' == j) || (str[i] - 'A' == j))//如果里面有对应字符
			{
				vec.push_back(str[i]); //将符合规则的字母字符先后写入向量
			}
		}
```

```
        }
        //规则三：非英文字母的其它字符保持原来的位置。
        for(int i = 0,k = 0;(i < len) && (k < vec.size()); i++)
        {
            if((str[i] >= 'a' && str[i] <= 'z') || (str[i] >= 'A' && str[i] <= 'Z'))字母的话。。。
            {
                str[i] = vec[k++];
            }
        }
        return str; //返回按规则排序好后的字符串
}
//主函数
int main()
{
        string str;
        while (getline(cin, str))
        {
            cout << String_Sorting(str) << endl;
        }
        return 0;
}
```

38. 图片整理

题目描述

Lily 上课时使用字母数字图片教小朋友们学习英语单词，每次都需要把这些图片按照大小

（ASCII 码值从小到大）排列收好。请大家给 Lily 帮忙，通过 C 语言解决。

本题含有多组样例输入。

输入描述:

Lily 使用的图片包括"A"到"Z"、"a"到"z"、"0"到"9"。输入字母或数字个数不超过 1024。

输出描述:

Lily 的所有图片按照从小到大的顺序输出

示例 1

输入

复制

```
Ihave1nose2hands10fingers
```

输出

复制

```
0112Iaadeeefghhinnnorsssv
```

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


int main(){
        string data;
        while(cin>>data)
        {
                sort(data.begin(),data.end());
                cout<<data<<endl;
        }
        return 0;
}
/*
本文系「人工智能安全」（微信公众号）原创，转载请联系本文作者（同博客作者）。
欢迎你转发分享至朋友圈，并给予「关注、星标、点赞」三连支持。互相欣赏，互相批判。
我是一名有诗人气质的网络安全工程师
期待与你的思想交流碰撞出智慧的花火
```

```
水木清华
2020-03-19
图片整理
*/
#include <iostream>
#include <algorithm>
using namespace std;
//字符排序的函数接口
string String_Sorting(string str)
{
	//按字符的 ASCII 码排序
	sort(str.begin(), str.end());
	//返回排序后的字符串
	return str;
}
//主函数
int main()
{
	string str;
	while (getline(cin, str))
	{
		cout << String_Sorting(str) << endl;
	}
	return 0;
}
```

39. 蛇形矩阵

题目描述

蛇形矩阵是由 1 开始的自然数依次排列成的一个矩阵上三角形。

例如，当输入 5 时，应该输出的三角形为：

1 3 6 10 15

2 5 9 14

4 8 13

7 12

11

请注意本题含有多组样例输入。

输入描述:
输入正整数 N（N 不大于 100）
输出描述:
输出一个 N 行的蛇形矩阵。

示例 1

输入

复制

```
4
```

输出

复制

```
1 3 6 10

2 5 9

4 8

7
```

```
#include<iostream>

using namespace std;

int main(int argc, char** argv)
{
	int n;

	while (cin >> n)//第几行
	{
		int beg = 1;
		for (int i = 1; i <= n; ++i)//一共有几行，n 行
		{
			cout << beg;//起始位置
			int temp = beg;
			for (int j = i + 1; j <= n; ++j)
			{
				temp += j;//每一行，加上 2 开始
				cout << ' '<<temp;
			}
			beg += i;//开始每次加上 i，几行。
			cout << endl;
		}
	}
}
```

40. 字符串加密

题目描述

有一种技巧可以对数据进行加密，它使用一个单词作为它的密匙。下面是它的工作原理：首先，选择一个单词作为密匙，如 TRAILBLAZERS。如果单词中包含有重复的字母，只保留第 1 个，其余几个丢弃。现在，修改过的那个单词属于字母表的下面，如下所示：

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

T R A I L B Z E S C D F G H J K M N O P Q U V W X Y

上面其他用字母表中剩余的字母填充完整。在对信息进行加密时，信息中的每个字母被固定于顶上那行，并用下面那行的对应字母一一取代原文的字母(字母字符的大小写状态应该保留)。因此，使用这个密匙，Attack AT DAWN(黎明时攻击)就会被加密为 Tpptad TP ITVH。

请实现下述接口，通过指定的**密匙和明文**得到密文。

本题有多组输入数据。

输入描述：
先输入 key 和要加密的字符串
输出描述：
返回加密后的字符串

示例 1

输入

复制

```
nihao


ni
```

输出

复制

```
le
```

```
#include<iostream>
#include<string>
#include<vector>
using namespace std;
void decode(string s, string code)
{
	string biaozhun = "abcdefghijklmnopqrstuvwxyz";
	for (int i = 0; i < 26; i++)
	{

		if (s.find(biaozhun[i]) != string::npos|| s.find((char)toupper(biaozhun[i])) != string::npos)
			continue;
		else
			s += biaozhun[i];

	}
	for (int i = 0; i < code.size(); i++)
	{
		int index = biaozhun.find(code[i]);
		if (index == -1)
		{
			index= biaozhun.find((char)tolower(code[i]));
```

```
            }
            if ('a' <= code[i] && code[i] <= 'z')
                cout <<(char) tolower(s[index]);
            else
            {
                cout << (char)toupper(s[index]);
            }

        }
        cout << endl;


}
int main() {

        string str, code;
        while (cin >> str>>code)
        {

            string str1;
            for (int i = 0; i < str.size(); i++)
            {
                if (str1.find(str[i]) != string::npos)
                    continue;
                else
                    str1 += str[i];
            }
            decode(str1, code);
        }
}
```

这个 先生成表

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

int main()
{
        string h = "abcdefghijklmnopqrstuvwxyz";
        string h1 = "abcdefghijklmnopqrstuvwxyz";
        string key, str;
```

```
    while (cin >> key >> str)//用 key 生成表，用 str 读取
    {
        string temp;
        vector<int> counter(26, 0);
        int f[26] = { 0 };
        int index;
        string key1;
        for (int i = 0; i < key.size(); i++)
        {
            if (key[i] >= 'a' && key[i] <= 'z')//小写
                index = key[i] - 'a';
            else if (key[i] >= 'A' && key[i] <= 'Z')//大写
                index = key[i] - 'A';
            f[index]++;//统计 key 中每个字母出现的次数
            if (f[index] == 1)//第一次出现则计算
                key1 += key[i];//按照顺序，不重复
        }
        for (int j = 0; j < key1.size(); j++)//这里是确定经过密钥修改后的字母表
        {
            h[j] = key1[j];//前面的都填充
        }
        int k = 0;
        int l = key1.size();//从后面开始
        while(k < 26)
        {
            if (key1.find(h1[k]) == -1)
            {
                h[l] = h1[k];//字母不存在，就要替换
                l++;
            }
            k++;
        }//执行到这里的时候，经过密钥修改后的字母表完成 就是 h 了
        for (int i = 0; i < str.size(); i++)
        {
            if (h1.find(str[i]) != -1)//这里不能直接写 if(h1.find(str[i])),
                                      //因为有可能查到的字母位置是下标 0
                cout << h[h1.find(str[i])];//有的可能是空格，h1 是一直不变的
        }
        cout << '\n';
    }
    return 0;
}
```

41. 小球落地 5 次经过的路程

题目描述

假设一个球从任意高度自由落下，每次落地后**反跳回原高度的一半**; 再落下, 求它在第 5 次落地时，**共经历多少米?第 5 次反弹多高**?

最后的误差判断是小数点 6 位

输入描述:
输入起始高度，int 型
输出描述:
分别输出第 5 次落地时，共经过多少米第 5 次反弹多高

示例 1

输入

复制

```
1
```

输出

复制

```
2.875
```

```
0.03125
```

```
#include<iostream>
using namespace std;
int main(){
	double data;
	while(cin >> data){
		double sum = data;
		double height = data;
		for(int i = 2; i <= 5; ++i){
			height /= 2;?/每次是一半
			sum += height * 2; //来回是两次。落地
		}
		cout << sum << endl << height/2 << endl;
	}
	return 0;
}
```

42. 统计字符

题目描述

输入一行字符，分别统计出包含**英文字母、空格、数字和其它字符的个数。**

本题包含多组输入。

输入描述:
输入一行字符串，可以有空格
输出描述:
统计其中英文字符，空格字符，数字字符，其他字符的个数

示例 1

输入

复制

```
1qazxsw23 edcvfr45tgbn hy67uj m,ki89ol.\\/;p0-=\\][
```

输出

复制

```
26

3

10

12
```

```
#include<iostream>
#include<string>

using namespace std;

int main()
{
	string s;
	while(getline(cin,s))//一行获取
	{
		int a=0;
		int b=0;
		int c=0;
		int d=0;
		for(int i=0;i<s.size();i++)
		{
			if(s[i]==' ') b++;//空格
			else if(s[i]>='0'&&s[i]<='9') c++;//数字
			else if((s[i]>='a'&&s[i]<='z')||(s[i]>='A'&&s[i]<='Z')) a++;//字母
```

```
                else d++;//其他
            }
            cout<<a<<endl<<b<<endl<<c<<endl<<d<<endl;
        }
        return 0;
}
```

43. 迷宫问题

题目描述

定义一个二维数组 N*M（其中 2<=N<=10;2<=M<=10），如 5 × 5 数组下所示：

```
int maze[5][5] = {
0, 1, 0, 0, 0,
0, 1, 0, 1, 0,
0, 0, 0, 0, 0,
0, 1, 1, 1, 0,
0, 0, 0, 1, 0,
};
```

它表示一个迷宫，其中的 1 表示墙壁，0 表示可以走的路，只能横着走或竖着走，不能斜着走，要求编程序找出从左上角到右下角的最短路线。入口点为[0,0],既第一空格是可以走的路。

本题含有多组数据。

输入描述:
输入两个整数，分别表示二位数组的行数，列数。再输入相应的数组，其中的 1 表示墙壁，0 表示可以走的路。数据保证有唯一解,不考虑有多解的情况，即迷宫只有一条通道。
输出描述:
左上角到右下角的最短路径，格式如样例所示。

示例 1

输入

复制

```
5 5

0 1 0 0 0

0 1 0 1 0

0 0 0 0 0

0 1 1 1 0

0 0 0 1 0
```

输出

复制

```
(0,0)

(1,0)

(2,0)

(2,1)

(2,2)

(2,3)

(2,4)

(3,4)

(4,4)
```

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

int main(){
    int n, m;
    int t[4][2] = {{-1,0}, {0,-1}, {1,0},{0,1}};
    int v[12][12];
    while(cin >> n >> m)
    {
        vector<pair<int,int> > r;//按照下标排列
        stack<pair<int,int> > s;
        fill(v[0], v[0]+12*12, 1);
```

fill 函数的作用是：将一个区间的元素都赋予 val 值。函数参数：fill(first,last,val);//first 为容器的首迭代器，last 为容器的末迭代器，val 为将要替换的值。

```
        for(int i = 1; i <= n; i++)
            for(int j = 1; j <= m; j++)
                cin >> v[i][j];
        s.push(make_pair(1,1));//肯定是 0
        v[1][1] = 1;//不能走回头路
        while(!s.empty())
        {
            pair<int, int> p = s.top();
            s.pop();
            for(int i = 0; i < 4; i++){ //  上下左右 4 点
                int x = p.first + t[i][0];
                int y = p.second + t[i][1];
                if(v[x][y] == 0) { // 0 说明该点能走，但未被访问过
                    v[x][y] = p.first * 100 + p.second; //  用于记录路径的上个点
                    if(x == n && y == m) break; //    到达终点跳出
                    s.push(make_pair(x,y));
                }
            }
        }
        for (int x = n, y = m, t = v[n][m]; v[x][y] != 1; x = t/100, y = t%100)
        {
            t = v[x][y];
            r.push_back(make_pair(x, y));//
        }
        r.push_back(make_pair(1, 1));
        reverse(r.begin(), r.end());
        for(int i = 0; i < r.size(); i++)
        {
            cout << '(' << r[i].first-1 << ',' << r[i].second-1 << ')' << endl;//要减去 1
        }
    }
    return 0;
}
```

Dfs

```
#include<iostream>
#include<vector>
#include<map>
using namespace std;
int m,n;
vector<vector<int>> maze;
vector<pair<int,int>> tmppath;//临时路径
```

```
vector<pair<int,int>> finalpath;//最终路径
void find(int x,int y)
{
	if(x==m-1&&y==n-1)//出口
	{
		maze[x][y]=1;
		tmppath.push_back(pair<int,int>(x,y));
		if(finalpath.empty()==1||tmppath.size()<finalpath.size())
		{
			finalpath=tmppath;
		}
		maze[x][y]=0;
		tmppath.pop_back();
	}
	else if(x>=m||y>=n||x<0||y<0||maze[x][y]==1)//越界
	{
		return;
	}
	else
	{	//cout<<x<<y<<endl;
		maze[x][y]=1;
		tmppath.push_back(pair<int,int>(x,y));//四个方向
		find(x+1,y);
		find(x-1,y);
		find(x,y+1);
		find(x,y-1);
		maze[x][y]=0;//回退
		tmppath.pop_back();
	}

}
int main()
{
	while(cin>>m>>n)
	{
		maze = vector<vector<int>>(m, vector<int>(n, 0));//m 行 n 列
		for(int i=0;i<m;i++)
		{
			for(int j=0;j<n;j++)
			{
				cin>>maze[i][j];
			}
		}
		tmppath.clear();
```

```
            finalpath.clear();
            find(0,0);
            vector<int>it;
                for(vector<pair<int,int>>::iterator i=finalpath.begin();i!=finalpath.end();i++)
                {
                    pair<int,int>p=*i;//每个要乘以 i
                    cout<<'('<<p.first<<','<<p.second<<')'<<endl;
                }
            maze.clear();
            }

}
```

44. 名字的漂亮度

题目描述

给出一个名字,该名字有 26 个字符串组成,定义这个字符串的“漂亮度”是其所有字母“漂亮度”的总和。

每个字母都有一个“漂亮度”，范围在 1 到 26 之间。没有任何两个字母拥有相同的“漂亮度”。字母忽略大小写。

给出多个名字，计算每个名字最大可能的“漂亮度”。

本题含有多组数据。

输入描述:
整数 N，后续 N 个名字
输出描述:
每个名称可能的最大漂亮程度

示例 1

输入

复制

```
2

zhangsan

lisi
```

输出

复制

```
192

101

漂亮度 1-26,次数最多最大呗，用 map,vector
```

将输入数据去重，并记录出现的次数，次数最多的记为 26，其次记为 25……，最大漂亮度为次数和单个字符漂亮度乘积的和

Kg

```
#include <iostream>
```

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

int main()
{
	int test;
	while (cin >> test)
	{
		while (test--)
		{
			string st;
			cin >> st;//字符
			int i, a[26] = {0}, k = 26, res = 0;
			for (i = 0; i < st.length(); ++i)
			{
				if (st[i] >= 'a' && st[i] <= 'z')
					a[st[i] - 'a']++;//每个字母出现的次数
				else
					a[st[i] - 'A']++;
			}
			sort(a, a + 26);//从大到小排列
			for (i = 25; i >= 0; --i)
				res += a[i] * k--;//最大就是 26，后面都是 0 呗，也没啥用。
			cout << res << endl;
		}
	}
	return 0;
}
```

45. 截取字符串

题目描述

输入一个字符串和一个整数 k，**截取字符串的前 k 个字符**并输出

本题输入含有多组数据

输入描述:

1.输入待截取的字符串

2.输入一个正整数 k，代表截取的长度

输出描述:

截取后的字符串

示例 1

输入

复制

```
abABCcDEF
6
```

输出

复制

```
abABCc
```

示例 2

输入

复制

```
ffIKEHauv
```

```
1
```

```
bdxPKBhih
```

```
6
```

输出

复制

```
f
```

```
bdxPKB
```

```
#include <iostream>
#include <string>
#include <stdio.h>

using namespace std;

int main(int argc, char **argv)
{
	string input;
	int k = 0;

	while(cin >> input){
		cin >> k;
		//printf("%s\n", input.substr(0, k).c_str());
		cout << input.substr(0, k) << endl;
		input.clear();
	}

	//cin >> input;

	return 0;
}
```

46. 从单向链表删除指定的值

题目描述

输入一个单向链表和一个节点的值，从单向链表中**删除等于该值的节点**，删除后如果链表中无节点则返回空指针。

链表的值不能重复。

构造过程，例如输入一行数据为:

6 2 **1 2 3 2 5 1 4 5 7 2** 2

则第一个参数 6 表示输入总共 6 个节点，第二个参数 2 表示头节点值为 2，剩下的 2 个一组表示**第 2 个节点值后面插入第 1 个节点值**，为以下表示:

1 2 表示为

2->1

链表为 2->1

3 2 表示为

2->3

链表为 2->3->1

5 1 表示为

1->5

链表为 2->3->1->5

4 5 表示为

5->4

链表为 2->3->1->5->4

7 2 表示为

2->7

链表为 2->7->3->1->5->4

最后的链表的顺序为 2 7 3 1 5 4

最后一个参数为 2，表示要删掉节点为 2 的值

删除 结点 2

则结果为 7 3 1 5 4

链表长度不大于 1000，每个节点的值不大于 10000。

测试用例保证输入合法

输入描述：

输入一行，有以下 4 个部分：

1 输入**链表结点个数**
2 输入头结点的值
3 按照格式插入各个结点
4 输入要删除的结点的值

输出描述：

输出一行

输出删**除结点后的序列，每个数后都要加空格**

示例 1

输入

复制

```
5 2 3 2 4 3 5 2 1 4 3
```

输出

复制

```
2 5 4 1
```

说明

形成的链表为 2->5->3->4->1

删掉节点 3，返回的就是 2->5->4->1

示例 2

输入

复制

```
6 2 1 2 3 2 5 1 4 5 7 2 2
```

输出

复制

```
7 3 1 5 4
```

说明

如题

```
#include<iostream>
#include<vector>
#include<list>
#include<algorithm>
using namespace std;
int main() {
	int count = 0,list_begin;
	while (cin >> count >> list_begin) {
		list<int>list_a;
		list_a.push_back(list_begin);//头结点
		for (int i = 0; i < count - 1; i++) {
			int temp1, temp2;
			cin >> temp1 >> temp2;//temp2->temp1 链表格式
			list_a.insert(++find(list_a.begin(), list_a.end(), temp2), temp1);/找到 temp2 在后面插入
		}
		int delete_value;
		cin >> delete_value;//要删除的节点
		list_a.erase(find(list_a.begin(), list_a.end(), delete_value));//删除地址
		for (list<int>::iterator it=list_a.begin(); it!= list_a.end(); it++) {
			cout << *it<< " ";
		}
		cout << endl;
	}


}
```

47. 多线程

题目描述

问题描述：有 4 个线程和 1 个公共的字符数组。线程 1 的功能就是向数组输出 A，线程 2 的功能就是向字符输出 B，线程 3 的功能就是向数组输出 C，线程 4 的功能就是向数组输出

D。要求按顺序向数组赋值 ABCDABCDABCD，ABCD 的个数由线程函数 1 的参数指定。[注：

C 语言选手可使用 WINDOWS SDK 库函数]

接口说明：

void init()；//初始化函数

void Release(); //资源释放函数

unsignedint__stdcall ThreadFun1(PVOID pM) ；//线程函数 1，传入一个 int 类型的指针

[取值范围：1 – 250，测试用例保证]，用于初始化输出 A 次数，资源需要线程释放

unsignedint__stdcall ThreadFun2(PVOID pM) ;//线程函数 2，无参数传入

unsignedint__stdcall ThreadFun3(PVOID pM) ;//线程函数 3，无参数传入

Unsigned int __stdcall ThreadFunc4(PVOID pM);//线程函数 4，无参数传入

char g_write[1032]; //线程 1,2,3,4 按顺序向该数组赋值。不用考虑数组是否越界，测试

用例保证

输入描述：
本题含有多个样例输入。
输入一个 int 整数
输出描述：
对于每组样例，输出多个 ABCD

示例 1

输入

复制

```
10
```

输出

复制

```
ABCDABCDABCDABCDABCDABCDABCDABCDABCDABCD
```

```
#include<iostream>
#include<string>
using namespace std;
int main(){
	int n;
	while(cin >> n){
		for(int i=0; i<n;i++){
			cout << "ABCD" ;
			}
		cout << endl;
	}
	return 0;
}
```

48. 计算字符串距离

题目描述

Levenshtein 距离，又称编辑距离，指的是两个字符串之间，由一个转换成另一个所需的最少编辑操作次数。许可的编辑操作包括将一个字符替换成另一个字符，插入一个字符，删除一个字符。编辑距离的算法是首先由俄国科学家 Levenshtein 提出的，故又叫 Levenshtein Distance。

Ex：

字符串 A:abcdefg

字符串 B: abcdef

通过增加或是删掉字符”g”的方式达到目的。这两种方案都需要一次操作。把这个操作所需要的次数定义为两个字符串的距离。

要求：

给定任意两个字符串，写出一个算法计算它们的编辑距离。

本题含有多组输入数据。

输入描述：
每组用例一共 2 行，为输入的两个字符串
输出描述：
每组用例输出一行，代表字符串的距离

示例 1

输入

复制

```
abcdefg

abcdef
```

```
abcde

abcdf

abcde

bcdef
```

输出

复制

```
1

1

2
```

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
//空间可以优化成 O(n)
int main() {
    string s1, s2;
    while (cin >> s1 >> s2) {
        vector<vector<int>> dp(s1.size() + 1, vector<int>(s2.size() + 1, 0));
        for (int i = 1; i <= s2.length(); i++) dp[0][i] = i;
        for (int i = 1; i <= s1.length(); i++) dp[i][0] = i;
        for(int i=1;i<=s1.length();i++)
            for (int j = 1; j <= s2.length(); j++) {
                int min1 = min(dp[i - 1][j], dp[i][j - 1]) + 1;
                dp[i][j] = min((s1[i - 1] == s2[j - 1] ? 0 : 1) + dp[i - 1][j - 1], min1);
```

```
                }
            cout << dp[s1.size()][s2.size()] << endl;
        }
}
```

49. 挑 7

题目描述

输出 7 有关数字的个数，包括 7 的倍数，还有包含 7 的数字（如 17，27，37...70，71，

72，73...）的个数（一组测试用例里可能有多组数据，请注意处理）

输入描述:
一个正整数 N。(N 不大于 30000)
输出描述:
不大于 N 的与 7 有关的数字个数，例如输入 20，与 7 有关的数字包括 7,14,17.

示例 1

输入

复制

```
20


10
```

输出

复制

```
3


1
```

```
#include<iostream>
using namespace std;
bool isRelateTo7(int n){
	if(n%7==0){
		return true;//倍数
	}
	while(n!=0){
		if(n%10==7){
			return true;//含有 7
		}
		n=n/10;
	}
	return false;
}
int main(){
	int N;
	while(cin>>N){
		int count=0;
		for(int i=1;i<=N;i++){
			if(isRelateTo7(i)){
				count++;
			}
		}
	    cout<<count<<endl;
	}
}
```

50. 高精度整数加法

题目描述

输入两个用字符串表示的整数，求它们所表示的数之和。

字符串的长度不超过 10000。

本题含有多组样例输入。

输入描述:
输入两个字符串。保证字符串只含有'0'~'9'字符
输出描述:
输出求和后的结果

示例 1

输入

复制

```
9876543210


1234567890
```

输出

复制

```
11111111100
```

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

int main()
{
    string s1, s2;
    while(cin>>s1>>s2)
```

```
    {
        int len = min(s1.length(), s2.length());

        string ret;
        int end1 = s1.length() - 1;
        int end2 = s2.length() - 1;
        int pmt = 0;
        while(len--)
        {
            int t1 = s1[end1] - '0';
            int t2 = s2[end2] - '0';
            ret += (t1 + t2 + pmt) % 10 + '0';
            pmt = (t1 + t2 + pmt) / 10;
            end1--;end2--;//最小的相加
        }

        string tmp = s1.length() >= s2.length()? s1:s2;
        for(int i = tmp.length() - min(s1.length(), s2.length()) - 1; i >= 0; i--)
        {
            int t = tmp[i] - '0';
            ret += (t + pmt) % 10 + '0';
            pmt = (t + pmt) / 10;
        }
        if(pmt)
            ret += (pmt + '0');//如果还有，加起来

        reverse(ret.begin(), ret.end());//最后反转
        cout << ret << endl;
    }
}
```

51. 查找字符串中第一个只出现一次

题目描述

找出字符串中第一个只出现一次的字符

输入描述:
输入几个非空字符串
输出描述:
输出第一个只出现一次的字符，如果不存在输出-1

示例 1

输入

复制

```
asdfasdfo

aabb
```

输出

复制

```
o

-1
```

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

using namespace std;

int main(){
        string str;
```

```
        while(cin >> str){
            vector<char> alphabet(26, 0);
            for(auto c:str){
                alphabet[c - 'a']++;
            }
            bool found = false;
            for(auto c:str){
                if(alphabet[c - 'a'] == 1){
                    cout << c << endl;
                    found = true;
                    break;
                }
            }
            if(found == false){
                cout << -1 << endl;
            }
        }
}
```

52. 查找组成一个偶数最接近的两个

题目描述

任意一个偶数（大于 2）都可以由 2 个素数组成，组成偶数的 2 个素数有很多种情况，本题目要求输出组成指定偶数的两个素数差值最小的素数对。

本题含有多组样例输入。

输入描述：
输入一个偶数
输出描述：
输出两个素数

示例 1

输入

复制

```
20
```

输出

复制

```
7

13
```

题目描述

任意一个偶数（大于 2）都可以由 2 个素数组成，组成偶数的 2 个素数有很多种情况，本题目要求输出组成指定偶数的两个素数差值最小的素数对。

本题含有多组样例输入。

输入描述：
输入一个偶数
输出描述：
输出两个素数

示例 1

输入

复制

```
20
```

输出

```
复制
```

```
7

13
```

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include <list>
#include<string>
#include <sstream>
#include <thread>
#include <mutex>
#include <condition_variable>


using namespace std;

int issushu(int a) {
	for (int i = 2; i < a;i++)
	{
		if(a%i==0)return 0;//整除就是 0
	}
	return 1;
}

int main()
{
	int num;

	while (cin >> num) {
		int half = num / 2;
		for (int i = 0; i <half;i++)//到一半
```

```
            {
                if (issushu(half+i)&&issushu(half-i))//如果相差 2*half
                {
                    cout << half - i << endl;
                    cout << half + i << endl;
                    break;
                }
            }
        }
        return 0;
}
```

53. DNA 序列

题目描述

一个 DNA 序列由 A/C/G/T 四个字母的排列组合组成。G 和 C 的比例（定义为 GC-Ratio）是序列中 G 和 C 两个字母的总的出现次数除以总的字母数目（也就是序列长度）。在基因工程中，这个比例非常重要。因为高的 GC-Ratio 可能是基因的起始点。

给定一个很长的 DNA 序列，以及要求的最小子序列长度，研究人员经常会需要在其中找出 GC-Ratio 最高的子序列。

本题含有多组样例输入。

输入描述:
输入一个 `string` 型基因序列，和 `int` 型子串的长度
输出描述:
找出 `GC` 比例最高的子串,如果有多个输出第一个的子串

示例 1

输入

复制

```
AACTGTGCACGACCTGA

5
```

输出

复制

```
GCACG
```

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	string str;
	int N;
	while(cin>>str>>N)
	{
		int len=str.length();
		int pos=0;
		int max=0;
		for(int i=0;i<len-N+1;i++)
		{
			int count=0;
			for(int j=0;j<N;j++)
				if(str[i+j]=='G'||str[i+j]=='C')
					count++;//个数一直加
			if(count>max)
			{
				max=count;
				pos=i;
			}
```

```
            }
            for(int i=0;i<N;i++)
                cout<<str[pos+i];
            cout<<endl;
        }
        return 0;
}
```

54. MP3 光标位置

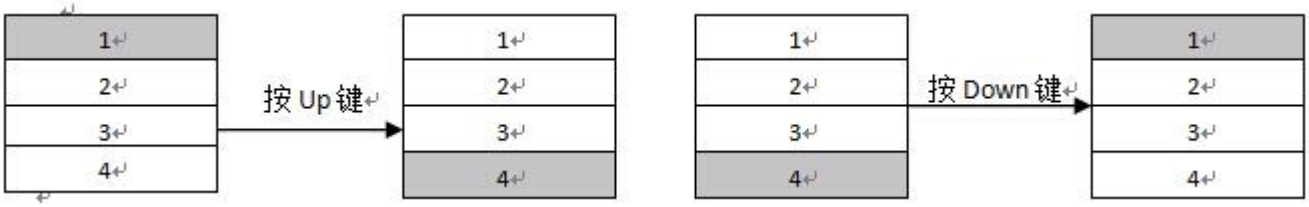
题目描述

MP3 Player 因为屏幕较小，显示歌曲列表的时候每屏只能显示几首歌曲，用户要通过上下键才能浏览所有的歌曲。为了简化处理，假设每屏只能显示 4 首歌曲，光标初始的位置为第 1 首歌。

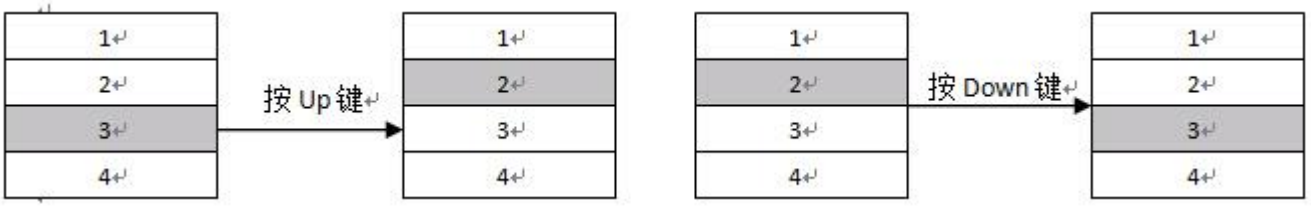
现在要实现通过上下键控制光标移动来浏览歌曲列表，控制逻辑如下：

1. 歌曲总数<=4 的时候，不需要翻页，只是挪动光标位置。

2. 光标在第一首歌曲上时，按 Up 键光标挪到最后一首歌曲；光标在最后一首歌曲时，按 Down 键光标挪到第一首歌曲。

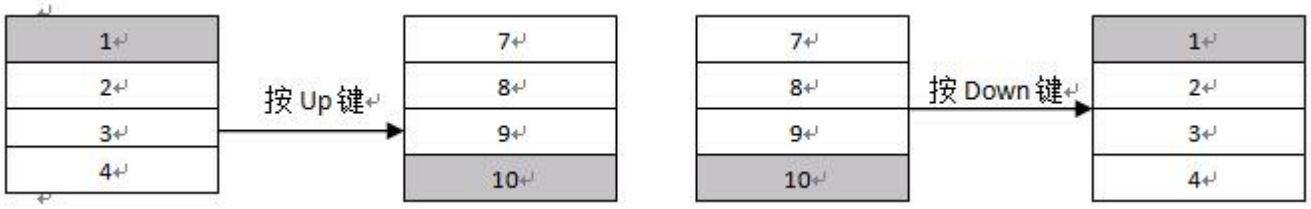


其他情况下用户按 Up 键，光标挪到上一首歌曲；用户按 Down 键，光标挪到下一首歌曲。

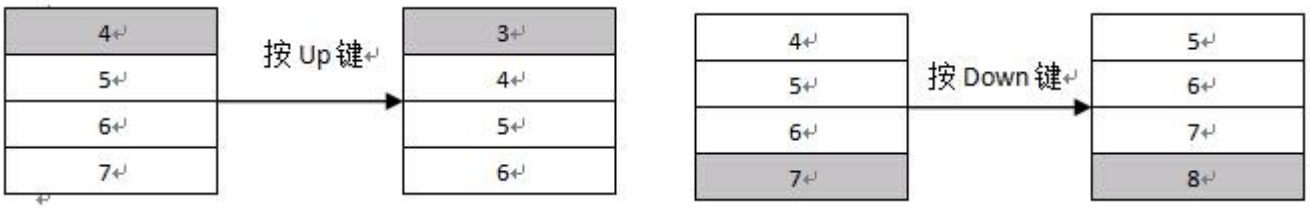


2. 歌曲总数大于 4 的时候（以一共有 10 首歌为例）：

特殊翻页：屏幕显示的是第一页（即显示第 1 – 4 首）时，光标在第一首歌曲上，用户按 Up 键后，屏幕要显示最后一页（即显示第 7-10 首歌），同时光标放到最后一首歌上。同样的，屏幕显示最后一页时，光标在最后一首歌曲上，用户按 Down 键，屏幕要显示第一页，光标挪到第一首歌上。



一般翻页：屏幕显示的不是第一页时，光标在当前屏幕显示的第一首歌曲时，用户按 Up 键后，屏幕从当前歌曲的上一首开始显示，光标也挪到上一首歌曲。光标当前屏幕的最后一首歌时的 Down 键处理也类似。



其他情况，不用翻页，只是挪动光标就行。

输入描述:
输入说明：
1 输入歌曲数量
2 输入命令 U 或者 D

本题含有多组输入数据！

输出描述:
输出说明
1 输出当前列表
2 输出当前选中歌曲

示例 1

输入

复制

```
10

UUUU
```

输出

复制

```
7 8 9 10

7
```

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    int n;
    string order;
    while(cin>>n>>order)//输入数量和命令
    {
        int num=1, first=1; //将 n 首歌曲编号 1：n , num 为光标当前所在歌曲的编号,first 为当前屏幕显示页的第一首歌曲的编号，初始化都为 1
        if(n<=4) //歌曲总数不超过 4 时，所有歌曲一页即可显示完，不需翻页，first 始终不变
        {
            for(int i=0;i<order.size();i++)
            {
                if( num==1 && order[i]=='U' ) num=n; //最后，往上就是最后一个
                else if( num==n && order[i]=='D' ) num=1;//最后，往下第一个
                else if(order[i]=='U') num--;//其他就是上下
                else    num++;
            }
            for(int i=1;i<=n-1;i++)
                cout<<i<<' ';//输出格式，要连在一起
            cout<<n<<endl;
            cout<<num<<endl;//最后在哪个
        }
        else   //歌曲总数大于 4 时，显示完全所有歌曲需要翻页，屏幕总是显示 4 首歌曲
        {
            for(int i=0;i<order.size();i++)
            {
                if( first==1 && num==1 && order[i]=='U' ) {first=n-3;num=n;} //特殊翻页 1 光标在 1,第一首歌在 1
                else if( first==n-3 && num==n && order[i]=='D' ) {first=1;num=1;} //特殊翻页 2，光标在 n,第一首歌在 n-3
                else if( first!=1 && num==first && order[i]=='U' ) {first--;num--;} //一般翻页 1 往上翻页
                else if( first!=n-3 && num==first+3 && order[i]=='D' ) {first++;num++;} //一般翻页 2  往下翻页
                else if( order[i]=='U' ) num--; //其他情况 1 光标变化
                else num++; //其他情况 2  光标变化
            }
            for(int i=first;i<first+3;i++)//第一个，一共有四个
                cout<<i<<' ';
            cout<<first+3<<endl;
            cout<<num<<endl;//当前的
```

```
            }
        }
        return 0;
}
```

55. 查找两个字符串最长子串

查找两个字符串 a,b 中的**最长公共子串。**若有多个，输出在较短串中最先出现的那个。

注：子串的定义：将一个字符串删去前缀和后缀（也可以不删）形成的字符串。请和“子序列”的概念分开!

本题含有多组输入数据!

输入描述:

输入两个字符串

输出描述:

返回重复出现的字符

示例 1

输入

复制

```
abcdefghijklmnop
```

```
abcsafjklmnopqrstuvw
```

输出

复制

```
Jklmnop
```

```
#include <iostream>
#include <string>

using namespace std;

int main(){
    string a,b;
    while(cin >> a >> b){
        if(a.size()>b.size()){
            swap(a,b);
        }
        string str_m;
        for(int i=0;i<a.size();i++){
            for(int j=i;j<a.size();j++){
                string tmp = a.substr(i,j-i+1);
                if(int(b.find(tmp))<0){
                    break;
                }
                else if(str_m.size()<tmp.size()){
                    str_m = tmp;
                }
            }
        }
        cout << str_m <<endl;
    }
    return 0;
}
while True:
    try:
```

```
        s1 = input()
        s2 = input()
        if len(s1) > len(s2):
            s1, s2 = s2, s1
        m, n = len(s1), len(s2)
        dp = [[0] * (n+1) for _ in range(m+1)]
        index, max_len = 0, 0
        for i in range(1, m+1):
            for j in range(1, n+1):
                if s1[i-1] == s2[j-1]:
                    dp[i][j] = dp[i-1][j-1] + 1
                    if dp[i][j] > max_len:
                        max_len = dp[i][j]
                        index = i
                else:
                    dp[i][j] = 0
        print(s1[index-max_len:index])
    except:
        Break
```

56. 24 点游戏算法

题目描述

问题描述：给出 4 个 1-10 的数字，通过加减乘除，得到数字为 24 就算胜利

输入：

4 个 1-10 的数字。[数字允许重复，但每个数字仅允许使用一次，测试用例保证无异常数字。

输出：

true or false

本题含有多组样例输入。

输入描述：

输入 4 个 int 整数
输出描述:
返回能否得到 24 点，能输出 true，不能输出 false

示例 1

输入

复制

```
7 2 1 10
```

输出

复制

```
true
```

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int flag=0;
void f(int num,double sum,vector <double> v1){
    if(flag==1){
        return;
    }
    if(sum==24){
        flag=1;
        return;
    }
    if(num>4){
        return;
    }
    for(int i=0;i<4;i++){
```

```
            switch(i){
                case 0:
                    f(num+1,sum+v1[num],v1);
                    break;
                case 1:
                    f(num+1,sum-v1[num],v1);
                    break;
                case 2:
                    f(num+1,sum*v1[num],v1);
                    break;
                case 3:
                    if(v1[num]!=0&&sum!=0){
                        f(num+1,sum/v1[num],v1);
                    }
                    break;
            }
        }

}
int main(){
    int a,b,c,d;
    while(cin>>a>>b>>c>>d){
        vector <double> temp;
        temp.push_back(a);
        temp.push_back(b);
        temp.push_back(c);
        temp.push_back(d);
        //sort(temp.begin(),temp.end());
        //int flag=0;
        do{
            f(1,temp[0],temp);
            if(flag==1){
                break;
            }
        }while(next_permutation(temp.begin(),temp.end()));
        if(flag) cout<<"true"<<endl;
        else cout<<"false"<<endl;
        //cout<<endl;
    }
}
```

57. 矩阵乘法

题目描述

如果 A 是个 x 行 y 列的矩阵，B 是个 y 行 z 列的矩阵，把 A 和 B 相乘，其结果将是另一个 x 行 z 列的矩阵 C。这个矩阵的每个元素是由下面的公式决定的

$$C_{ij} = \sum_{k=1}^{y} A_{ik} B_{kj}$$

矩阵的大小不超过 100*100

输入描述:

输入包含多组数据，每组数据包含：

第一行包含一个正整数 x，代表第一个矩阵的行数

第二行包含一个正整数 y，代表第一个矩阵的列数和第二个矩阵的行数

第三行包含一个正整数 z，代表第二个矩阵的列数

之后 x 行，每行 y 个整数，代表第一个矩阵的值

之后 y 行，每行 z 个整数，代表第二个矩阵的值

输出描述:

对于每组输入数据，输出 x 行，每行 z 个整数，代表两个矩阵相乘的结果

示例 1

输入

复制

```
2

3

2

1 2 3

3 2 1

1 2

2 1

3 3
```

输出

复制

```
14 13

10 11
```

```
#include<iostream>
#include<vector>
using namespace std;
int main(){
    int x, y, z;
    while (cin >> x >> y >> z){
        vector<vector<int>> arr1(x, vector<int>(y, 0));
        vector<vector<int>> arr2(y, vector<int>(z, 0));
        vector<vector<int>> arr3(x, vector<int>(z, 0));
        for(int i = 0; i < x; ++i){
            for(int j = 0; j < y; ++j)
                cin >> arr1[i][j];
        }
        for(int i = 0; i < y; ++i){
            for(int j = 0; j < z; ++j)
                cin >> arr2[i][j];
        }
        for(int i = 0; i < x; ++i){
            for(int j = 0; j < y; ++j)
                for(int k = 0; k < z; ++k)
                    arr3[i][k] += arr1[i][j] * arr2[j][k];
        }
        for(int i = 0; i < x; ++i){
            for(int j = 0; j < z-1; ++j)
                cout << arr3[i][j] << " ";
            cout << arr3[i][z-1] << endl;
        }
    }
    return 0;
}
```

58. 矩阵乘法计算量估量

题目描述

矩阵乘法的运算量与矩阵乘法的顺序强相关。

例如：

A 是一个 50×10 的矩阵，B 是 10×20 的矩阵，C 是 20×5 的矩阵

计算 A*B*C 有两种顺序：（（AB）C）或者（A（BC）），前者需要计算 15000 次乘法，后者只需要 3500 次。

编写程序计算不同的计算顺序需要进行的乘法次数。

本题含有多组样例输入！

输入描述：
输入多行，先输入要计算乘法的矩阵个数 n，每个矩阵的行数，列数，总共 2n 的数，最后输入要计算的法则
计算的法则为一个字符串，仅由左右括号和大写字母（'A'~'Z'）组成，保证括号是匹配的且输入合法！
输出描述：
输出需要进行的乘法次数

示例 1

输入

复制

```
3

50 10

10 20

20 5

(A(BC))
```

输出

复制

```
3500
```

```
#include<iostream>
#include<vector>
#include<string>
using namespace std;
int main(){
    int n;
    string s;
    while(cin>>n){
        int a[n][2];
        for(int i=0;i<n;++i)
            cin>>a[i][0]>>a[i][1];
        int k=0,sum=0;
        int p=0,q=0;
```

```
            vector<int> vec;
            cin>>s;
            for(int i=0;i<s.length();++i){
                if(s[i]!=')'){
                    if(s[i]=='('
                        p++;
                    else
                        vec.push_back(k++);
                }
                else{
                    if(++q>p)break;   //测试用例中有‘)’个数多于‘(’个数的情况，故加入该判断语句。
                    int y=vec.back();
                    vec.pop_back();
                    int x=vec.back();
                    vec.pop_back();
                    sum+=a[x][0]*a[x][1]*a[y][1];
                    a[x][1]=a[y][1];
                    vec.push_back(x);
                }
            }
            cout<<sum<<endl;
        }
        return 0;
}
```

59. 字符串通配符

题目描述

问题描述：在计算机中，通配符一种特殊语法，广泛应用于文件搜索、数据库、正则表达式等领域。现要求各位实现字符串通配符的算法。

要求：

实现如下 2 个通配符：

*：匹配 0 个或以上的字符（字符由英文字母和数字 0-9 组成，不区分大小写。下同）

？：匹配 1 个字符

输入：

通配符表达式；

一组字符串。

输出：

返回匹配的结果，正确输出 true，错误输出 false

本题含有多组样例输入！

输入描述：
先输入一个带有通配符的字符串，再输入一个需要匹配的字符串
输出描述：
返回匹配的结果，正确输出 `true`，错误输出 `false`

示例 1

输入

复制

```
te?t*.*
```

```
txt12.xls
```

输出

复制

```
false
```

```
#include <bits/stdc++.h>

bool bMatch(std::string reg, std::string s) {
	if((reg.size() == 0) || (0 == s.size())){
		if(reg.size() == 1 && reg[0] == '*') return true;
		return reg.size() == s.size();
	}
	char c = reg[0];
	if('?' == c || reg[0] == s[0]) {
		return bMatch(reg.substr(1), s.substr(1));
	} else if('*' == c) {
		return bMatch(reg, s.substr(1)) || bMatch(reg.substr(1), s);
	}
	return false;
}

int main()
{
	std::string reg, s;
	while(std::cin >> reg >> s) {
		std::transform(reg.begin(), reg.end(), reg.begin(),::toupper);
//以下是 std::transform 的两个声明，一个是对应于一元操作，一个是对应于二元操作：

template <class InputIterator, class OutputIterator, class UnaryOperation>
  OutputIterator transform (InputIterator first1, InputIterator last1,
原文链接：https://blog.csdn.net/fengbingchun/article/details/63252470
		std::transform(s.begin(), s.end(), s.begin(), ::toupper);

		std::cout << (bMatch(reg,s) ? "true" : "false") << std::endl;
	}
}

#include<bits/stdc++.h>
#include<cstdio>
#include<cstdlib>
```

```
#include<iostream>
#include<cmath>
#include<vector>
#include<stack>
#include<cstring>
#include<algorithm>

using namespace std;

  int main()
{
	string s1,s2;
	while(cin>>s1>>s2){
		vector<vector<bool>> f(s1.length()+1,vector<bool>(s2.length()+1,false));
		f[0][0]=true;
		for (int i = 1; i<=s1.length(); ++i)
			for (int j = i; j<=s2.length(); ++j) {
				if (s1[i] == '?')
					f[i][j] = f[i-1][j-1];
				else if (s1[i] == '*')
					f[i][j] = f[i-1][j] || f[i-1][j-1] || f[i][j-1];
				else
					f[i][j] = f[i-1][j-1] && (s1[i] == s2[j]);
			}
		cout<<(f[s1.length()][s2.length()]?"true":"false")<<endl;
	}
	return 0;
}
```

60. 扑克牌大小

题目描述

扑克牌游戏大家应该都比较熟悉了，一副牌由 54 张组成，含 3~A、2 各 4 张，小王 1 张，大王 1 张。牌面从小到大用如下字符和字符串表示（其中，小写 joker 表示小王，大写 JOKER 表示大王）：

3 4 5 6 7 8 9 10 J Q K A 2 joker JOKER

输入两手牌，两手牌之间用"-"连接，每手牌的每张牌以空格分隔，"-"两边没有空格，如：4 4 4 4-joker JOKER。

请比较两手牌大小，输出较大的牌，如果不存在比较关系则输出 ERROR。

基本规则：

（1）输入每手牌可能是个子、对子、顺子（连续 5 张）、三个、炸弹（四个）和对王中的一种，不存在其他情况，由输入保证两手牌都是合法的，顺子已经从小到大排列；

（2）除了炸弹和对王可以和所有牌比较之外，其他类型的牌只能跟相同类型的存在比较关系（如，对子跟对子比较，三个跟三个比较），不考虑拆牌情况（如：将对子拆分成个子）；

（3）大小规则跟大家平时了解的常见规则相同，个子、对子、三个比较牌面大小；顺子比较最小牌大小；炸弹大于前面所有的牌，炸弹之间比较牌面大小；对王是最大的牌；

（4）输入的两手牌不会出现相等的情况。

输入描述：

输入两手牌，两手牌之间用"-"连接，每手牌的每张牌以空格分隔，"-"两边没有空格，如 4 4 4 4-joker JOKER。

输出描述：

输出两手牌中较大的那手，不含连接符，扑克牌顺序不变，仍以空格隔开；如果不存在比较关系则输出 ERROR。

示例 1

输入

复制

```
4 4 4 4-joker JOKER
```

输出

复制

```
joker JOKER
```

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

int main(){
	string str;
	while(getline(cin, str)){
		if(str.find("joker JOKER") != string::npos){
			cout << "joker JOKER" << endl;
			continue;
		}

		int pos = str.find('-');
		string s1 = str.substr(0, pos);
		string s2 = str.substr(pos + 1);

		int size1 = count(s1.begin(), s1.end(), ' ') + 1;
		int size2 = count(s2.begin(), s2.end(), ' ') + 1;

		string temp = "3 4 5 6 7 8 9 10 J Q K A 2";
		if(size1 != size2){
			if(size1 == 4){
				cout << s1 << endl;
			}
			else if(size2 == 4){
				cout << s2 << endl;
```

```
            }
            else{
                cout << "ERROR" << endl;
            }
        }
        else{
            string first1 = s1.substr(0, s1.find(' '));
            string first2 = s2.substr(0, s2.find(' '));

            if(temp.find(first1) < temp.find(first2)){
                cout << s2 << endl;
            }
            else{
                cout << s1 << endl;
            }
        }

    }

    return 0;
}

/*像这种没有太多算法的题其实比较类似于实际应用软件中大多数的代码
  而这种代码真的是越短越好吗？
  个人认为：思路清晰，简洁得当，风格一致，注释合理才是优雅的代码。
  评论排在第一的代码除了短，并不觉得有多好，很多情况都没考虑，
  但是由于用例简单能 AC 就算过，其实健壮性不高只能拿来比赛，
  如果是工程代码，一味地追求浓缩，对于一个团队来讲，
  你的代码只能你自己看了，不利于团队协作和维护          */
#include <iostream>
#include <sstream>
#include <vector>
#include <algorithm>
using namespace std;
vector<string> split(string str,char sep){
    stringstream ss(str);
    string temp;
    vector<string> res;
    while(getline(ss,temp,sep)){
        res.push_back(temp);
    }
    return res;
}
int judgePoker(vector<string> poker){
```

```
    int flag=-1;
    if(poker.size()==1){
            flag=0;//个子
    }
    else if(poker.size()==2){
        if(poker[0]==string("joker")||poker[1]==string("joker"))
            flag=5;//对王
        else
            flag=1;//普通对子
    }
    else if(poker.size()==3)
        flag=2;//三个
    else if(poker.size()==4)
        flag=3;//炸弹
    else if(poker.size()==5)
        flag=4;//顺子
    return flag;
}

int main(){
    string str;
    vector<string> table={"3","4","5","6","7","8","9","10",
                          "J","Q","K","A","2","joker","JOKER"};
    while(getline(cin,str)){
        int win=-1;//0 表示不能比，1 表示第一幅，2 表示第二幅
        vector<string> vec=split(str,'-');
        vector<string> poker1=split(vec[0],' ');
        vector<string> poker2=split(vec[1],' ');
        int flag1=-1,flag2=-1;
        flag1=judgePoker(poker1);
        flag2=judgePoker(poker2);
        if(flag1==5||flag2==5||flag1==3||flag2==3){
            if(flag1==5)//一方有对王
                win=1;
            else if(flag2==5)
                win=2;
            else if(flag1==flag2&&flag1==3){//都是炸弹
                auto it1=find(table.begin(),table.end(),poker1[0]);
                auto it2=find(table.begin(),table.end(),poker2[0]);
                if(it1<it2)
                    win=2;
                else
                    win=1;
            }
```

```
            else if(flag1==3&&flag2!=3)//只有一方有炸弹
                win=1;
            else if(flag1!=3&&flag2==3)
                win=2;
        }
        else if(flag1==flag2){
            auto it1=find(table.begin(),table.end(),poker1[0]);
            auto it2=find(table.begin(),table.end(),poker2[0]);
            if(it1<it2)
                win=2;
            else
                win=1;
        }
        else
            win=0;
        if(!win)
            cout<<"ERROR"<<endl;
        else if(win==1){
            int i=0;
            for(;i<poker1.size()-1;i++)
                cout<<poker1[i]<<" ";
            cout<<poker1[i]<<endl;
        }
        else if(win==2){
            int i=0;
            for(;i<poker2.size()-1;i++)
                cout<<poker2[i]<<" ";
            cout<<poker2[i]<<endl;
        }
    }
    return 0;
}
```

61. 合法 Ip

题目描述

现在 IPV4 下用一个 32 位无符号整数来表示，一般用点分方式来显示，点将 IP 地址分成 4 个部分，每个部分为 8 位，表示成一个无符号整数（因此不需要用正号出现），如 10.137.17.1，是我们非常熟悉的 IP 地址，一个 IP 地址串中**没有空格出现（因为要表示成一个 32 数字）。**

现在需要你用程序来判断 IP 是否合法。

注意本题有多组样例输入。

输入描述：
输入一个 ip 地址，保证是 xx.xx.xx.xx 的形式（xx 为整数）
输出描述：
返回判断的结果 YES or NO

示例 1

输入

复制

```
10.138.15.1

255.0.0.255

255.255.255.1000
```

输出

复制

```
YES

YES

NO
```

```
#include <iostream>
#include <string>
#include <vector>
#include <algorithm>
#include<iterator>
using namespace std;
int main() {
	string s;
	while (cin >> s) {
		int n = s.size(); vector<int> v;
		bool isValid = true;
		int num_i = count(s.begin(), s.end(), '.');
		if (num_i != 3)
			cout << "NO" << endl;
		for (int i = 0; i < 4; i++) {
			int index = s.find('.');
			v.push_back(stoi(s.substr(0, index)));
			s = s.substr(index + 1);
		}
		for (int i = 0; i < v.size(); i++) {
			if (v[i] < 0 || v[i]>255)
				isValid = false;
		}
		if (isValid)
			cout << "YES" << endl;
		else
			cout << "NO" << endl;
```

```
    }
}
```

62. 表示数字

题目描述

将一个字符中所有的整数前后加上符号“*”，其他字符保持不变。连续的数字视为一个整数。

注意：本题有多组样例输入。

输入描述：
输入一个字符串
输出描述：
字符中所有出现的数字前后加上符号“*”，其他字符保持不变

示例 1

输入

复制

```
Jkdi234klowe90a3
```

```
5151
```

输出

复制

```
Jkdi*234*klowe*90*a*3*


*5151*
```

```
#include <iostream>
#include <string>
using namespace std;
int main(){
	string s;
	while(cin>>s){
		for(int i=0;i<s.length();i++){
			if(s[i]>='0'&&s[i]<='9'&&(i==0||s[i-1]<'0'||s[i-1]>'9')){
				s=s.substr(0,i)+"*"+s.substr(i);
				i++;
			}
			if(s[i]>='0'&&s[i]<='9'&&(i+1==s.length()||s[i+1]<'0'||s[i+1]>'9')){
				s=s.substr(0,i+1)+"*"+s.substr(i+1);
				i++;
			}
		}
		cout<<s<<endl;
	}
	return 0;
}
```

63. 自动售货系统

题目描述

1 总体说明

考生需要模拟实现一个简单的自动售货系统，实现投币、购买商品、退币、查询库存商品及存钱盒信息的功能。

系统初始化时自动售货机中商品为 6 种商品,商品的单价参见 1.1 规格说明,存钱盒内放置 1 元、2 元、5 元、10 元钱币，商品数量和钱币张数通过初始化命令设置，参见 2.1 系统初始化。

1.1 规格说明

1. 商品:每种商品包含商品名称、单价、数量三种属性，其中商品名不重复。考生不能修改商品名称和单价，初始化命令设置商品数量。这些信息在考试框架中进行定义，考生在实现功能代码时可直接使用。

商品　名称	单价	
A1	2	
A2	3	
A3	4	
A4	5	
A5	8	
A6	6	

2. 存钱盒信息：钱币面额、张数两种属性。初始化命令设置各种面额钱币张数。这些信息在考试框架中进行定义，考生在实现功能代码时可直接使用。

钱币面额	张数
10 元	X
5 元	X
2 元	X
1 元	X

3. **退币原则**：

1) 根据系统存钱盒内钱币的 信息 ， 按钱币总张数最少的原则进行退币。

2) 如果因零钱不足导致不能退币，则尽最大可能退币，以减少用户损失。

例如：假设存钱盒内只有 4 张 2 元，无其它面额钱币。如果需要退币 7 元，系统因零钱不足无法退币，则继续尝试退币 6 元，最终系统成功退币 3 张 2 元,用户损失 1 元钱币。

4. 投币操作说明：每次投币成功，投入的钱币面额累加到投币余额；同时，本次投入的钱币放入存钱盒中，存钱盒相应面额钱币增加。

5. 投币余额：指当前自动售货机中用户剩余的可购买商品的钱币总额；例如：投入 2 元面额的钱币，投币余额增加 2 元；购买一件价格 2 元的商品，投币余额减少 2 元；

6. 投币余额约束：投币余额不能超过 10 元。

7. 退币操作说明：退币操作需要遵守 **退币原则** ；退币成功后，投币余额清零，同时扣除存钱盒相应的金额。

8. 购买商品操作说明：一次仅允许购买一件商品；购买商品成功后，自动售货机中对应商品数量减 1，投币余额扣除本次购买商品的价格。

2 操作说明

命令字与第一个参数间使用一个空格分隔，多条命令采用分号隔开。考试系统会对输入命令格式进行处理，考生不需要关注输入**命令格式**的合法性，只需要实现命令处理函数。

2.1 系统初始化

命令格式:

r A1 数量 -A2 数量 -A3 数量 -A4 数量 -A5 数量 -A6 数量 1 元张数 -2 元张数 -5 元张数 -10 元张数

参数名称	参数说明	类型	
A1 数量	商品 A1 数量	整数	
A2 数量	商品 A2 数量	整数	
A3 数量	商品 A3 数量	整数	
A4 数量	商品 A4 数量	整数	
A5 数量	商品 A5 数量	整数	
A6 数量	商品 A6 数量	整数	
1 元张数	面额 1 元钱币张数	整数	
2 元张数	面额 2 元钱币张数	整数	
5 元张数	面额 5 元钱币张数	整数	
10 元张数	面额 10 元钱币张数	整数	

商品和各种面额钱币取值范围只是作为初始化命令的限制，其它场景下不限制取值范围；考试框架已经实现取值范围的检查，考生不需要关注。

功能说明：设置自动售货机中商品数量和存钱盒各种面额的钱币张数；

约束说明：系统在任意阶段均可执行 **r** 初始化系统；考生不需要关注参数的合法性，不需要关注增加或缺少参数的场景；

输出说明：输出操作成功提示（执行完 **r** 命令后系统会自动输出操作结果，考生不需要再次调用输出函数），例：

命令	输出
r 6-5-4-3-2-1 4-3-2-1;	S001:Initialization is successful

2.2 投币

命令格式：**p 钱币面额**

功能说明：

（1） 如果投入非 1 元、2 元、5 元、10 元的钱币面额（钱币面额不考虑负数、字符等非正整数的情况），输出“E002:Denomination error”；

（2） 如果存钱盒中 1 元和 2 元面额钱币总额小于本次投入的钱币面额，输出“E003:Change is not enough, pay fail”，但投入 1 元和 2 元面额钱币不受此限制。

（3） 如果自动售货机中商品全部销售完毕，投币失败。输出“E005:All the goods sold out”；

（4） 如果投币成功，输出“S002:Pay success,balance=X”；

约束说明：

（1） 系统在任意阶段都可以投币；

（2） 一次投币只能投一张钱币；

（3） 同等条件下，错误码的优先级：E002 > E003 > E005；

输出说明：如果投币成功，输出“S002:Pay success,balance=X”。

例:

命令	输出
p 10;	S002:Pay success,balance=10

2.3 购买商品

命令格式：**b 商品名称**

功能说明:

（1） 如果购买的商品不在商品列表中，输出“E006:Goods does not exist”；

（2） 如果所购买的商品的数量为 0，输出“E007:The goods sold out”；

（3） 如果投币余额小于待购买商品价格，输出“E008:Lack of balance”；

（4） 如果购买成功，输出“S003:Buy success,balance=X”；

约束说明:

（1） 一次购买操作仅能购买一件商品，可以多次购买；

（2） 同等条件下，错误码的优先级：E006 > E007 > E008；

输出说明:

如果购买成功，输出“S003:Buy success,balance=X”。

例:

命令	输出
b A1;	S003:Buy success,balance=8

2.4 退币

命令格式：**c**

功能说明：

（1） 如果投币余额等于 0 的情况下，输出“E009:Work failure”；

（2） 如果投币余额大于 0 的情况下，按照 **退币原则** 进行“找零”，输出退币信息；

约束说明：

（1） 系统在任意阶段都可以退币；

（2） 退币方式必须按照 **退币原则** 进行退币；

输出说明：如果退币成功，按照 **退币原则** 输出退币信息。

例，退 5 元钱币:

命令	输出
c;	1 yuan coin number=0 2 yuan coin number=0 5 yuan coin number=1 10 yuan coin number=0

2.5 查询

命令格式：**q 查询类别**

功能说明：

（1） 查询自动售货机中商品信息，包含商品名称、单价、数量。**根据商品数量从大到小进行排序；商品数量相同时，按照商品名称的先后顺序进行排序** 。

例如：A1 的商品名称先于 A2 的商品名称，A2 的商品名称先于 A3 的商品名称。

（2） 查询存钱盒信息，包含各种面额钱币的张数；

（3） 查询类别如下表所示:

查询类别	查询内容
0	查询商品信息
1	查询存钱盒信息

如果“查询类别”参数错误，输出“E010:Parameter error”。“查询类别”参数错误时，不进行下面的处理；

输出说明：

“查询类别”为 0 时，输出自动售货机中所有商品信息（商品名称单价数量）例:

命令	输出
q 0;	A1 2 6 A2 3 5 A3 4 4 A4 5 3

	A5 8 2 A6 6 0

“查询类别”为 1 时，输出存钱盒信息（各种面额钱币的张数），格式固定。例：

命令	输出
q 1;	1 yuan coin number=4 2 yuan coin number=3 5 yuan coin number=2 10 yuan coin number=1

输入描述：
依照说明中的命令码格式输入命令。
输出描述：
输出执行结果

示例 1

输入

复制

```
r 22-18-21-21-7-20 3-23-10-6;c;q0;p 1;b A6;c;b A5;b A1;c;q1;p 5;


r 28-12-11-1-16-10 19-30-8-11;b A1;p 1;
```

输出

复制

```
S001:Initialization is successful

E009:Work failure

E010:Parameter error

S002:Pay success,balance=1

E008:Lack of balance

1 yuan coin number=1

2 yuan coin number=0

5 yuan coin number=0

10 yuan coin number=0

E008:Lack of balance

E008:Lack of balance

E009:Work failure

E010:Parameter error

S002:Pay success,balance=5

S001:Initialization is successful

E008:Lack of balance

S002:Pay success,balance=1
```

题解(3)

讨论(56)

通过的代码

笔记

纠错

收藏

```
//注意：1、E004:Pay the balance is beyond the scope biggest 指的是投币错误，我之前看出了balance 大于 10 报错，这样理解是错误的
//2、E009 和 E10 这两个输出后不要换行
#include<iostream>
#include<string>
#include<vector>
#include<sstream>
using namespace std;
const int price[6]={2,3,4,5,8,6};
vector<string> split(string str, const char c){//拆分输入字符串
	vector<string> sstr;
	istringstream iss(str);
	string seg="";
	while(getline(iss, seg , c ))
	{
		sstr.push_back(seg);
	}
	return sstr;
}
void judgeInit(string init,vector<vector<int>>&inf){//info[0]记录商品数量，info[1]记录钱币数量
	vector<string> two_init = split(init, ' ');//init："22-18-21-21-7-20 3-23-10-6"通过空格分开
	vector<string>first = split(two_init[0], '-');//two_init[0]："22-18-21-21-7-20"通过空格分开
	vector<string>second = split(two_init[1], '-');//two_init[1]:"3-23-10-6"通过空格分开
	int i = 0;
	for (auto e : first){
		int n = atoi(e.c_str());//atoi 把 char 转为 int；e.c_str()为指向 e 的指针
		inf[0][i++] = n;
	}
	i = 0;
	for (auto e : second){
		int n = atoi(e.c_str());
		inf[1][i++] = n;
	}
	cout << "S001:Initialization is successful" << endl;
```

```
}
bool goodsSoldOut(const vector<vector<int>>inf){
	for (int i = 0; i < 6; i++){
		if (inf[0][i] > 0){
			return false;
		}
	}
	return true;
}
void judgePay(int num, int &balance, vector<vector<int>>&inf){//num 为硬币面值
	if (num != 1 && num != 2 && num != 5 && num != 10){
		cout << "E002:Denomination error" << endl;
		return;
	}
	else if (inf[1][0]+inf[1][1]*2 < num){
		cout << "E003:Change is not enough, pay fail" << endl;
		return;
	}
	else if (num > 10){//投币大于 10 元
		cout << "E004:Pay the balance is beyond the scope biggest" << endl;
		return;
	}
	else if (goodsSoldOut(inf)){
		cout << "E005:All the goods sold out" << endl;
		return;
	}
	else{
		switch (num){
		case 1:inf[1][0]++; balance += num; break;
		case 2:inf[1][1]++; balance += num; break;
		case 5:inf[1][2]++; balance += num; break;
		case 10:inf[1][3]++; balance += num; break;
		}
		cout << "S002:Pay success,balance=" << balance << endl;
		return;
	}

}
void judgeBuy(string good, int& balance, vector<vector<int>>&inf){
	//买商品，balance 以及相应的商品数量会发生相应的变化。
	if (good != "A1" && good != "A2" && good != "A3" && good != "A4" && good != "A5" &&
good != "A6"){
		cout << "E006:Goods does not exist" << endl;
		return;
```

```
    }
    else {
        if (inf[0][good[1]-'1'] == 0){//商品没了
            cout << "E007:The goods sold out" << endl;
        }
        else if (balance < price[good[1]-'1']){//钱不够了
            cout << "E008:Lack of balance" << endl;
        }
        else{//商品还有，钱也够，那就找零钱吧，所以要更新 balance 的数量，商品的数量，钱币的数量在退币的时候发生变化
            balance -= price[good[1]-'1'];
            inf[0][good[1]-'1']--;
            cout << "S003:Buy success,balance=" << balance << endl;
        }
        return;
    }
}
void refound(int&balance, vector<vector<int>>&inf){
    if (balance <= 0){
        cout << "E009:Work failure"<<endl;//注意这里不用换行
        return;
    }
    else{
        int a10 = 0,a5=0,a2=0,a1=0;
        while (balance>0){
            if (balance >= 10){
                if (inf[1][3] > 0){
                    inf[1][3]--;
                    balance -= 10;
                    a10++;
                }
                else if (inf[0][2] > 0){
                    inf[1][2]--;
                    balance -= 5;
                    a5++;
                }
                else if (inf[1][1] > 0){
                    inf[1][1]--;
                    balance -= 2;
                    a2++;
                }
                else if (inf[0][0] > 0){
                    inf[1][0]--;
                    balance -= 1;
```

```
            a1++;
        }
        else{
            balance = 0;
            break;
        }
    }
    else if(balance>=5){
        if (inf[1][2] > 0){//先判断有没有 5 元硬币
            inf[1][2]--;
            balance -= 5;
            a5++;
        }
        else if (inf[0][1] > 0){//再判断有没有 2 元硬币
            inf[1][1]--;
            balance -= 2;
            a2++;
        }
        else if (inf[1][0] > 0){//再判断有没有 1 元硬币
            inf[1][0]--;
            balance -= 1;
            a1++;
        }
        else{//再判断是否终止退币
            balance = 0;
            break;
        }
    }
    else if (balance >= 2){
        if (inf[1][1] > 0){//再判断有没有 2 元硬币
            inf[1][1]--;
            balance -= 2;
            a2++;
        }
        else if (inf[1][0] > 0){//再判断有没有 1 元硬币
            inf[1][0]--;
            balance -= 1;
            a1++;
        }
        else{//再判断是否终止退币
            balance = 0;
            break;
        }
    }
```

```
                else{//balance >= 1
                    if (inf[1][0] > 0){//再判断有没有 1 元硬币
                        inf[1][0]--;
                        balance -= 1;
                        a1++;
                    }
                    else{//再判断是否终止退币
                        balance = 0;
                        break;
                    }
                }
            }
            cout << "1 yuan coin number=" <<a1<< endl;
            cout << "2 yuan coin number=" <<a2<< endl;
            cout << "5 yuan coin number=" <<a5<< endl;
            cout << "10 yuan coin number=" <<a10<< endl;
        }
}
void query(string flag, vector<vector<int>>&inf){
        if (flag == "0"){
            cout << "A1 2 " << inf[0][0] << endl;
            cout << "A2 3 " << inf[0][1] << endl;
            cout << "A3 4 " << inf[0][2] << endl;
            cout << "A4 5 " << inf[0][3] << endl;
            cout << "A5 8 " << inf[0][4] << endl;
            cout << "A6 6 " << inf[0][5] << endl;
        }
        else if (flag == "1"){
            cout << "1 yuan coin number=" << inf[1][0] << endl;
            cout << "2 yuan coin number=" << inf[1][1] << endl;
            cout << "5 yuan coin number=" << inf[1][2] << endl;
            cout << "10 yuan coin number=" << inf[1][3] << endl;
        }
        else{
            cout << "E010:Parameter error"<<endl;//注意这个输出不用换行
        }
}
int main(){
        string inputs;
        while (getline(cin, inputs)){
            vector<string>ss1 = split(inputs, ';');//拆分输入字符串
            vector<vector<int>>info(2,vector<int>(6,0));//info[0]存储 A1-A6 的饮料数量， info[1]
存储 1,2,5,10 元纸币的数量
            int balance = 0;
```

```
            for (auto e : ss1){
                if (e[0] == 'r'){//初始化
                    judgeInit(e.substr(2), info);//初始化成功
                }
                else if (e[0] == 'p'){//投入钱币,注意 balance 和 info[1]会因为投币而发生相应变
化
                    int pay = atoi(e.substr(2).c_str());//投入钱数
                    judgePay(pay, balance, info);
                }
                else if (e[0] == 'b'){//买物品
                    string good = e.substr(2);
                    judgeBuy(good, balance, info);
                }
                else if (e[0] == 'c'){//退钱
                    refound(balance, info);
                }
                else if (e[0] == 'q'){//查询商品
                    query(e.substr(2), info);
                }

            }
        }
        return 0;
}
```

64. 自守数

题目描述

自守数是指一个数的**平方的尾数等于该数自身的自然数**。例如：25^2 = 625，76^2 = 5776，9376^2 = 87909376。请求出 n 以内的自守数的个数

接口说明

/*

功能: 求出 n 以内的自守数的个数

输入参数:

int n

返回值:

n 以内自守数的数量。

*/

public static int CalcAutomorphicNumbers(int n)

{

/*在这里实现功能*/

return 0;

}

本题有多组输入数据，请使用 while(cin>>)等方式处理

输入描述:

`int` 型整数

输出描述:
n 以内自守数的数量。

示例 1

输入

复制

```
2000
```

输出

复制

```
8
```

```
#include<iostream>
using namespace std;
class Solution{
public:
    static int CalcAutomorphicNumbers(int number){
        int i = 0, index;
        int count = 0;
        while(i <= number){
            index = i;
            int pow = index * index;
            while(index){
                if(pow%10 != index%10)
                    break;
                index = index / 10;
                pow = pow/10;
            }
            if(index == 0)
                count++;
```

```
                i++;
            }
            return count;
        }
};
int main()
{
    int n;
    Solution sol;
    while(cin >> n)
        cout << sol.CalcAutomorphicNumbers(n) << endl;
    return 0;
}

/*
本文系「人工智能安全」（微信公众号）原创，转载请联系本文作者（同博客作者）。
欢迎你转发分享至朋友圈，并给予「关注、星标、点赞」三连支持。互相欣赏，互相批判。
我是一名有诗人气质的网络安全工程师
期待与你的思想交流碰撞出智慧的花火
水木清华
2020-03-10
求自守数的解题思路
规律：个位数为 0、1、5、6 的数才可能是自守数，故采用筛选法，只判断符合该条件的数
思路 1：可以把整数（数及其平方）转换为字符串，通过比较长字符串的末尾是否与短字符串相同即可
如：25 * 25 = 625，字符串'625'的末尾'25'与字符串'25'的相同
思路 2：若该数的平方与该数的差，去模该数对应的各个进制位均等于零，则该数为自守数
如：25 * 25 = 625，625 - 25 = 600，600 % (10*1) = 0，600 % （10 * 2） = 0
*/
#include <iostream>
#include <string>
using namespace std;
//求整数 n 以内自守数的接口
int CalcAutomorphicNumbers(int n)
{
    int count = 0;
    for(int i = 0; i<= n; i++)
    {
        //仅对个位数符合条件的数执行自守数的判断
        if((i%10 == 0) || (i%10 == 1) || (i%10 == 5) || (i%10 == 6))
        {
            long j = i*i;
            string s1 = to_string(i);
            string s2 = to_string(j);
```

```
                int pos = s2.size()- s1.size();
                if(s2.find(s1,pos) != -1)
                {
                    count++;
                }
            }
        }
        return count;
}
int main()
{
        int n;
        while(cin >> n)
        {
            cout << CalcAutomorphicNumbers(n) << endl;
        }
        return 0;
}
```

65. 字符统计

题目描述

输入一个只包含小写英文字母和数字的字符串，按照不同字符统计个数由多到少输出统计结果，如果统计的个数相同，则按照 ASCII 码由小到大排序输出。

本题含有多组样例输入

输入描述：
一个只包含小写英文字母和数字的字符串。
输出描述：
一个字符串，为不同字母出现次数的降序表示。若出现次数相同，则按 ASCII 码的升序输出。

示例 1

输入

复制

```
aaddccdc
```

```
1b1bbbbbbbbb
```

输出

复制

```
cda
```

```
b1
```

说明

第一个样例里，c 和 d 出现 3 次，a 出现 2 次，但 c 的 ASCII 码比 d 小，所以先输出 c，再输出 d，最后输出 a.

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

int main()
{
    string strA;
    while(cin>>strA)
    {
```

```
        int nlen = strA.length();
        vector<int> vNum(128,0);//总共 128

        for(int i =0;i<nlen;i++)
        {
            vNum[strA[i]]++;
        }

        for(int j =48;j<128;j++)//48 就是数字
        {
            int MaxTimes = 0;
            int index;
            for(int k =48;k<128;k++)
            {
                if(vNum[k]>MaxTimes)
                {
                    MaxTimes = vNum[k];//最大数
                    index = k;//就是索引
                }

            }

            if(MaxTimes>0)
            {
                char c = index;
                 cout<<c;

            }
              vNum[index] = 0;//用完返回 0
        }
        cout<<endl;
    }
    return 0;
}
```

66. Redraiment 走法

题目描述

Redraiment 是走梅花桩的高手。Redraiment 可以选择任意一个起点，从前到后，但只能从**低处往高处的桩子走**。他希望走的步数最多，你能替 Redraiment 研究他最多走的步数吗?

本题含有多组样例输入

输入描述:
输入多行，先输入数组的个数，再输入相应个数的整数
输出描述:
输出结果

示例 1

输入

复制

```
6

2 5 1 5 4 5

3

3 2 1
```

输出

复制

```
3


1
```

说明

```
6 个点的高度各为 2 5 1 5 4 5

如从第 1 格开始走,最多为 3 步, 2 4 5

从第 2 格开始走,最多只有 1 步,5

而从第 3 格开始走最多有 3 步,1 4 5

从第 5 格开始走最多有 2 步,4 5

所以这个结果是 3。
```

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

using namespace std;

void find(vector<int> num){
```

```
    int n = (int)num.size();
    vector<int> lis(n,1);
    int max = 0;

    for(int i = 1; i < n; i++){
        for(int j = 0; j < i; j++){
            if(num[i] > num[j] && lis[i] < lis[j] + 1){
                lis[i] = lis[j] + 1;
            }
        }
        if(max < lis[i]){
            max = lis[i];
        }
    }
    cout<<max<<endl;

}


int main(int argc, const char * argv[]) {
    int n;
    while(cin>>n){
        vector<int> num(n);
        for(int i = 0; i < n; i++){
            cin>>num[i];
        }
        find(num);
    }
    return 0;
}
```

67. 求解立方根

题目描述

计算一个数字的立方根，不使用库函数。

保留一位小数。

输入描述:
待求解参数，为 double 类型（一个实数）
输出描述:
输入参数的立方根。保留一位小数。

示例 1

输入

复制

```
216
```

输出

复制

```
6.0
```

```
#include <iostream>
#include <iomanip>
#include<cmath>

using namespace std;
double calculate(double n) {
	double low = 0;
	double high = n + 1;
	double mid;
```

```
        while (low <= high) {
            mid = low + (high - low) / 2;
            if (abs((pow(mid,3) - n)) < 0.001) {
                break;
            } else if ((pow(mid,3) - n) > 0) {
                high = mid;
            } else if ((pow(mid,3) - n) < 0) {
                low = mid;
            }
        }
        return mid;
}


int main() {
        double num;
        double res;
        cin >> num;
        if (num > 0) {
            res = calculate(num);
        } else {
            res = -1 * calculate(-num);
        }
        cout << fixed << setprecision(1) << res;
        return 0;
}
```

68. 字符串的最后长度

题目描述

计算字符串最后一个单词的长度，单词以空格隔开，字符串长度小于 5000。

输入描述:
输入一行，代表要计算的字符串，非空，长度小于 5000。
输出描述:
输出一个整数，表示输入字符串最后一个单词的长度。

示例 1

输入

复制

```
hello nowcoder
```

输出

复制

```
8
```

说明

最后一个单词为 nowcoder，长度为 8

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
	string str;
	getline(cin,str);
	int count=0;
	int len=str.length();
	for(int i=(len-1);i>=0;i--)//从尾部开始
	{
		if(str[i]!=' ')
		{
		count++;
		}
		else
		{
			break;
		}
```

```
        }
        cout<<count<<endl;

        return 0;
}
```

69. 计算某字母出现次数

题目描述

写出一个程序，接受一个由字母、数字和空格组成的字符串，和一个字母，然后输出输入字符串中该字母的出现次数。不区分大小写，字符串长度小于 500。

输入描述:
第一行输入一个由字母和数字以及空格组成的字符串，第二行输入一个字母。
输出描述:
输出输入字符串中含有该字符的个数。

示例 1

输入

复制

```
ABCabc

A
```

输出

复制

```
2
```

```
#include<iostream>
#include <string>
#include <algorithm>

using namespace std;

int main()
{
	string s;
	getline(cin,s);
	char c;
	cin>>c;

	int count=0;
	for(auto i : s)
	{
		if(tolower(i)==tolower(c))//都是小写
			count++;
	}

	cout<<count<<endl;


	return 0;
}
```

70. 明明的随机数

题目描述

明明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了N 个 1 到 1000 之间的随机整数（N≤1000），对于其中重复的数字，只保留一个，把其余相同的数去掉，不同的数对应着不同的学生的学号。然后再把这些数从小到大排序，按照排好的顺序去找同学做调查。请你协助明明完成“去重”与“排序”的工作(同一个测试用例里可能会有多组数据(用于不同的调查)，希望大家能正确处理)。

注：测试用例保证输入参数的正确性，答题者无需验证。测试用例不止一组。

当没有新的输入时，说明输入结束。

输入描述：
注意：输入可能有多组数据(用于不同的调查)。每组数据都包括多行，第一行先输入随机整数的个数 N，接下来的 N 行再输入相应个数的整数。具体格式请看下面的"示例"。
输出描述：
返回多行，处理后的结果

示例 1

输入

复制

```
3
2
2
1
11
```

```
10

20

40

32

67

40

20

89

300

400

15
```

输出

复制

```
1

2

10

15

20

32

40

67

89

300

400
```

说明

```
样例输入解释：
```

样例有两组测试

第一组是 3 个数字，分别是：2，2，1。

第二组是 11 个数字，分别是：10，20，40，32，67，40，20，89，300，400，15。

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

int main()
{
	int N,value;

	std::set<int> m_set;
	while(std::cin >> N)
	{
		m_set.clear();
		while(N--)
		{
			std::cin >> value;
			m_set.insert(value);
		}
		for (std::set<int>::iterator it = m_set.begin(); it != m_set.end(); it++)
			std::cout << *it << std::endl;
	}
	return 0;
}

#include <iostream>
using namespace std;
int main() {
	int N, n;
	while (cin >> N) {
		int a[1001] = { 0 };
		while (N--) {
			cin >> n;
			a[n] = 1;
		}
		for (int i = 0; i < 1001; i++)
			if (a[i])
```

```
                cout << i << endl;
        }
        return 0;
}
```

71. 字符串分离

题目描述

•连续输入字符串，请按**长度为 8 拆分每个字符串后输出**到新的字符串数组；

•长度不是 8 整数倍的字符串请在后面补数字 0，空字符串不处理。

输入描述:
连续输入字符串(输入多次,每个字符串长度小于 100)
输出描述:
输出到长度为 8 的新字符串数组

示例 1

输入

复制

```
abc


123456789
```

输出

复制

```
abc00000

12345678

90000000
```

```
#include<bits/stdc++.h>

using namespace std;

int main() {
	string str;
	while(cin >> str) {
		while(str.size() > 8) {
			cout << str.substr(0, 8) << endl;
			str = str.substr(8);//每次都取出 8 个字符
		}
		str.resize(8, '0');//不够用 0 填充
		cout << str << endl;
	}
	return 0;
}
```

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
	string src;
	while(cin>>src)
	{
		int len=src.length();
		for(int i=0;i<len/8;++i)
		{
		    if(len>=8)
				cout<<src.substr(i*8,8)<<endl;
		}
		int tail=len%8;
		if(tail)
		{

			string out=src.substr(len/8*8,tail);
			for(int i=tail;i<8;++i)out+='0';
```

```
                cout<<out<<endl;
            }
        }
        return 0;
}
```

72. 句子逆序

题目描述

将一个英文语句以单词为单位逆序排放。例如“I am a boy”，逆序排放后为

“boy a am I”

所有单词之间用一个空格隔开，语句中除了英文字母外，不再包含其他字符

输入描述:
输入一个英文语句，每个单词用空格隔开。保证**输入只包含空格和字母**。
输出描述:
得到逆序的句子

示例 1

输入

复制

```
I am a boy
```

输出

复制

```
boy a am I
```

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

using namespace std;

int main()
{
	string s;
	vector<string> d;
	while(cin>>s)
		d.push_back(s);
	auto it = d.rbegin();
	for(;it!=d.rend()-1;it++)
		cout<<*it<<" ";
	cout<<*it<<endl;//最后一个没有空格
	return 0;
}
```

73. 坐标移动

题目描述

开发一个坐标计算工具， A 表示向左移动，D 表示向右移动，W 表示向上移动，S 表示向下移动。从（0,0）点开始移动，从输入字符串里面读取一些坐标，并将最终输入结果输出到输出文件里面。

输入：

合法坐标为 A(或者 D 或者 W 或者 S) + 数字（两位以内）

坐标之间以;分隔。

非法坐标点需要进行丢弃。如 AA10； A1A； $%$； YAD; 等。

下面是一个简单的例子 如：

A10;S20;W10;D30;X;A1A;B10A11;;A10;

处理过程：

起点（0,0）

+ A10 = （-10,0）

+ S20 = (-10,-20)

+ W10 = (-10,-10)

+ D30 = (20,-10)

+ x = 无效

+ A1A = 无效

+ B10A11 = 无效

+ 一个空 不影响

+ A10 = (10,-10)

结果 （10， -10）

注意请处理多组输入输出

输入描述:
一行字符串
输出描述:
最终坐标，以逗号分隔

示例 1

输入

复制

```
A10;S20;W10;D30;X;A1A;B10A11;;A10;
```

输出

复制

```
10,-10
```

列表内容

1. cin>>
   用法 1：最基本，也是最常用的用法，输入一个数字：

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

main ()

{ int a,b; cin>>a>>b; cout<<a+b<<endl;
```

```
}
```

输入：2[回车]3[回车]
输出：5

注意:>> 是会过滤掉不可见字符（如 空格 回车，TAB 等）
cin>>noskipws>>input[j];//不想略过空白字符，那就使用 noskipws 流控制

用法 2：接受一个字符串，遇“空格”、“TAB”、“回车”都结束

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

main ()

{ char a[20]; cin>>a; cout<<a<<endl;

}
```

输入：jkljkljkl
输出：jkljkljkl

输入：jkljkl jkljkl //遇空格结束
输出：jkljkl

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

int main()
{
    string str;

    while(cin >> str)//将每个坐标拿进来
    {
        //初始化坐标
        int x = 0, y = 0;
        //存储单步操作
        vector<string> steps;

        //把字符串拆分
```

```
            int wordlen = 0;
            for(int i = 0; i < str.size(); ++i)
            {
                while(str[i] != ';')
                    wordlen ++, i ++;
                steps.push_back(str.substr(i - wordlen, wordlen));
                wordlen = 0;
            }
            //分解成功
            //for(auto x : steps) cout << x << endl;

            //对单个 steps 执行坐标变换
            for(int i = 0; i < steps.size(); ++i)
            {
                int num = 0;
                //长度 3    A10
                if(steps[i].length() == 3 && steps[i][1] <= '9' && steps[i][1] >= '0' &&    steps[i][2]
<= '9' && steps[i][2] >= '0')
                    num = (steps[i][1] - '0') * 10 + steps[i][2] - '0';//两位以内
                //长度 2    A5
                if(steps[i].length() == 2 && steps[i][1] <= '9' && steps[i][1] >= '0')
                    num = steps[i][1] - '0';//一位

                switch(steps[i][0])//ASDW  就是加减的问题了
                {
                    case 'A': x -= num; break;
                    case 'D': x += num; break;
                    case 'W': y += num; break;
                    case 'S': y -= num; break;
                    default: break;
                }
            }
            cout << x << ',' << y << endl;
        }

        return 0;
}
```

74. 密码验证合格程序

题目描述

密码要求:

1.长度超过 8 位

2.包括大小写字母.数字.其它符号,以上四种至少三种

3.不能有相同长度大于 2 的子串重复

输入描述:
一组或多组长度超过 2 的字符串。每组占一行
输出描述:
如果符合要求输出：OK，否则输出 NG

示例 1

输入

复制

```
021Abc9000

021Abc9Abc1

021ABC9000

021$bc9000
```

输出

复制

```
OK

NG

NG

OK
```

```
#include<iostream>
#include<string>
using namespace std;
int func1(string &n){//四个是否有
int t[4]={0};
for(int i=0;i<n.size();i++){
if(n[i]<='9'&&n[i]>='0')t[0]=1;
else if(n[i]<='z'&&n[i]>='a')t[1]=1;
else if(n[i]<='Z'&&n[i]>='A')t[2]=1;
else t[3]=1;
}
return t[0]+t[1]+t[2]+t[3];
}

bool func2(string &n){//相同长度子串
for(int i=0;i+3<n.size();i++){//每个 3 的长度，看看是否有相等的。
string d=n.substr(i,3);
if(n.find(d,i+3)!=-1)return true;//索引从 d 开始寻找
}
return false;
}
int main()
{
	string n;
	while(cin>>n){
if((n.size()<=8)||func1(n)<3||func2(n))
cout<<"NG"<<endl;
else cout<<"OK"<<endl;}
}
```

75. 删除字符串中出现次数最小的字符串

题目描述

实现删除字符串中出现次数最少的字符，若多个字符出现次数一样，则都删除。输出删除这些单词后的字符串，字符串中其它字符保持原来的顺序。

注意每个输入文件有多组输入，即多个字符串用回车隔开

输入描述:
字符串只包含小写英文字母，不考虑非法输入，输入的字符串长度小于等于 20 个字节。
输出描述:
删除字符串中出现次数最少的字符后的字符串。

示例 1

输入

复制

```
abcdd

aabcddd
```

输出

复制

```
dd
```

```
aaddd
```

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
	string str;
	while (cin >> str)
	{
		int a[26] = {0};
		for(int i = 0; i < str.size(); i++)
		{
			// 通过 - 'a' 来得到正确的序号，以满足刚好能在 a[26]的下标
			a[str[i] - 'a']++;
		}

		// 判断最小，获得 min 的值 100; //
		int min = a[str[0] - 'a'];
		for(int i = 0; i < str.size(); i++)
		{
			if(a[str[i] - 'a'] < min)
			{
				min = a[str[i] - 'a'];	//出现次数最小
			}
		}

		// 凡是比 min 大的成员就输出
		for(int i = 0; i < str.size(); i++)
		{
			if(a[str[i] - 'a'] > min)
			{
				cout << str[i];
			}
		}

		// 切记要最后输出回车
		cout << endl;
	}

	 return 0;
}
```

76. 合唱队

题目描述

计算最少出列多少位同学，使得剩下的同学排成合唱队形

说明：

N 位同学站成一排，音乐老师要请其中的(N-K)位同学出列，使得剩下的 K 位同学排成合唱队形。

合唱队形是指这样的一种队形：设 K 位同学从左到右依次编号为 1，2…，K，他们的身高分别为 T1，T2，…，TK， 则他们的身高满足存在 i（1<=i<=K）使得

T1<T2<……<Ti-1<Ti>Ti+1>……>TK。

你的任务是，已知所有 N 位同学的身高，计算最少需要几位同学出列，可以使得剩下的同学排成合唱队形。

注意不允许改变队列元素的先后顺序

请注意处理多组输入输出!

输入描述:
整数 N
输出描述:
最少需要几位同学出列

示例 1

输入

复制

```
8


186 186 150 200 160 130 197 200
```

输出

复制

```
4
```

最长子序列问题

先找到每一个位置 i 左侧的最长上升子序列长度 numL[i]：每一个位置左侧最长子序列长度等于其左侧比它小的所有位置的最长子序列长度中的最大值+1

再找到每一个位置 i 右侧的最长下降子序列长度 numR[i]：每一个位置右侧最长子序列长度等于其右侧比它小的所有位置的最长子序列长度中的最大值+1

然后求出所有位置的**最长序列长度=左侧最长子序列长度+右侧最长子序列长度**-1（因为该位置被算了两次，所以减 1）

然后用数目减去最长序列长度就是答案

```
#include<bits/stdc++.h>
using namespace std;
int main()
```

```
{
    int len;
    while(cin>>len)
    {
    vector<int> hight(len);
    vector<int> numL(len);
    vector<int> numR(len);
    for(int &a:hight)
        cin>>a;

    for(int i=0;i<len;i++)
    {
        for(int j=0;j<i;j++)
        {
            if(hight[i]>hight[j])
            {
                numL[i]=max(numL[i],numL[j]);
            }

        }
       numL[i]=numL[i]+1;
    }

    for(int i=len-1;i>=0;i--)
    {
        for(int j=len-1;j>i;j--)
        {
            if(hight[i]>hight[j])
            {
                numR[i]=max(numR[i],numR[j]);
            }

        }
       numR[i]=numR[i]+1;

    }
    int max=0;
    for(int i=0;i<len;i++)
    {

        if(max<numL[i]+numR[i]-1)
            max=numL[i]+numR[i]-1;
    }
```

```
        cout<<len-max<<endl;
        }
}
```

77. 数据分类处理

题目描述

信息社会，有海量的数据需要分析处理，比如公安局分析身份证号码、 QQ 用户、手机号码、银行帐号等信息及活动记录。

采集输入大数据和分类规则，通过大数据分类处理程序，将大数据分类输出。

请注意本题有多组输入用例。

输入描述:

一组输入整数序列 I 和一组规则整数序列 R，I 和 R 序列的第一个整数为序列的个数（个数不包含第一个整数）；整数范围为 0~0xFFFFFFFF，序列个数不限

输出描述:

从 R 依次中取出 R<i>，对 I 进行处理，找到满足条件的 I：

I 整数对应的数字需要连续包含 R<i>对应的数字。比如 R<i>为 23，I 为 231，那么 I 包含了 R<i>，条件满足 。

按 R<i>从小到大的顺序：

(1)先输出 R<i>；

(2)再输出满足条件的 I 的个数；

(3)然后输出满足条件的 I 在 I 序列中的位置索引(从 0 开始)；

(4)最后再输出 I。

附加条件：

(1)R<i>需要从小到大排序。相同的 R<i>只需要输出索引小的以及满足条件的 I，索引大的需要过滤掉

(2)如果没有满足条件的 I，对应的 R<i>不用输出

(3)最后需要在输出序列的第一个整数位置记录后续整数序列的个数(不包含"个数"本身)

序列 I：

15,123,456,786,453,46,7,5,3,665,453456,745,456,786,453,123（第一个 15 表明后续有 15 个整数）

序列 R：5,6,3,6,3,0（第一个 5 表明后续有 5 个整数）

输出：

30, *3*,*6*,*0*,*123*,3,453,7,3,9,453456,13,453,14,123,6,7,1,456,2,786,4,46,8,665,9,453456,11,456,12,786

说明：

30----后续有 30 个整数

3----从小到大排序，第一个 R<i>为 0，但没有满足条件的 I，不输出 0，而下一个 R<i>是 3

6--- 存在 6 个包含 3 的 I

0--- 123 所在的原序号为 0

123--- 123 包含 3，满足条件

示例 1

输入

复制

```
15 123 456 786 453 46 7 5 3 665 453456 745 456 786 453 123


5 6 3 6 3 0
```

输出

复制

```
30 3 6 0 123 3 453 7 3 9 453456 13 453 14 123 6 7 1 456 2 786 4 46
8 665 9 453456 11 456 12 786
```

对 R 进行去重和排序，然后针对 R 中的元素，**遍历 I 是否包含该元素**，输出 I 中包含了的元素和对应的索引，全都不包含时不输出

好久发斯蒂芬

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

using namespace std;

typedef unsigned long uint32; // long int  保证至少 32 位

int main(){
    int n;
```

```
vector<string> I; // 用字符串存储，便于搜索
vector<uint32> R;
vector<uint32> output;
int I_sz, R_sz;

while(cin >> I_sz){
    I.clear();
    R.clear();      // 测试用例不止一组，因此每次都需要将 I, R 清空，或者将定义挪到这里
    output.clear();

    while(I_sz--){ // 存储 I
        cin >> n;
        I.push_back(to_string(n));
    }
    cin >> R_sz;
    while(R_sz--){ // 存储 R
        cin >> n;
        R.push_back(n);
    }

    // R 排序以及去重操作
    sort(R.begin(), R.end());
    auto unique_end = unique(R.begin(), R.end());
    R.erase(unique_end, R.end());

    for(int i = 0; i < R.size(); ++i){
        vector<uint32> tmp; // 临时 vec，如果 tmp 满足条件就放到 output
        for(int j = 0; j < I.size(); ++j){
            if(string::npos != I[j].find(to_string(R[i]))){ // to_string() c++11 支持
                tmp.push_back(j);//索引
                tmp.push_back(stoul(I[j])); // stoul()，将字符串转成 unsigned long
            }
        }
        if(!tmp.empty()){ // 在 I 中找到满足条件的 I<j>
            output.push_back(R[i]);//自身
            output.push_back(tmp.size() / 2); // 找到几个满足条件的 I[j]，几个，索引和素质
            output.insert(output.end(), tmp.begin(), tmp.end());
        }
    }

    if(!output.empty()){
        cout << output.size() << " ";//有几个
```

```
                for(int i = 0; i < output.size()-1; ++i){ // 注意最后一个数字后不跟空格，而是换行
                    cout << output[i] << " ";
                }
                cout << output[output.size()-1] << endl;
            }
        }

        return 0;
}
```

78. 字符串加密

题目描述

1、对输入的字符串进行加解密，并输出。

2、加密方法为：

当内容是英文字母时则用该英文字母的后一个字母替换，同时字母变换大小写,如字母 a 时则替换为 B；字母 Z 时则替换为 a；

当内容是数字时则把该数字加 1，如 0 替换 1，1 替换 2，9 替换 0；

其他字符不做变化。

3、解密方法为加密的逆过程。

本题含有多组样例输入。

输入描述：

输入说明
输入一串要加密的密码
输入一串加过密的密码
输出描述:
输出说明
输出加密后的字符
输出解密后的字符

示例 1

输入

复制

```
abcdefg


BCDEFGH
```

输出

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

void encode(string& s) {
	//注意这里修改字符串的话要用 & ， 引用
	for (auto & i : s) {
		//如果是数字
		if (isdigit(i)) {
			if (i == '9')i = '0';
			else
				i += 1;
		}
		else if (isupper(i)) {
```

```
            if (i == 'Z')i = 'a';
            else i = tolower(i) + 1;
        }
        else if (islower(i)) {
            if (i == 'z')       i = 'A';
            else i = toupper(i) + 1;//字母自己加，编程后面字母

        }
    }
}

void decode(string& s) {
    for (auto & c : s) {
        if (isdigit(c)) {
            if (c == '0')       c = '9';
            else c -= 1;
        }
        else if (isupper(c)) {
            if (c == 'A')       c = 'z';
            else
                c = tolower(c) - 1;
        }
        else if (islower(c)) {
            if (c == 'a')       c = 'Z';
            else
                c = toupper(c) - 1;
        }
    }
}
int main() {
    string input;
    while (getline(cin, input)) {
        encode(input);
        cout << input << endl;
        string str;
        getline(cin, str);
        decode(str);
        cout << str << endl;
    }
    return 0;
}
```

79. 字符串合并处理

题目描述

按照指定规则对输入的字符串进行处理。

详细描述：

将输入的两个字符串合并。

对合并后的字符串进行排序，要求为：下标为奇数的字符和下标为偶数的字符分别从小到大排序。这里的下标意思是字符在字符串中的位置。

对排序后的字符串进行操作，如果字符为 ‘0’ —— ‘9’ 或者 ‘A’ —— ‘F’ 或者 ‘a’ —— ‘f’ ，则对他们所代表的 16 进制的数进行 BIT 倒序的操作，并转换为相应的大写字符。如字符为 ‘4’ ，为 0100b，则翻转后为 0010b，也就是 2。转换后的字符为 ‘2’ ；如字符为 ‘7’ ，为 0111b，则翻转后为 1110b，也就是 e。转换后的字符为大写 ‘E’ 。

举例：输入 str1 为"dec"，str2 为"fab"，合并为 “decfab” ，分别对 “dca” 和 “efb” 进行排序，排序后为 “abcedf” ，转换后为 “5D37BF”

注意本题含有多组样例输入

输入描述:
本题含有多组样例输入。每组样例输入两个字符串，用空格隔开。
输出描述:
输出转化后的结果。每组样例输出一行。

示例 1

输入

复制

```
dec fab
```

输出

复制

```
5D37BF
```

```
 char Intput[] = {"0123456789abcdefABCDEF"}; //输入参照字典（数字 + 大小写字母）
//    int Output[] = "084c2a6e195d3b7f5d3b7f"; //输出参照字典（小写）
    char Output[] = {"084C2A6E195D3B7F5D3B7F"}; //输出参照字典（数字 + 大写字母）
#include<iostream>
#include<string>
#include<algorithm>

using namespace std;

const string helper1 = "0123456789abcdefABCDEF";
const string helper2 = "084C2A6E195D3B7F5D3B7F";

int main(){
      string str1, str2;
```

```
    while(cin >> str1 >> str2){
        string s, s1, s2;
        s = str1 + str2;
        int len = s.size();
        for(int i = 0; i < len; ++i){
            if(i % 2 == 0)
                s1 += s[i];
            else
                s2 += s[i];
        }
        sort(s1.begin(), s1.end());//奇数和偶数排序
        sort(s2.begin(), s2.end());
        s.clear();
        for(int i = 0,j = 0, k = 0; i < len; ++i){
            if(i % 2 == 0)
                s += s1[j++];//复原
            else
                s += s2[k++];
        }
        for(int i = 0; i < len; ++i){
            int n = helper1.find(s[i]);//找到下标
            if(n != -1)
                s[i] = helper2[n];//存在就找到对应的
        }
        cout << s << endl;
    }
    return 0;
}
```

80. 密码截取

题目描述

Catcher 是 MCA 国的情报员，他工作时发现敌国会用一些对称的密码进行通信，比如像这些 ABBA，ABA，A，123321，但是他们有时会在开始或结束时加入一些无关的字符以防止别国破解。比如进行下列变化 ABBA->12ABBA,ABA->ABAKK,123321->51233214 。因为截获的串太长了，而且存在多种可能的情况（abaaab 可看作是 aba,或 baaab 的加密

形式），Cathcer 的工作量实在是太大了，他只能向电脑高手求助，你能帮 Catcher 找出最长的有效密码串吗?

本题含有多组样例输入。

输入描述:
输入一个字符串
输出描述:
返回有效密码串的最大长度

示例 1

输入

复制

```
ABBA
```

输出

复制

```
4
```

```
#include <iostream>
#include <algorithm>
#include <string>
```

```
#include <string.h>

/*
函数名：palindrome_passwd_len
输入参数：类型 char *s ,表示一个密码字符串;类型 int step=1 表示判断偶数回文，step=2 表示判断基数回文
输出参数：类型 int，表示回文密码长度
返回值：返回回文密码最大有效长度
*/
int palindrome_passwd_len(char *s, int step) {
	int len = strlen(s);
	int low = 0, high = 0;
	int max = 0;
	if (NULL == s) {
		return 0;
	}
	for (int i = 0; i < len; i++) {
		low = i;
		high = low + step;
		while (low >= 0 && high < len && (s[low] == s[high])) {
			low--;
			high++;
		}
		if ((high - low - 1) > max) {
			max = high - low - 1;
		}
	}
	return max;
}

int main(void) {
	char str[10000];
	int max = 0, len = 0;
	while (cin >> str) {
		len = palindrome_passwd_len(str, 1);
		max = palindrome_passwd_len(str, 2);
		if (len >= max) {
			printf("%d\n", len);
		} else {
			printf("%d\n", max);
		}
	}
	return 0;
}
```

```
#include<iostream>
#include<string>

using namespace std;
//1.设 i 为第二个，j 为第一个，对两者中间子串进行判断是否为回文子串；
//2.分成奇偶两个效果，进行判别两者是否以 i 为中心为回文子串。

// int main(){
//         string str;
//         int i, j;
//         while(cin >> str){
//                 int R = 1;
//                 for(i = 1; i < str.length(); i++){
//                         for(j = 1; i-j >= 0 && i+j < str.length() && str[i+j] == str[i-j]; j++);
//                         if(j*2-1 > R)
//                                 R = j*2 - 1;
//                         for(j = 0; i-1-j >= 0 && i+j < str.length() && str[i-1-j] == str[i+j]; j++);
//                         if(j*2 > R)
//                                 R = j*2;
//                 }
//                 cout << R << endl;
//         }
//         return 0;
// }

int main(){
        string str;
        while(cin >> str){
                int maxLen = 0;
                for(int i = 1; i < str.length(); i++){
                        //寻找以 i-1,i 为中点偶数长度的回文
                        int low = i-1, high = i;
                        while(low >= 0 && high < str.length() && str[low] == str[high]){
                                low--; high++;
                        }
                        if(high - low - 1 > maxLen)
                                maxLen = high - low -1;
                        //寻找以 i 为中心的奇数长度的回文
                        low = i- 1; high = i + 1;
                        while(low >= 0 && high < str.length() && str[low] == str[high]){
                                low--; high++;
                        }
```

```
                if(high - low - 1 > maxLen)
                    maxLen = high - low -1;
            }
            cout << maxLen << endl;
        }
        return 0;
}
```

81. 整数与 Ip 间的转换

题目描述

原理：ip 地址的每段可以看成是一个 0-255 的整数，把每段拆分成一个二进制形式组合起来，然后把这个二进制数转变成

一个长整数。

举例：一个 ip 地址为 10.0.3.193

每段数字　　相对应的二进制数

10　　00001010

0　　00000000

3　　00000011

193　　11000001

组合起来即为：00001010 00000000 00000011 11000001,转换为 10 进制数就是：

167773121，即该 IP 地址转换后的数字就是它了。

本题含有多组输入用例，每组用例需要你将一个 ip 地址转换为整数、将一个整数转换为 ip 地址。

输入描述:
输入
1 输入 IP 地址
2 输入 10 进制型的 IP 地址
输出描述:
输出
1 输出转换成 10 进制的 IP 地址
2 输出转换后的 IP 地址

示例 1

输入

复制

```
10.0.3.193

167969729
```

输出

复制

```
167773121

10.3.3.193
```

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

void test33(){
	string ip;
	long num;
	while (std::cin >> ip >> num) {
		vector<int> vec;
		string tmp;
		for (long i = 0; i < ip.length(); i++) {
			if (ip[i] != '.') {
				tmp += ip[i];
			}else{
				vec.push_back(atoi(tmp.c_str()));
				tmp.clear();
			}
			if (i == (ip.length()-1)){
				vec.push_back(atoi(tmp.c_str()));
				tmp.clear();
			}
		}
		long ipNum = 0;
		for (vector<int>::iterator it = vec.begin(); it != vec.end(); it++) {
			//std::cout << *it << " ";
			ipNum = ipNum * 256 + *it;
		}
		std::cout << ipNum << "\n";

		stack<int> st;
		while (num > 0) {
			st.push(num % 256);
			num /= 256;
		}
		while (st.size() > 0) {
			if (st.size() == 1) {
				std::cout << st.top() << "\n" ;
			}else
				std::cout << st.top() << ".";
			st.pop();
		}
```

```
        }
}

int main(int argc, const char * argv[]) {
        // insert code here...
        test33();

}
```

82. 判断两个 Ip 是否属于同一子网

题目描述

子网掩码是用来判断任意两台计算机的 IP 地址是否属于同一子网络的根据。

子网掩码与 IP 地址结构相同，是 32 位二进制数，其中网络号部分全为“1”和主机号部分全为“0”。利用子网掩码可以判断两台主机是否中同一子网中。若两台主机的 IP 地址分别与它们的子网掩码相“与”后的结果相同，则说明这两台主机在同一子网中。

示例：

I P 地址　192.168.0.1

子网掩码　255.255.255.0

转化为二进制进行运算：

I P 地址　11010000.10101000.00000000.00000001

子网掩码　11111111.11111111.11111111.00000000

AND 运算

11000000.10101000.00000000.00000000

转化为十进制后为：

192.168.0.0

I P 地址　192.168.0.254

子网掩码　255.255.255.0

转化为二进制进行运算：

I P 地址　11010000.10101000.00000000.11111110

子网掩码　11111111.11111111.11111111.00000000

AND 运算

11000000.10101000.00000000.00000000

转化为十进制后为：

192.168.0.0

通过以上对两台计算机 IP 地址与子网掩码的 AND 运算后，我们可以看到它运算结果是一样的。均为 192.168.0.0，所以这二台计算机可视为是同一子网络。

输入一个子网掩码以及两个 ip 地址，判断这两个 ip 地址是否是一个子网络。

若 IP 地址或子网掩码格式非法则输出 1，若 IP1 与 IP2 属于同一子网络输出 0，若 IP1 与 IP2 不属于同一子网络输出 2。

C 库函数 - strtok()

 C 标准库 - <string.h>

描述

C 库函数 **char *strtok(char *str, const char *delim)** 分解字符串 **str** 为一组字符串，**delim** 为分隔符。

声明

下面是 strtok() 函数的声明。

```
char *strtok(char *str, const char *delim)
```

参数

- **str** -- 要被分解成一组小字符串的字符串。
- **delim** -- 包含分隔符的 C 字符串。

返回值

该函数返回被分解的第一个子字符串，如果没有可检索的字符串，则返回一个空指针。

实例

下面的实例演示了 strtok() 函数的用法。

实例

```
#include <string.h> #include <stdio.h> int main () { char str[80] = "This is - www.runoob.com - website"; const char s[2] = "-"; char *token; /* 获取第一个子字符串 */ token = strtok(str, s); /* 继续获取其他的子字符串 */ while( token != NULL ) { printf( "%s\n", token ); token = strtok(NULL, s); } return(0); }
```

让我们编译并运行上面的程序，这将产生以下结果：

```
This is
```

```
www.runoob.com

website
```

```
#include<bits/stdc++.h>
using namespace std;
bool subLegal(long val)
{
	if ((val & 1) == 1) return false;
	int flag = 0;
	long tmp = 1;
	int times = 32;
	while (times--)
	{
		if ((val & tmp) && (flag == 0))
		{
			flag = 1;
		}
		else if (((val & tmp) == 0) && flag == 1)
		{
			return false;
		}
		tmp <<= 1;
	}
	if (flag == 0) return false;
	return true;
}
bool isLegal(string a)
{
	char tmp[101];
	strcpy(tmp, a.c_str());
	char *s = strtok(tmp, ".");
	int count = 0;
	while (s)
	{
		if (++count > 4) return false;
		for (int i = 0; i < strlen(s); i++)
		{
			if (s[i] < '0' || s[i] > '9') return false;
		}
		if (strlen(s) > 3 || strlen(s) == 0) return false;
		int val = atoi(s);
		if (val < 0 || val > 255) return false;
		s = strtok(NULL, ".");
```

```
        }
        if (count < 4) return false;
        return true;
}
long binary(string a)
{
        char tmp[101];
        strcpy(tmp, a.c_str());
        long ret = 0;
        int weight = 1 * pow(256, 3);
        char *s = strtok(tmp, ".");
        while (s)
        {
                ret += atoi(s) * weight;
                weight /= 256;
                s = strtok(NULL, ".");
        }
        return ret;
}
int main()
{
        string s1, s2, s3;
        while (cin >> s1)
        {
                cin >> s2 >> s3;
                if (!isLegal(s1) || !isLegal(s2) || !isLegal(s3) || !subLegal(binary(s1)))
                {
                        cout << 1 << endl;
                        continue;
                }
                if ((binary(s1) & binary(s2)) == (binary(s1) & binary(s3)))
                {
                        cout << 0 << endl;
                }
                else
                {
                        cout << 2 << endl;
                }
        }
}
```

83. 称砝码

题目描述

现有一组砝码，重量互不相等，分别为 m1,m2,m3...mn；

每种砝码对应的数量为 x1,x2,x3...xn。现在要用这些砝码去称物体的重量(放在同一侧)，问能称出多少种不同的重量。

注:

称重重量包括 0

输入描述:

```
输入包含多组测试数据。

对于每组测试数据：

第一行：n --- 砝码数(范围[1,10])

第二行：m1 m2 m3 ... mn --- 每个砝码的重量(范围[1,2000])

第三行：x1 x2 x3 .... xn --- 每个砝码的数量(范围[1,6])
```

输出描述:
利用给定的砝码可以称出的不同的重量数

示例 1

输入

复制

```
2

1 2

2 1
```

输出

复制

```
5
```

怎么去重，用 set 集合。

1. 首先根据输入顺序，将砝码用数字序列表示，例如 2 个 1g 和 1 个 2g，就用 1 1 2 的序列表示；
2. set 序列用来表示**加入当前砝码之前能产生的重量种类**；
3. set 初始化为{0}；当第一个 1g 砝码放入时，则 set 中需要插入原先 set 中的所有元素+1g 后的结果；即{0, 0+1};
4. 当第二个 1g 加入时，则 set 会插入{0+1, 1+1},就变成了{0, 1, 2};
5. 重复上述步骤加入所有砝码；则**最后 set 的大小即为能产生的重量种类**

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

int main()
{
	int n, a[10], tmp;//一共几个
	while (cin >> n)
	{
		vector<int> v;
		set<int> s;
		for(int i = 0; i < n; i++) cin >> a[i];//每个重量
		for(int i = 0; i < n; i++)
		{
			cin >> tmp;//每个有几个
			for(int j = 0; j < tmp; j++) v.push_back(a[i]);//全部排开
		}
		s.insert(0);
		for(int i = 0; i < v.size(); i++)
		{
			set<int> t(s);//用 s 初始化，之前的
			for(auto it = t.begin(); it != t.end(); it ++)
			{
				s.insert(*it + v[i]);//每个元素，加上进来的元素
			}
		}
		cout << s.size() << endl;包括 0
	}
	return 0;
}
```

84. 学英语

题目描述

Jessi 初学英语，为了快速读出一串数字，编写程序将数字转换成英文：

如 22：twenty two，123：one hundred and twenty three。

说明：

数字为正整数，长度不超过九位，不考虑小数，转化结果为英文小写；

输出格式为 twenty two；

非法数据请返回“error”；

关键字提示：and，billion，million，thousand，hundred。

本题含有多组输入数据。

输入描述：
输入一个 long 型整数
输出描述：
输出相应的英文写法

示例 1

输入

复制

```
2356
```

输出

复制

```
two thousand three hundred and fifty six
```

```
#include <iostream>
#include <string>

using namespace std;

/*
如 22：twenty two，123：one hundred and twenty three。
说明：
数字为正整数，长度不超过九位，不考虑小数，转化结果为英文小写；
输出格式为 twenty two；
非法数据请返回“error”；
关键字提示：and，billion，million，thousand，hundred。*/

string less1000(long num)
{
	string ret;
	string nums[9] = { "one", "two", "three", "four", "five", "six", "seven", "eight", "nine" };
	string twenty_ninety[8] = { "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty",
"ninety" };
	string ten_nineteen[10] = { "ten", "eleven", "twelve", "thirteen", "fourteen",
	"fifteen", "sixteen", "seventeen", "eighteen", "nineteen"};
	int b, t;
	if (num >= 100)
	{
		b = num / 100;
		num = num % 100;
		ret += (nums[b - 1] + " hundred");
		if(num > 0)
			ret += " and ";
	}
	if (num >= 20)
	{
		t = num / 10;
		num = num % 10;
//		if(!ret.empty())
```

```
//                  ret += " and ";
            ret += twenty_ninety[t - 2];
            if(num > 0)
                  ret += " ";
      }
      if (num >= 10)
      {
//          if (!ret.empty())
//                  ret += " ";
            ret += ten_nineteen[num - 10];
            return ret;
      }
      if (num > 0)
      {
            ret += nums[num - 1];
      }
      return ret;
}

string parse(long num)
{
      if (num <= 0 || num > 999999999)
            return "error";
      int b, m, t;
      string ret;
      if (num >= 1000000000)
      {
            b = num / 1000000000;
            num = num % 1000000000;
            ret += (less1000(b) + " billion");
      }
      if (num >= 1000000)
      {
            m = num / 1000000;
            num = num % 1000000;
            if (!ret.empty())
                  //    ret += " and ";
                  ret += " ";
            ret += (less1000(m) + " million");
      }
      if (num >= 1000)
      {
            t = num / 1000;
            num = num % 1000;
```

```
            if (!ret.empty())
                //    ret += " and ";
                ret += " ";
            ret += (less1000(t) + " thousand");
        }
        if (num > 0)
        {
            if (!ret.empty())
            //ret += " and ";
                ret += " ";
            ret += less1000(num);
        }

        return ret;
}

int main()
{
        long num;
        while (cin >> num)
            cout << parse(num) << endl;
}
```

85. 输出单向链表中的倒数第 k 个节点

题目描述

输入一个单向链表，输出该链表中倒数第 k 个结点，链表的倒数第 1 个结点为链表的尾指针。

链表结点定义如下：

```
struct ListNode

{

int       m_nKey;
```

ListNode* m_pNext;

};

正常返回倒数第 k 个结点指针，异常返回空指针

本题有多组样例输入。

输入描述:
输入说明
1 输入链表结点个数
2 输入链表的值
3 输入 k 的值
输出描述:
输出一个整数

示例 1

输入

复制

```
8
```

```
1 2 3 4 5 6 7 8


4
```

输出

复制

```
5
```

```
//#include <iostream>
#include <bits/stdc++.h>

using namespace std;

typedef struct ListNode{
    int key;
    ListNode* next;
}ListNode,*PNode;

int main(){
    int n;
    while(cin>>n){
        int k;
        PNode head = (PNode)malloc(sizeof(ListNode));
        PNode tail = head;
        PNode res = head;
        head->next=NULL;
        int temp;
        for(int i=0;i<n;i++){
            PNode t = (PNode)malloc(sizeof(ListNode));
            cin>>temp;
            t->key=temp;
            tail->next=t;
            t->next=NULL;
            tail=t;
        }
        cin>>k;
```

```
            if(k==0){
                cout<<0<<endl;
            }
            else{
                for(int i=0;i<=n-k;i++){
                    res = res->next;
                }
                cout<<res->key<<endl;
            }
        }
        return 0;
}
```

86. 输入 n 个整数，输出其中最小的 k 个

题目描述

输入 n 个整数，输出其中最小的 k 个。

本题有多组输入样例，请使用循环读入，比如 while(cin>>)等方式处理

输入描述:
第一行输入两个整数 n 和 k
第二行输入一个整数数组
输出描述:
输出一个从小到大排序的整数数组

示例 1

输入

复制

```
5 2
```

```
1 3 5 7 2
```

输出

复制

```
1 2
```

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

int main(){
	int num, n;
	while(cin>>num>>n){
		int a[num];
		for(int i=0;i<num;i++){
			cin>>a[i];
		}
		sort(a,a+num);
		for(int i=0;i<n-1;i++){
			cout<<a[i]<<' ';
		}
		cout<<a[n-1]<<endl;
	}
	return 0;
}
```

87. 成绩排序

题目描述

查找和排序

题目：输入任意（用户，成绩）序列，可以获得成绩从高到低或从低到高的排列,相同成绩都按先录入排列在前的规则处理。

例示：

jack　　70

peter　　96

Tom　　70

smith　　67

从高到低　成绩

peter　　96

jack　　70

Tom　　70

smith　　67

从低到高

smith　　67

jack　　70

Tom　　70

peter　　96

注：0 代表从高到低，1 代表从低到高

本题含有多组输入数据！

输入描述：
输入多行，先输入要排序的人的个数，然后分别输入他们的名字和成绩，以一个空格隔开
输出描述：
按照指定方式输出名字和成绩，名字和成绩之间以一个空格隔开

示例 1

输入

复制

```
3

0

fang 90

yang 50

ning 70
```

输出

复制

```
fang 90

ning 70

yang 50
```

```
#include<iostream>
#include<string>
#include<vector>
#include<algorithm>
using namespace std;
class user {
public:
	string name;
	int score;
};
bool cpmpare0(user a, user b) {
	return a.score    > b.score ;
}
bool cpmpare1(user a, user b) {
	return a.score < b.score;
}
int main() {

	int count,flag;
	while (cin>>count>>flag) {
		vector<user> biao;
		for (int i = 0; i < count; i++) {
			user temp;
			cin >> temp.name >> temp.score ;
			biao.push_back(temp);
		}
		if(flag)
			stable_sort(biao.begin(), biao.end(), cpmpare1);
		else
			stable_sort(biao.begin(), biao.end(), cpmpare0);
```

```
            for (int i = 0; i < count; i++) {
                cout << biao[i].name << " " << biao[i].score << endl;
            }
        }
}
```

88. 火车站进站

题目描述

给定一个正整数 N 代表火车数量，0<N<10，接下来输入火车入站的序列，一共 N 辆火车，每辆火车以数字 1-9 编号，火车站只有一个方向进出，同时停靠在火车站的列车中，只有后进站的出站了，先进站的才能出站。

要求输出所有火车出站的方案，以字典序排序输出。

输入描述:
有多组测试用例，每一组第一行输入一个正整数 N（0<n<10），第二行包括 n 个正整数，范围为 1 到 9。< span="">
输出描述:
输出以字典序从小到大排序的火车出站序列号，每个编号以空格隔开，每个输出序列换行，具体见 sample。

示例 1

输入

复制

```
3


1 2 3
```

输出

复制

```
1 2 3
```

```
1 3 2
```

```
2 1 3
```

```
2 3 1
```

```
3 2 1
```

说明

第一种方案：1 进、1 出、2 进、2 出、3 进、3 出

第二种方案：1 进、1 出、2 进、3 进、3 出、2 出

第三种方案：1 进、2 进、2 出、1 出、3 进、3 出

第四种方案：1 进、2 进、2 出、3 进、3 出、1 出

第五种方案：1 进、2 进、3 进、3 出、2 出、1 出

> 请注意，`[3,1,2]`这个序列是不可能实现的。

火车进站题目解析

题目解析：

我看题目的第一眼，理解是 1 2 3 都进去了，然后都出来了，突然 2 号车又回来了，先进站，然后到 1 号车回来了，1 进站，最后 3 进站，这样出站顺序一定有 3 1 2 了。但是答案不是这样的>_<，只能从答案反推题目完整题意。

题目漏了一个关键的条件：**火车站内只有 1 条铁轨！另外输入数据中的第二行指定了进站顺序，这样所有火车就不能想进站就进站了**，要听铁路局指挥安排！所以不能用排列公式直接计算 C(3, 3)，只能依次猜测出站的真正可能性，所以初始条件是 1 2 3 这个顺序就很重要了。

2020.11.8 修正：

之前理解不够完善，实际上题目中的火车站内只有 1 根铁轨，但**火车站外有多根铁轨**（不然的话整个铁轨就是一条直线了，火车顺序永远只有一种了，这不符合题意），且必须只能按照给定的顺序进站，比如示例 1，只能按照 1 2 3 的顺序进站，意即一定是 1 号车先进站。但是题目没有规定 **1 出站的时间点**，1 可以在 2 进站之后，等 2 出站了再出站；也可以在 2 进站之前先出站，这样第 **1 个出站的一定是 1 号车，顺序必定是 1 x x。**

所以用穷举列出示例 1 的所有情形：

1 单独先进出的情况，有两种：

1 先进站，再出站，然后 2 进站，3 进站，3 出战，2 出站，出站顺序：1 3 2

1 先进站，再出站，然后 2 进站，2 出战，3 进站，3 出站，出站顺序：1 2 3

1 2 先进站，然后 2 先出站的情况，也有两种：

1 先进站，2 进站，然后 2 出站，1 出站，3 进站，3 出站。出站顺序：2 1 3

1 先进站，2 进站，然后 2 出站，3 进站，3 出战，1 出站。出站顺序：2 3 1

1 2 3 依次全部进站的情况，只能依次出站，只有一种：

1 先进站，2 进站，3 进站，3 出站，2 出站，1 出站。出站顺序：3 2 1

只要满足一点：进站顺序必须满足输入数据的第二行，即永远是 1 2 3！

作为火车调度员，每次操作要么入站，要么出站，很明显，我们用递归就可以穷举出所有可能性：

```
// 火车进站
#include <iostream>
#include <vector>
#include <stack>
#include <deque>
#include <algorithm>

using namespace std;
vector<vector <int>> answer;

// 火车调度函数，每次调度只有两种情况：进站或出站
void schedule(deque<int> & in, vector<int> & out, stack<int> & station)
{
    if (in.empty() && station.empty()) {
        answer.push_back(out);
        return;
    }

    //进站分支，注意进站分支和出站分支是独立的，因为 schedule 是递归函数！
    if (!in.empty()) {
        // 保护现场
        auto temp = in.front();

        // 进站
        station.push(in.front());
        in.pop_front();
```

```
        schedule(in, out, station);

        // 还原现场
        in.push_front(temp);
        station.pop();
    }

    //出站分支，注意进站分支和出站分支是独立的，因为 schedule 是递归函数！
    if (!station.empty()) {
        // 保护现场
        auto temp = station.top();

        //出站
        out.push_back(station.top());
        station.pop();
        schedule(in, out, station);

        // 还原现场
        out.pop_back();
        station.push(temp);
    }
}

int main()
{
    auto n = 0;                 //火车数量
    auto train_num = 0;       //火车编号

    while (cin >> n) {
        stack<int> station;       // 站内火车栈表
        deque<int> in;              // 等待进站火车队列
        vector<int> out;       // 已出站火车队列

        for (auto i = n; i > 0; i--) {
            cin >> train_num;
            in.push_back(train_num);
        }

        schedule(in, out, station);
        sort(answer.begin(), answer.end());

        for (auto &i : answer) {
            for (auto &j : i) {
                cout << j << " ";
```

```
                }
                cout << endl;
            }
            answer.clear();
        }
        return 0;
}
```

89. 整形数组合并

题目描述

题目标题：

将两个整型数组按照升序合并，并且过滤掉重复数组元素。

输出时相邻两数之间没有空格。

请注意本题有多组样例。

输入描述：
输入说明，按下列顺序输入：
1 输入第一个数组的个数
2 输入第一个数组的数值
3 输入第二个数组的个数
4 输入第二个数组的数值
输出描述：
输出合并之后的数组

示例 1

输入

复制

```
3

1 2 5

4

-1 0 3 2
```

输出

复制

```
-101235
```

```
#include<iostream>
#include<vector>
#include<stdlib.h>
#include<time.h>

using namespace std;
void quicksort(vector<int>& temp,int left,int right){
    if(left>=right) return;
    int vip;
    srand(time(NULL));
    vip=rand()%(right-left+1)+left;
    swap(temp[left],temp[vip]);//注意 left 不要写成 0
    int v=temp[left];
```

```
    int le=left,ge=right,i=left+1;
    while(i<=ge){//这里不能用 for 循环，有的不需要 i++
        if(temp[i]<v){
            swap(temp[i],temp[le]);
            le++;i++;
        }
        else if(temp[i]>v){
            swap(temp[i],temp[ge]);
            ge--;
        }
        else
            i++;
    }
    quicksort(temp,left,le-1);
    quicksort(temp,ge+1,right);
}
int removerepeat(vector<int>& temp){
    if(temp.size()==1) return 0;
    int low=0,fast=1;//low 左边和本身是严格符合已去重的
    while(fast<temp.size()){
        if(temp[fast]!=temp[low]){
            low++;
            swap(temp[fast++],temp[low]);
        }
        else
            fast++;
    }
    return low;

}

int main(){
    int num1,num2;
    int a;
    while(cin>>num1){
        vector<int> temp;
        while(num1--){
            cin>>a;
            temp.push_back(a);
        }
        cin>>num2;
        while(num2--){
            cin>>a;
            temp.push_back(a);
```

```
            }
            quicksort(temp,0,temp.size()-1);//排序完成
            int pos=removerepeat(temp);//双指针去重完成
            for(int i=0;i<=pos;i++){
                cout<<temp[i];
            }
            cout<<endl;
        }


}
#include<iostream>
#include<set>
using namespace std;
int main(){
    int num1,num2;
    int temp;
    while(cin>>num1){
        set<int> myset;
        while(num1--){
            cin>>temp;
            myset.insert(temp);
        }
        cin>>num2;
        while(num2--){
            cin>>temp;
            myset.insert(temp);
        }
        for(auto i:myset)
            cout<<i;
        cout<<endl;
    }


}
```

90. 字符串字符匹配

题目描述

判断短字符串中的所有字符是否在长字符串中全部出现。

请注意本题有多组样例输入。

输入描述:
输入两个字符串。第一个为短字符串，第二个为长字符串。两个字符串均由小写字母组成。
输出描述:
如果短字符串的所有字符均在长字符串中出现过，则输出 `true`。否则输出 `false`。

示例 1

输入

复制

```
bc
abc
```

输出

复制

```
true
```

```
#include <iostream>
#include <string>
```

```
using namespace std;

int main()
{
	string str1 = "";
	string str2 = "";
	while(cin>>str1)
	{
		cin>>str2;

		int size1 = str1.length();
		int size2 = str2.length();
		int a1[256] = {0};
		int a2[256] = {0};
		for(int i = 0;i<size1;i++)
		{
			a1[int(str1.at(i) - 0x00)]++;
		}
		for(int i = 0;i<size2;i++)
		{
			a2[int(str2.at(i) - 0x00)]++;
		}
		bool flag = true;
		for(int i = 0;i<256;i++)
		{
			if((a1[i] > 0) && (a2[i] == 0))
			{
				flag = false;
				break;
			}
		}
		if(flag)
			cout<<"true"<<endl;
		else
			cout<<"false"<<endl;
	}
}
```

91. 字符串中找出连续最长的数字串

题目描述

输入一个字符串，返回其最长的数字子串，以及其长度。若有多个最长的数字子串，则将它们全部输出（按原字符串的相对位置）

本题含有多组样例输入。

输入描述:
输入一个字符串。
输出描述:
输出字符串中最长的数字字符串和它的长度，中间用逗号间隔。如果有相同长度的串，则要一块儿输出（中间不要输出空格）。

示例 1

输入

复制

```
abcd12345ed125ss123058789


a8a72a6a5yy98y65ee1r2
```

输出

复制

```
123058789,9


729865,2
```

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

int main()
{
	string str;
	while(cin>>str)
	{
		int maxlen=0;
		string maxstring="";
		int start=0;
		while(start<str.size()-1)
		{
			int end=start+1;
			if(isdigit(str[start]))
			{
				while(isdigit(str[end]))
				{
					end++;
				}
				if((end-start)>maxlen)
				{
					maxlen=end-start;
					maxstring=str.substr(start,end-start);
				}
				else if((end-start)==maxlen)
				{
					maxstring+=str.substr(start,end-start);
				}
				start=end;
			}
			else
			{
				start++;
			}
		}
		cout<<maxstring<<','<<maxlen<<endl;
	}
	return 0;
}
```

92. 数组分组

题目描述

输入 int 型数组，询问该数组能否分成两组，使得两组中各元素加起来的和相等，并且，所有 5 的倍数必须在其中一个组中，所有 3 的倍数在另一个组中（不包括 5 的倍数），能满足以上条件，输出 true；不满足时输出 false。

本题含有多组样例输入。

输入描述:
`第一行是数据个数，第二行是输入的数据`
输出描述:
`返回 true 或者 false`

示例 1

输入

复制

```
4

1 5 -5 1

3

3 5 8
```

输出

复制

```
true
```

说明

第一个样例：

第一组：5 -5 1

第二组：1

第二个样例：由于 3 和 5 不能放在同一组，所以不存在一种分法。

```
/*建立一个全排列来计算
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
int main()
	{
	int n;
	while(cin>>n)
		{
		int sum3=0,sum5=0,sum_other=0,sum=0;//sum_other 用于非 35 数的求和
		int temp;//临时变量用作输入
		bool flag=0;
		vector<int> v_Other;
		for(int i=0;i<n;i++)
			{
```

```
            cin>>temp;
            sum+=temp;
            if(temp%5==0)
                sum5+=temp;
            else if(temp%3==0)
                sum3+=temp;
            else
                v_Other.push_back(temp);
        }
        int abs35=abs(sum3-sum5);//计算差值，然后在 v_Other 中求和，绝对值等于 abs35
就行
        if(sum%2!=0)
            cout<<"false"<<endl;
        else//计算全排列
            {
            string s;
            for(int i=0;i<v_Other.size();i++)
                s+="+-";
            sort(s.begin(),s.end());//使用全排列是一定要排序
            do
            {
                sum_other=0;
                for(int i=0;i<v_Other.size();i++)
                    {
                    if(s[i]=='+')
                        sum_other+=v_Other[i];
                    else
                        sum_other-=v_Other[i];
                    }
                if(abs(sum_other)==abs35)
                {
                    cout<<"true"<<endl;
                    flag=1;
                    break;
                }
            }while(next_permutation(s.begin(),s.end()));
            if(flag==0)
                cout<<"false"<<endl;
        }
    }
    return 0;
}
*/
/*使用递归来做*/
```

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
bool judge(int i,int n,vector<int> &v1,int sum1,int sum2)
    {
    if(i==n)
        return abs(sum1)==sum2;
    else
        return judge(i+1,n,v1,sum1+v1[i],sum2)||judge(i+1,n,v1,sum1-v1[i],sum2);
}
int main()
    {
    int n;
    while(cin>>n)
        {
        int sum3=0,sum5=0,sum_other=0,sum=0;//sum_other 用于非 35 数的求和
        int temp;//临时变量用作输入
        bool flag=0;
        vector<int> v_Other;
        for(int i=0;i<n;i++)
            {
            cin>>temp;
            sum+=temp;
            if(temp%5==0)
                sum5+=temp;
            else if(temp%3==0)
                sum3+=temp;
            else
                v_Other.push_back(temp);
        }
        int abs35=abs(sum3-sum5);//计算差值，然后在 v_Other 中求和，绝对值等于 abs35
就行
        if(sum%2!=0)
            cout<<"false"<<endl;
        else
        {
            flag=judge(0,v_Other.size(),v_Other,0,abs35);
            if(flag)
                cout<<"true"<<endl;
            else
                cout<<"false"<<endl;
        }
    }
```

```
    return 0;
}
/*投机取巧算法
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int n,data;
    while(cin>>n)
    {
        int sum=0,sum3=0,sum5=0,temp=0,dis=0,sumOther=0,positive=0,negtive=0;
        for(int i=0;i<n;i++)
        {
            cin>>data;
            sum+=data;    //总的
            if(data%5==0)
                sum5+=data;
            else if(data%3==0)
                sum3+=data;
            else
            {
                sumOther+=data;
                if(data>0)
                positive+=data;
                else
                negtive+=data;
            }
        }
        dis=abs(sum5-sum3);
        if((sumOther-dis)%2==0)
        {
            temp=sum/2;

if(temp-sum5>=0&&positive>=temp-sum5||temp-sum5<0&&negtive<=temp-sum5)
                //只要差额部分满足就可以了，比如 sum5 的正差额可以用 positive 补齐，负差额可以用呢个 i 提 v 额补齐
                    cout<<"true"<<endl;
            else
                    cout<<"false"<<endl;
        }
        else
            cout<<"false"<<endl;
    }
```

}*/

93. 机票统计

题目描述

请实现一个计票统计系统。你会收到很多投票，其中有合法的也有不合法的，请统计每个候选人得票的数量以及不合法的票数。

本题有多组样例输入。

输入描述:
输入候选人的人数 n，第二行输入 n 个候选人的名字（均为大写字母的字符串），第三行输入投票人的人数，第四行输入投票。
输出描述:
按照输入的顺序，每行输出候选人的名字和得票数量，最后一行输出不合法的票数。

示例 1

输入

复制

```
4

A B C D

8
```

```
A D E CF A GG A B
```

输出

复制

```
A : 3

B : 1

C : 0

D : 1

Invalid : 3
```

```
#include <iostream>
#include <string>
#include <vector>
#include <unordered_map>

using namespace std;

int main() {

    int n, m,invalid=0;
    vector<string> names, votes;
    unordered_map<string, int> nam, vot;
    string name, vote;
    unordered_map <string, int> ::iterator ptr;
    while (cin >> n)
    {
        for (int i = 0; i < n; ++i) {
            cin >> name;
            names.push_back(name);
            nam.insert(make_pair(name, 0));
        }
```

```
            cin >> m;
            for (int i = 0; i < m; ++i) {
                cin >> vote;
                ptr = nam.find(vote);
                if (ptr == nam.end())
                    invalid += 1;
                else
                    ptr->second += 1;
            }
            for (vector<string>::iterator i = names.begin(); i != names.end(); ++i)
            {
                ptr = nam.find(*i);
                cout << (*i) << " : " << ptr->second << endl;
            }
            cout << "Invalid : " << invalid << endl;

            names.clear();
            votes.clear();
            nam.clear();
            vot.clear();
            invalid=0;
        }

        return 0;
}
```

94. 人民币转换

题目描述

考试题目和要点：

1、中文大写金额数字前应标明“人民币”字样。中文大写金额数字应用壹、贰、叁、肆、伍、陆、柒、捌、玖、拾、佰、仟、万、亿、元、角、分、零、整等字样填写。

2、中文大写金额数字到“元”为止的，在“元”之后，应写“整字，如 532.00 应写成“人民币伍佰叁拾贰元整”。在”角“和”分“后面不写”整字。

3、阿拉伯数字中间有“0”时，中文大写要写“零”字，阿拉伯数字中间连续有几个“0”时，中文大写金额中间只写一个“零”字，如 6007.14，应写成“人民币陆仟零柒元壹角肆分“。

4、10 应写作“拾”，100 应写作“壹佰”。例如，1010.00 应写作“人民币壹仟零拾元整”，110.00 应写作“人民币壹佰拾元整”

5、十万以上的数字接千不用加“零”，例如，30105000.00 应写作“人民币叁仟零拾万伍仟元整”

本题含有多组样例输入。

输入描述:
输入一个 double 数
输出描述:
输出人民币格式

示例 1

输入

复制

```
151121.15
```

10012.02

```
#include<iostream>
#include<string>
#include<vector>
using namespace std;
const vector<string> helper1 = {"零","壹","贰","叁","肆","伍","陆","柒","捌","玖"};
const vector<string> helper2 = {"元", "万", "亿"};
const vector<string> helper3 = {"", "拾", "佰", "仟"};
string parts(int num){
	string str;
	if(num > 0 && num <= 9)
		str += helper1[num];
	else if(num >= 10 && num <= 19){
		if(num % 10 == 0)
			str += helper3[1];
		else
			str += helper3[1] + helper1[num%10];
	}else if(num >= 20 && num <= 99){
		if(num % 10 == 0)
			str += helper1[num/10] + helper3[1];
		else
			str += helper1[num/10] + helper3[1] + helper1[num%10];
	}else if(num >= 100 && num <= 999){
		if(num % 100 == 0)
			str += helper1[num/100] + helper3[2];
		else if(num % 100 <= 9)
			str += helper1[num/100] + helper1[0] + helper1[num%100];
		else
			str += helper1[num/100] + helper3[2] + parts(num % 100);
	}else if(num >= 1000 && num <= 9999){
		if(num % 1000 == 0)
			str += helper1[num/1000] + helper3[3];
		else if(num % 1000 <= 99)
			str += helper1[num/1000] + helper3[3] + helper1[0] + parts(num % 1000);
		else
			str += helper1[num/1000] + helper3[3] + parts(num % 1000);
	}
	return str;
}
int main(){
	double money;
	while (cin >> money){
```

```
            money += 0.0001; //  此处+0.0001 防止 double 转换 int 产生误差
            // 分两步，第一步处理整数
            int data = static_cast<int>(money);
            vector<int> vec;
            string res = "人民币";
            while (data){
                vec.push_back(data % 10000);
                data /= 10000;
            }
            for (int i = vec.size() - 1; i >= 0; --i){
                res += parts(vec[i]);
                res += helper2[i];
                if (i != 0 && i - 1 >= 0 && vec[i - 1] <= 999 && vec[i - 1] != 0)
                    res += helper1[0];
            }
            // 第二步处理小数
            int deci = static_cast<int>((money - static_cast<int>(money)) * 100);
            if (deci == 0)
                res += "整";
            else if (deci < 10)
                res += helper1[deci] + "分";
            else if (deci % 10 == 0)
                res += helper1[deci / 10] + "角";
            else
                res += helper1[deci / 10] + "角" + helper1[deci % 10] + "分";
            cout << res << endl;
        }
        return 0;
}
```

95. 记负均正

题目描述

首先输入要输入的整数个数 n，然后输入 n 个整数。输出为 n 个整数中负数的个数，和所有正整数的平均值，结果保留一位小数。

本题有多组输入用例。

输入描述：

首先输入一个正整数 n，
然后输入 n 个整数。

输出描述：

输出负数的个数，和所有正整数的平均值。

示例 1

输入

复制

```
5


1 2 3 4 5
```

输出

复制

```
0 3.0
```

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    // ifstream in("input.txt");
    // cin.rdbuf(in.rdbuf());
    int numVal;
    while (cin >> numVal)
    {
        int numNeg = 0, tmpNum;
        vector<double> vecPos;
```

```
            for (int i = 0; i < numVal; i++)
            {
                cin >> tmpNum;
                if (tmpNum < 0)
                    numNeg++;
                else if (tmpNum > 0)
                    vecPos.push_back(tmpNum);
            }
            double average = (accumulate(vecPos.begin(), vecPos.end(), 0.0) / vecPos.size());
            printf("%d %.1f\n", numNeg, average);
        }
}
```

96. 输入整型数组和排序标识，对其元素按照升序或降序进行排序

题目描述

输入整型数组和排序标识，对其元素按照升序或降序进行排序（一组测试用例可能会有多组数据）

本题有多组输入，请使用 while(cin>>)处理

输入描述:
第一行输入数组元素个数
第二行输入待排序的数组，每个数用空格隔开
第三行输入一个整数 0 或 1。0 代表升序排序，1 代表降序排序
输出描述:
输出排好序的数字

示例 1

输入

复制

```
8


1 2 4 9 3 55 64 25


0


5


1 2 3 4 5


1
```

输出

复制

```
1 2 3 4 9 25 55 64


5 4 3 2 1
```

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

bool AscCompare(int a, int b)
{
```

```
    return a < b;
}

bool DescCompare(int a, int b)
{
    return a > b;
}

int main(int argc, char **argv)
{
    int num;
    int method;
    while (cin >> num) {
        vector<int> nums;
        int data;
        for (int i = 0; i < num; i++) {
            cin >> data;
            nums.push_back(data);
        }
        cin >> method;
        if (method == 1) {
            sort(nums.begin(), nums.end(), DescCompare);
        } else {
            sort(nums.begin(), nums.end(), AscCompare);
        }
        for (int i = 0; i < num; i++) {
            cout << nums[i] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

97. 记负均正 2

题目描述

从输入任意个整型数，统计其中的负数个数并求所有非负数的平均值，结果保留一位小数，

如果没有非负数，则平均值为 0

本题有多组输入数据，输入到文件末尾，请使用 while(cin>>)读入

数据范围小于 1e6

输入描述:
输入任意个整数，每行输入一个。
输出描述:
输出负数个数以及所有非负数的平均值

示例 1

输入

复制

```
-13

-4

-7
```

输出

复制

```
3

0.0
```

```
#include<iostream>
#include<string.h>
#include<cstdio>
using namespace std;
int num,count1,count2;
double ave,sum;
int main(){
	while(cin>>num){
		if(num>=0){
			sum+=num;
			count1++;
		}else{
			count2++;
		}
	}
	if(sum!=0)
		ave=sum*1.0/count1;
	printf("%d\n%.1lf\n",count2,ave);
	return 0;
}
```

98. 识别有效的 IP 地址和掩码并进行

题目描述

请解析 IP 地址和对应的掩码，进行分类识别。要求按照 A/B/C/D/E 类地址归类，不合法的

地址和掩码单独归类。

所有的 IP 地址划分为 A,B,C,D,E 五类

A 类地址 1.0.0.0~126.255.255.255;

B 类地址 128.0.0.0~191.255.255.255;

C 类地址 192.0.0.0~223.255.255.255;

D 类地址 224.0.0.0~239.255.255.255；

E 类地址 240.0.0.0~255.255.255.255

私网 IP 范围是：

10.0.0.0～10.255.255.255

172.16.0.0～172.31.255.255

192.168.0.0～192.168.255.255

子网掩码为二进制下前面是连续的 1，然后全是 0。（例如：255.255.255.32 就是一个非法的掩码）

注意二进制下全是 1 或者全是 0 均为非法

注意：

1. 类似于【0.*.*.*】和【127.*.*.*】的 IP 地址不属于上述输入的任意一类，也不属于不合法 ip 地址，计数时可以忽略

2. 私有 IP 地址和 A,B,C,D,E 类地址是不冲突的

输入描述:
多行字符串。每行一个 IP 地址和掩码，用~隔开。
输出描述:
统计 A、B、C、D、E、错误 IP 地址或错误掩码、私有 IP 的个数，之间以空格隔开。

示例 1

输入

复制

```
10.70.44.68~255.254.255.0

1.0.0.1~255.0.0.0

192.168.0.2~255.255.255.0

19..0.~255.255.255.0
```

输出

复制

```
1 0 1 0 0 2 1
```

```
#include <stdio.h>
#include <stdlib.h>
int main(){
```

```
	char ip[1000];
	int dgt[8]={0},i=0,j=0,A=0,B=0,C=0,D=0,E=0,b,bflag,psn=0,uilg=0,flag=0;
	while(scanf("%s",ip)!=EOF){
		for(i=0;i<8;i++)dgt[i]=0;
		j=0;flag=0;bflag=0;
		for(i=0;ip[i]!='\0';i++){
			if(ip[i]==';'){flag=3;break;}
			if(ip[i]=='.'||ip[i]=='~'){flag++;j++;continue;}
			if(flag>=2){uilg++;flag=3;printf("%d ",dgt[i]);break;}
			if('0'<=ip[i]&&ip[i]<='9'){
			dgt[j]=dgt[j]*10+ip[i]-'0';
			flag=0;}
			else { uilg++;flag=3; break;}
		}
		    if(dgt[4]==0&&flag<2){ uilg++;flag=3; continue;}
		    else if(dgt[7]==255&&flag<2){ uilg++;flag=3; continue;}
		  else {
		for(i=7;i>=4;i--){
		   if(flag==3)break;
	  for(j=0;j<8;j++){
				b=1&(dgt[i]>>j);
			       // printf("b=%d %d ",b,dgt[i]);
				if(b)bflag=1;
				if(bflag&&!b){ uilg++;flag=3;break;}
			 }
		}
		  }
		  if(flag>=2)continue;
		if(dgt[0]==0||dgt[0]==127) continue;
	    //                printf("%d      %d      %d      %d      %d      %d      %d      %d
",dgt[0],dgt[1],dgt[2],dgt[3],dgt[4],dgt[5],dgt[6],dgt[7]);
		if(0<=dgt[0]&&dgt[0]<=126)A++;
		if(128<=dgt[0]&&dgt[0]<=191)B++;
		if(192<=dgt[0]&&dgt[0]<=223)C++;
		if(224<=dgt[0]&&dgt[0]<=239)D++;
		if(240<=dgt[0]&&dgt[0]<=255){E++;}
		if(dgt[0]==10&&0<=dgt[1]&&dgt[1]<=255){psn++;}
		if(dgt[0]==172&&16<=dgt[1]&&dgt[1]<=31){psn++;}
		if(dgt[0]==192&&dgt[1]==168) {psn++; }
	}
	printf("%d %d %d %d %d %d",A,B,C,D,E,uilg,psn);
	return 0;
}
```

<string.h>是旧的 C 头文件，对应的是基于 char*的字符串处理函数；

<string>是包装了 std 的 C++头文件，对应的是新的 string 类（看下文）；

<cstring>是对应于旧 C 头文件的 std 版本。（包含 std）

<string>是 c++ 的头文件，其内包含了一个 string 类，string s1 就是建立一个 string 类的对象

<string.h> 的 c 语言的东西 并无类，所以不能 string s1

<cstring>文件实际上只是在一个命名空间 std 中 include 了 <string.h>

99. 简单错误记录

题目描述

开发一个简单错误记录功能小模块，能够记录出错的代码所在的文件名称和行号。

处理：

1、 记录最多 8 条错误记录，循环记录，最后只用输出最后出现的八条错误记录。对相同的错误记录只记录一条，但是**错误计数增加。最后一个斜杠后面的带后缀名的部分（保留最后 16 位）和行号完全匹配的记录才做算是”相同“的错误记录。**

2、 超过 16 个字符的文件名称，只记录文件的最后有效 16 个字符；

3、 输入的文件可能带路径，记录文件名称不能带路径。

4、循环记录时，只以第一次出现的顺序为准，后面重复的不会更新它的出现时间，仍以第一次为准

输入描述:
每组只包含一个测试用例。一个测试用例包含一行或多行字符串。每行包括带路径文件名称，行号，以空格隔开。
输出描述:
将所有的记录统计并将结果输出，格式：文件名 代码行数 数目，一个空格隔开，如：

示例 1

输入

复制

```
D:\zwtymj\xccb\ljj\cqzlyaszjvlsjmkwoqijggmybr 645

E:\je\rzuwnjvnuz 633

C:\km\tgjwpb\gy\atl 637
```

```
F:\weioj\hadd\connsh\rwyfvzsopsuiqjnr 647

E:\ns\mfwj\wqkoki\eez 648

D:\cfmwafhhgeyawnool 649

E:\czt\opwip\osnll\c 637

G:\nt\f 633

F:\fop\ywzqaop 631

F:\yay\jc\ywzqaop 631
```

输出

复制

```
rzuwnjvnuz 633 1

atl 637 1

rwyfvzsopsuiqjnr 647 1

eez 648 1

fmwafhhgeyawnool 649 1

c 637 1
```

```
f 633 1


ywzqaop 631 2
```

其实就是统计指定文件的指定行号的数量，例如如果输入是
E:\V1R2\product\fpgadrive.c 1325
E:\V1R2\product\fpgadrive.c 1325
E:\V1R2\product\fpgadrive.c 1325
则输出是
fpgadrive.c 1325 3

"""

方法 1：

题目太绕了，重新捋了一下

```
//题目意思：
//给一系列字符串，没条保留：“最后一个斜杠后面的字母(超过 16 位时只保留最后 16 位) 数字行号”
//按顺序读取以上字符串，当保留的部分是相同的，则记录出现次数+1
//全部读取完后，输出字符串和出现次数。当保存记录超过 8 条时只输出最后 8 条

//思路：
//因为哈希表存储顺序和输入顺序无关，所以用 pair<string,int>保存字符串和出现次数
//字符串部分，把文件名和行号合并到一起，空格隔开，和输出形式一致
//每次遍历 vector，重复时+1，不重复时追加（用 vector 可以保持顺序）
#include<iostream>
#include<vector>
#include<sstream>
#include<utility>
using namespace std;
int main(){
    vector<pair<string,int>> mypair;//保存字符串（文件名 行号）和出现次数
    string s;
    while(getline(cin,s)){
        string a,b,temp;
        stringstream ss(s);
        ss>>a>>b;//通过流将空格左右分开
        for(int i=a.size()-1,count=0;a[i]!='\\';i--) {//遇斜杠退出双斜杠表示斜杠
            temp=a[i]+temp;//temp 保存文件名
```

```
            if(++count==16) break;//满 16 位退出
        }
        temp+=' '+b;//文件名加行号一起作为 pair 的 first 因为只有都一样才算重复
        bool flag=false;//标记文件名是否重复过
        for(int i=0;i<mypair.size();i++)
            if(mypair[i].first==temp){
                (mypair[i].second)++;//重复过，出现次数+1
                flag=true;
                break;
        }
        if(!flag) mypair.push_back({temp,1});//未重复，追加在后面（保持顺序）
    }

    if(mypair.size()>8){//如果记录的超过 8，只输出后 8 个
        for(int i=mypair.size()-8;i<mypair.size();i++)
            cout<<mypair[i].first<<' '<<mypair[i].second<<endl;
    }
    else{//未超过 8 条，按序输出即可
        for(auto i:mypair)
            cout<<i.first<<' '<<i.second<<endl;
    }
}
```

9
4
0
4
1
4
2
4
3
4
4
4
5

方法 2：

大致就是哈希表存储出现的次数，怎么样保持有序呢，就是每条记录第一次不重复出现时，push 进 vector 中。

这样的话，输出的时候，安装 vector 的顺序往出输出，次数在哈希表里找

比方法 1 快，不需要每次输入完都遍历查找是否重复过

复制代码

```
map<string,int> mymap;//保存字符串（文件名 行号）和出现次数
    vector<string> ans;
    string s;
    while(getline(cin,s)){
        string a,b,temp;
        stringstream ss(s);
        ss>>a>>b;
        a=a.substr(a.rfind('\\')+1);//写法更简单~
        if(a.size()>16)//判断斜杠后字母个数超过 16
            a=a.substr(a.size()-16);
        temp=a+' '+b;
        if(mymap.count(temp)) mymap[temp]++;//通过哈希表判断是否重复过
        else {
            mymap[temp]++;
            ans.push_back(temp);
        }
    }
```

```
    if(ans.size()>8){
        for(int i=ans.size()-8;i<ans.size();i++)
            cout<<ans[i]<<' '<<mymap[ans[i]]<<endl;
    }
    else{
        for(auto i:ans)
            cout<<i<<' '<<mymap[i]<<endl;
    }
```

100. 查找兄弟单词

题目描述

定义一个单词的“兄弟单词”为：交换该单词字母顺序，而不添加、删除、修改原有的字母就能生成的单词。

兄弟单词要求和原来的单词不同。例如：ab 和 ba 是兄弟单词。ab 和 ab 则不是兄弟单词。

现在给定你 n 个单词，另外再给你一个单词 str，让你寻找 str 的兄弟单词里，字典序第 k 大的那个单词是什么?

注意：字典中可能有重复单词。本题含有多组输入数据。

输入描述：

先输入单词的个数 n，再输入 n 个单词。

再输入一个单词，为待查找的单词 x

最后输入数字 k

输出描述：

输出查找到 x 的兄弟单词的个数 m

然后输出查找到的按照字典顺序排序后的第 k 个兄弟单词，没有符合第 k 个的话则不用输出。

示例 1

输入

复制

```
3 abc bca cab abc 1
```

输出

复制

```
2
```

bca

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

using namespace std;

bool is_brothers(string target,string candidate){
	sort(target.begin(),target.end());
	sort(candidate.begin(),candidate.end());//字典序

	return target==candidate;
}

int main(){
	int n;
	while(cin>>n){
		int i,k=0;
		vector<string> v(n);
		for(i=0;i<n;i++){
			cin>>v[i];
		}

		int ID;
		string target;
		cin>>target>>ID;

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

		string brother;
		for(i=0;i<n;i++){
			if(v[i]!=target){
				if(is_brothers(target,v[i])){
					if(++k==ID)
						brother=v[i];
				}
			}
		}
		cout<<k<<endl;
		if(k>=ID)
			cout<<brother<<endl;
	}
```

```
    return 0;
}
```

101. 素数伴侣

题目描述

题目描述

若两个正整数的和为素数，则这两个正整数称之为“素数伴侣”，如 2 和 5、6 和 13，它们能应用于通信加密。现在密码学会请你设计一个程序，从已有的 N（N 为偶数）个正整数中挑选出若干对组成“素数伴侣”，挑选方案多种多样，例如有 4 个正整数：2，5，6，13，如果将 5 和 6 分为一组中只能得到一组“素数伴侣”，而将 2 和 5、6 和 13 编组将得到两组“素数伴侣”，能组成“素数伴侣”最多的方案称为“最佳方案”，当然密码学会希望你寻找出“最佳方案”。

输入:

有一个正偶数 N（N≤100），表示待挑选的自然数的个数。后面给出具体的数字，范围为[2,30000]。

输出:

输出一个整数 K，表示你求得的“最佳方案”组成“素数伴侣”的对数。

输入描述:

```
输入说明
1 输入一个正偶数 n
2 输入 n 个整数
注意：数据可能有多组
```

输出描述：
求得的“最佳方案”组成“素数伴侣”的对数。

示例 1

输入

复制

```
4

2 5 6 13

2

3 6
```

输出

复制

```
2

0
```

贪心和**深搜**（题目：素数伴侣）

中心思想：将所有数分为奇数和偶数，分别找到每一个奇数（偶数）的素数伴侣，可能有多

个，都记录下来。先匹配伴侣少的奇数（偶数），再匹配伴侣多的奇数（偶数），这样就会找到尽可能多的素数伴侣对，组成素数伴侣的对数最多就是奇数（偶数）的个数（取决于奇数和偶数谁个数少）

步骤及示例：

#1.将所有数分为偶数 **even**(ArrayList)和奇数 **odd**(ArrayList)，分别从小到大排序（假设长度较小的为 odd）

#2.开辟一个长度和 odd 相同的 list 数组，list[i]存储：odd[i]的素数伴侣 even[j]的所有 j 值

#3.将 list 按照元素个数的多少从小到大排序（并且去掉奇数没有素数伴侣的项）

爱发火

匈牙利算法，最大匹配数

将数据分为奇数和偶数两组，进行匹配，利用匈牙利算法求出最大匹配，即为所求。

匈牙利算法，最大匹配数

将数据分为奇数和偶数两组，进行匹配，利用匈牙利算法求出最大匹配，即为所求。

复制代码

```
#include <bits/stdc++.h>
using namespace std;
//判断一个数是否为素数
bool IsPrimer(int n)
{
    for (int i = 2; i * i < n; i++)
    {
        if (n % i == 0)
        {
            return false;
        }
    }
    return true;
}
```

```cpp
//匈牙利算法，为某一个目标奇数找到它的素数伴侣（偶数）,皆以下标表示
bool FindMate(const int& n, vector<vector<bool>>& map, vector<int>& match,
vector<bool> &vis)
{
    for (int i = 0; i < match.size(); i++)
    {
        if (!vis[i] && map[i][n])//偶数未被访问过并且能与目标奇数组成素数（有关系）
        {
            vis[i] = true;
            if (match[i] == -1 || FindMate(match[i], map, match, vis))//当前偶
数没有匹配或能给被抛弃的奇数找到新的偶数
            {
                match[i] = n;//找到这个偶数
                return true;
            }
        }
    }
    return false;
}

int main()
{
    int num = 0;
    while (cin >> num)
    {
        int count = 0;//匹配对数
        vector<int> even;//偶数
        vector<int> odd;//奇数
        int val = 0;
        //读取数据
        while (num--)
        {
            cin >> val;
            if (val % 2 == 0)
            {
                odd.push_back(val);
            }
            else
            {
                even.push_back(val);
            }
        }
        if (odd.empty() || even.empty())
        {
```

```
            cout << count << endl;
            continue;
        }
        //建立关系表，图中的边
        vector<vector<bool>> map(even.size(), vector<bool>(odd.size(), false));
        for (int i = 0; i < even.size(); i++)
        {
            for (int j = 0; j < odd.size(); j++)
            {
                if (IsPrimer(even[i] + odd[j]))
                {
                    map[i][j] = true;
                }
            }
        }
        //建立初始匹配表
        vector<int> match(even.size(), -1);
        //为每一个奇数都尝试去找对应的偶数，
        for (int i = 0; i < odd.size(); i++)
        {
            vector<bool> vis(even.size(), false);//每一次查找都相当于重新分配，标志要清零
            if (FindMate(i, map, match, vis))
            {
                count++;
            }
        }
        cout << count << endl;
    }
}
```

6
5
7
5
8
5
9
6
0
6
1
6
2
6
3
6
4
6
5
6
6
6
7
6
8
6
9
7
0
7
1
7
2
7
3
7
4
7
5
7
6
7
7
7

8
7
9
8
0
8
1
8
2
8
3
8
4
8
5
发顺丰

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

bool find(vector<int>& odd_numbers, vector<pair<vector<int>,int>>& table, int tableIdx)
{
	int idex = table[tableIdx].second;
	int idex2;
	int second;
	int oddvalue;
	while(idex < table[tableIdx].first.size())
	{
		if(tableIdx == 20)
		{
			second = table[tableIdx].second;
			oddvalue = table[tableIdx].first[second];
		}
		idex2 = table[tableIdx].first[idex];
		if(odd_numbers[idex2]== -1 || find(odd_numbers, table, odd_numbers[idex2]))
		{
			second = table[tableIdx].second;
			odd_numbers[idex2] = tableIdx;
			table[tableIdx].second = idex+1;
			return true;
		}
		idex++;
	}
```

```
    return false;
}

int main()
{
    int maxprimeNum = sqrt(30000.0) + 1;
    vector<int> primeNumArray;
    int num;
    int n;
    for(int i=2; i< maxprimeNum; i++)
    {
        bool flag=true;
        for(auto iter = primeNumArray.begin(); iter!=primeNumArray.end(); iter++)
        {
            if(i%(*iter) == 0)
            {
                flag = false;
                break;
            }
        }
        if(flag)
            primeNumArray.push_back(i);

    }
    while(cin>>n)
    {
        vector<int> odd_numbers;
        vector<int> even_numbers;
        while(n--)
        {
            cin>>num;
            if(num%2 == 0)
                even_numbers.push_back(num);
            else
                odd_numbers.push_back(num);

        }
        vector<pair<vector<int>,int>> table(even_numbers.size());
        for(int i=0;i<even_numbers.size();i++)
        {
            table[i].first.reserve(odd_numbers.size());
            for(int j=0;j<odd_numbers.size();j++)
            {
                int x = even_numbers[i]+odd_numbers[j];
```

```
                    if([&] {
                        for(auto &elem:primeNumArray)
                        {
                            if(x<=elem)
                                break;
                            if(x%elem == 0)
                                return true;
                        }
                        return false;
                    }())
                        {}//table[i].push_back(0);
                    else
                        table[i].first.push_back(j);
                }
                table[i].second = 0;
            }
            for(auto& elem:odd_numbers)
            {
                elem = -1;
            }
            for(int i=0;i<table.size();i++)
            {
                if(table[i].first.size())
                    find(odd_numbers, table, i);
            }
            int idx=0;
            for(auto&elem:table)
            {
                if(elem.second != 0)
                    idx++;
            }
            cout<<idx<<endl;
        }
        return 0;
}
```

102. 单词倒排

题目描述

对字符串中的所有单词进行倒排。

说明：

1、构成单词的字符只有 26 个大写或小写英文字母；

2、非构成单词的字符均视为单词间隔符；

3、要求倒排后的单词间隔符以一个空格表示；如果原字符串中相邻单词间有多个间隔符时，

倒排转换后也只允许出现一个空格间隔符；

4、每个单词最长 20 个字母；

输入描述：
输入一行以空格来分隔的句子
输出描述：
输出句子的逆序

示例 1

输入

复制

```
I am a student
```

输出

复制

```
student a am I
```

单词倒排

1. **使用 getline 获取一行字符串**
2. **使用空格“ ”代替字符串中非字母数字；**
3. **使用 stringstream 对处理过后的数据进行逐个单词读取，会自动跳过空格；**
4. **使用 vector 储存每个单词，然后用 reverse 函数倒排；**
5. **输出单词和空格，注意最后一个单词没有空格，最后输出换行符。**

单词倒排

1. **使用 getline 获取一行字符串**
2. **使用空格“ ”代替字符串中非字母数字；**
3. **使用 stringstream 对处理过后的数据进行逐个单词读取，会自动跳过空格；**
4. **使用 vector 储存每个单词，然后用 reverse 函数倒排；**
5. **输出单词和空格，注意最后一个单词没有空格，最后输出换行符。**


```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    string line;
    while(getline(cin, line))
    {
        for(auto &c : line)
        {
            if(c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z')
            {
                continue;
            }
            else
            {
                c = ' ';
            }
```

```
        }
        stringstream ss(line);
        string word;
        vector<string> vec;
        while(ss >> word)
        {
            vec.push_back(word);
        }
        reverse(vec.begin(), vec.end());
        for(int i = 0; i < vec.size(); i++)
        {
            cout << vec[i];
            if(i != vec.size() - 1)
            {
                cout << " ";
            }
        }
        cout << endl;
    }
}
```

```
3
6
3
7
```

收起

103. 数独（Sudoku）

题目描述

问题描述：数独（Sudoku）是一款大众喜爱的数字逻辑游戏。玩家需要根据 9X9 盘面上的已知数字，推算出所有剩余空格的数字，并且满足每一行、每一列、每一个粗线宫内的数字均含 1-9，并且不重复。

输入：

包含已知数字的 9X9 盘面数组[空缺位以数字 0 表示]

输出：

完整的 9X9 盘面数组

输入描述：
包含已知数字的 9X9 盘面数组[空缺位以数字 0 表示]
输出描述：
完整的 9X9 盘面数组

示例 1

输入

复制

```
0 9 2 4 8 1 7 6 3
```

```
4 1 3 7 6 2 9 8 5

8 6 7 3 5 9 4 1 2

6 2 4 1 9 5 3 7 8

7 5 9 8 4 3 1 2 6

1 3 8 6 2 7 5 9 4

2 7 1 5 3 8 6 4 9

3 8 6 9 1 4 2 5 7

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

输出

复制

```
5 9 2 4 8 1 7 6 3

4 1 3 7 6 2 9 8 5

8 6 7 3 5 9 4 1 2

6 2 4 1 9 5 3 7 8
```

```
7 5 9 8 4 3 1 2 6

1 3 8 6 2 7 5 9 4

2 7 1 5 3 8 6 4 9

3 8 6 9 1 4 2 5 7

9 4 5 2 7 6 8 3 1
```

```
#include<iostream>

using namespace std;

int num[9][9];
bool sign=false;


bool Check(int n,int key){
	for(int i=0;i<9;i++){
		int j=n/9;
		if(num[j][i]==key) return false;
	}
	for(int i=0;i<9;i++){
		int j=n%9;
		if(num[i][j]==key) return false;
	}
	int x=n/9/3*3;
	int y=n%9/3*3;
	for(int i=x;i<x+3;i++)
		for(int j=y;j<y+3;j++)
			if(num[i][j]==key) return false;
	return true;
}

void DFS(int n){
	if(n==81){
		for (int i = 0; i < 9; i++){
			for (int j = 0; j < 8; j++)
				cout << num[i][j] << " ";
			cout << num[i][8];
			cout << endl;
```

```
            }
            sign=true;
            return ;
        }
        if(n==56&&num[6][0]==2&&num[6][1]==1)
            num[6][2]=5;
        if(num[n/9][n%9]!=0) DFS(n+1);
        else{
            for(int i=1;i<=9;i++){
                if(Check(n,i)){
                    num[n/9][n%9]=i;
                    DFS(n+1);
                    if(sign) return ;
                    num[n/9][n%9]=0;
                }
            }
        }
        return ;
}
int main(){
        for(int i=0;i<9;i++)
            for(int j=0;j<9;j++)
                cin>>num[i][j];
        DFS(0);
        return 0;

}
```

104. 将真分数分解为埃及分数

题目描述

分子为 1 的分数称为埃及分数。现输入一个真分数(分子比分母小的分数，叫做真分数)，请

将该分数分解为埃及分数。如：8/11 = 1/2+1/5+1/55+1/110。

注：真分数指分子小于分母的分数，分子和分母有可能 gcd 不为 1！

如有多个解，请输出任意一个。

请注意本题含有多组样例输入！

输入描述：
输入一个真分数，String 型
输出描述：
输出分解后的 string

示例 1

输入

复制

```
8/11


2/4
```

输出

复制

```
1/2+1/5+1/55+1/110


1/3+1/6
```

说明

> 第二个样例直接输出 1/2 也是可以的

```
/**
数学家斐波那契提出的一种求解***分数的贪心算法，准确的算法表述应该是这样的：
设某个真分数的分子为 a，分母为 b;
把 c=(b/a+1)作为分解式中第一个***分数的分母；
将 a-b%a 作为新的 a；
将 b*c 作为新的 b；
如果 a 等于 1，则最后一个***分数为 1/b,算法结束；
如果 a 大于 1 但是 a 能整除 b，则最后一个***分数为 1/(b/a),算法结束；
否则重复上面的步骤。
**/
#include <iostream>
#include <string>

using namespace std;

int main()
{
	int a,b;
	char ch;
	while(cin>>a>>ch>>b)
	{
		while(a != 1)
		{
			if(b%(a-1) == 0)//能整除出来
			{
				cout<<1<<"/"<<b/(a-1)<<"+";
				a=1;
			}else
			{
				int c = b/a +1;
				cout<<1<<"/"<<c<<"+";//第一个
				a = a - b%a;
				b= b*c;
				if(b%a == 0)//能整除，出来
				{
					b = b/a;
					a= 1;
				}
```

```
                }
            }
            cout<<1<<"/"<<b<<endl;
        }

        return 0;
}
```

注意：本题说写出一种分解方法即可。所以 8/11 可以分解为 8 个 1/11 相加。

```
#include<iostream>
#include<string>
using namespace std;
int main() {
    string s;
    while (cin >> s) {
        string ans;
        int n;
        for (int i = 0; i < s.size(); i++) {
            if (s[i] == '/') {
                n = stoi(s.substr(0, i));
                s = "1/" + s.substr(i + 1) + "+";
                break;
            }
        }
        while (n--) {
            ans += s;
        }
        cout << ans.substr(0,ans.size()-1) << endl;
    }
}
```

105. 24 点运算

题目描述

计算 24 点是一种扑克牌益智游戏，随机抽出 4 张扑克牌，通过加(+)，减(-)，乘(*), 除(/)

四种运算法则计算得到整数 24，本问题中，扑克牌通过如下字符或者字符串表示，其中，

小写 joker 表示小王，大写 JOKER 表示大王：

3 4 5 6 7 8 9 10 J Q K A 2 joker JOKER

本程序要求实现：输入 4 张牌，输出一个算式，算式的结果为 24 点。

详细说明：

1.运算只考虑加减乘除运算，没有阶乘等特殊运算符号，没有括号，友情提醒，整数除法要当心，是属于整除，比如 2/3=0，3/2=1；

2.牌面 2~10 对应的权值为 2~10, J、Q、K、A 权值分别为为 11、12、13、1；

3.输入 4 张牌为字符串形式，以一个空格隔开，首尾无空格；如果输入的 4 张牌中包含大小王，则输出字符串“ERROR”，表示无法运算；

4.输出的算式格式为 4 张牌通过+-*/四个运算符相连，中间无空格，4 张牌出现顺序任意，只要结果正确；

5.输出算式的运算顺序从左至右，不包含括号，如 1+2+3*4 的结果为 24，2 A 9 A 不能变为(2+1)*(9-1)=24

6.如果存在多种算式都能计算得出 24，只需输出一种即可，如果无法得出 24，则输出“NONE”表示无解。

7.因为都是扑克牌，不存在单个牌为 0 的情况，且没有括号运算，除数(即分母)的数字不可能为 0

输入描述：

输入 4 张牌为字符串形式，以一个空格隔开，首尾无空格；

输出描述：

输出怎么运算得到 24，如果无法得出 24，则输出“NONE”表示无解，如果输入的 4 张牌中包含大小王，则输出字符串“ERROR”，表示无法运算；

示例 1

输入

复制

```
A A A A
```

输出

复制

```
NONE
```

说明

不能实现

示例 2

输入

复制

```
4 2 K A
```

输出

复制

```
K-A*4/2
```

说明

```
A+K*2-4 也是一种答案，输出任意一种即可
```

示例 3

输入

复制

```
B 5 joker 4
```

输出

复制

```
ERROR
```

说明

```
存在 joker，输出 ERROR
```

示例 4

输入

复制

```
K Q 6 K
```

输出

复制

```
NONE
```

说明

按一般的计算规则来看，K+K-(Q/6)=24，但是因为这个题目的运算不许有括号，所以只能为 K+K-Q/6=2 ，其他情况也不能运算出 24 点，故不存在，输出 NONE

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

char c[3];
bool is24(int index,int cn,int a[]){ //对应 1 a[0] a a 对应 int a[4]
	if(index==4){
		if(cn==24)//最后等于啥，一看就是用递归了
			return true;
		else
			return false;
	}
	bool flag=0;
	for(int i=0;i<4;++i){
		switch(i){
			case 0:if(is24(index+1,cn+a[index],a))//+
					{c[index-1]='+';flag=1;}
				break;
			case 1:if(is24(index+1,cn-a[index],a))//-
					{c[index-1]='-';flag=1;}
				break;
			case 2:if(is24(index+1,cn*a[index],a))// *
					{c[index-1]='*';flag=1;}
				break;
			case 3:if (cn % a[index] == 0){//整除，才放进去吗，可以不用这个判断
				 if(is24(index+1,cn/a[index],a))// 除法
					{c[index-1]='/';flag=1;}
			    }
				break;
		}
		if(flag==1)//表示可以
			break;
	}
	if(flag==1)
		return true;
	else
		return false;
}
```

```
int main(){
    string s[4];
    while(cin>>s[0]>>s[1]>>s[2]>>s[3]){
        int a[4];
        bool ff=0;
        for(int i=0;i<4;++i){
            if(s[i]=="joker"||s[i]=="JOKER"){
                cout<<"ERROR"<<endl;//大小王，直接退出
                ff=1;break;
            }
            if(s[i][0]>='2'&&s[i][0]<='9')
                a[i]=s[i][0]-'0';
            else if(s[i]=="10")
                a[i]=10;
            else if(s[i]=="J")
                a[i]=11;
            else if(s[i]=="Q")
                a[i]=12;
            else if(s[i]=="K")
                a[i]=13;
            else if(s[i]=="A")
                a[i]=1; //4 个数拿出来
        }
        if(ff)//直接下一个
            continue;
        sort(a,a+4);
        bool flag=0;
        do{
            if(is24(1,a[0],a)){
                flag=1;break;//如果存在了，就退出
            }
        }while(next_permutation(a,a+4));//每次四个排列
        if(flag){
            char s[4];
            for(int i=0;i<4;++i){
              switch(a[i])
              {
              case 1:s[i]='A';break;//每个 a[i]等于哪个，在出来
              case 11:s[i]='J';break;
              case 12:s[i]='Q';break;
              case 13:s[i]='K';break;
              case 2:s[i]='2';break;
              case 3:s[i]='3';break;
              case 4:s[i]='4';break;
```

```
                    case 5:s[i]='5';break;
                    case 6:s[i]='6';break;
                    case 7:s[i]='7';break;
                    case 8:s[i]='8';break;
                    case 9:s[i]='9';break;
                    }
                }
                cout<<s[0]<<c[0]<<s[1]<<c[1]<<s[2]<<c[2]<<s[3]<<endl;
            }
            else
                cout<<"NONE"<<endl;
        }
        return 0;
}
```