这道题比较好，正好可以很好的复习一些位运算的相关知识

**1.一个数组中有两个元素只出现一次，其他所有元素都出现两次，求这两个只出现一次的元素**

**[解题思路]：**将数组所有元素都进行异或得到一个不为0的结果，根据这个结果中的不为0的某一位将数组分成两组，将两组中的元素进行异或，如两个数组的异或值都不为0，则得到最后结果。

**2.一个数组中有一个元素只出现1次，其他所有元素都出现k次，求这个只出现1次的元素**

**[解题思路]：**

当k为偶数时，直接将这个数组逐个进行异或操作，即可。

当k为奇数时，将数组中每个元素的每一位相加mod k，得到结果即位出现1次的元素，时间复杂度O(nlen)，空间复杂度为O(1)

这里相加的是二进制的位，不是十进制的。比如K = 3，数据如下：

69（1000101）出现1次, 33(100001)出现3次, 147（10010011）出现3次。

那么运算按二进制逐位求和并模k。

（

01000101+

00100001+

00100001+

00100001+

10010011+

10010011+

10010011

）mod(3) =

01000101(63)。

01000101+

00100001+

00100001+

00100001+

10010011+

10010011+

10010011

等于31330137。每一位是1的个数。这已经不是二进制了。

最后的结果是31330137每一位mod(3)，得到二进制表示01000101 = 十进制63

上面是摘抄的别人的关于这道题的算法思想，，感觉很牛掰呀

### 一．逻辑运算符

#### 1.& 位与运算

##### 1) 运算规则

位与运算的实质是将参与运算的两个数据，按对应的二进制数逐位进行逻辑与运算。例如：int型常量4和7进行位与运算的运算过程如下：

4=0000 0000 0000 0100 &7 =0000 0000 0000 0111= 0000 0000 0000 0100

对于负数，按其补码进行运算。例如：例如：int型常量-4和7进行位与运算的运算过程如下： －4=1111 1111 1111 1100 &7 =0000 0000 0000 0111= 0000 0000 0000 0100

##### 2) 典型应用

###### (1) 清零

清零：快速对某一段数据单元的数据清零，即将其全部的二进制位为0。例如整型数a=321对其全部数据清零的操作为a=a&0x0。 321=0000 0001 0100 0001 &0=0000 0000 0000 0000

= 0000 0000 0000 0000

###### (2) 获取一个数据的指定位

获取一个数据的指定位。例如获得整型数a=的低八位数据的操作为a=a&0xFF。321=

0000 0001 0100 0001 & 0xFF =0000 0000 1111 11111

= 0000 0000 0100 0001

获得整型数a=的高八位数据的操作为a=a&0xFF00。==a&0XFF00==

321=0000 0001 0100 0001 & 0XFF00=1111 1111 0000 0000

= 0000 0001 0000 0000

###### (3)保留数据区的特定位

保留数据区的特定位。例如获得整型数a=的第7-8位（从0开始）位的数据操作为： 110000000

321=0000 0001 0100 0001 & 384=0000 0001 1000 0000

=0000 0001 0000 0000

#### 2. | 位或运算

##### 1) 运算规则

位或运算的实质是将参与运算的两个数据，按对应的二进制数逐位进行逻辑或运算。例如：int型常量5和7进行位或运算的表达式为5|7，结果如下：5= 0000 0000 0000 0101

| 7= 0000 0000 0000 0111=0000 0000 0000 0111

##### 2) 主要用途

(1) 设定一个数据的指定位。例如整型数a=321，将其低八位数据置为1的操作为a=a|0XFF。321= 0000 0001 0100 0001 | 0000 0000 1111 1111=0000 0000 1111 1111

逻辑运算符||与位或运算符|的区别

条件“或”运算符 (||) 执行 bool 操作数的逻辑“或”运算，但仅在必要时才计算第二个操作数。 x || y , x | y 不同的是，如果 x 为 true，则不计算 y（因为不论 y 为何值，“或”操作的结果都为 true）。这被称作为“短路”计算。

#### 3. ^ 位异或

##### 1) 运算规则

位异或运算的实质是将参与运算的两个数据，按对应的二进制数逐位进行逻辑异或运算。只有当对应位的二进制数互斥的时候，对应位的结果才为真。例如：int型常量5和7进行位异或运算的表达式为5^7，结果如下：5=0000 0000 0000 0101^7=0000 0000 0000 0111

= 0000 0000 0000 0010

##### 2) 典型应用

###### (1)定位翻转

定位翻转：设定一个数据的指定位，将1换为0，0换为1。例如整型数a=321,，将其低八位数据进行翻位的操作为a=a^0XFF;

###### (2)数值交换

数值交换。例如a=3,b=4。在例11-1中，无须引入第三个变量，利用位运算即可实现数据交换。以下的操作可以实现a,b两个数据的交换：

a=a^b;

b=b^a;

a=a^b;

#### 4．~ 位非

位非运算的实质是将参与运算的两个数据，按对应的二进制数逐位进行逻辑非运算。

### 二．位移运算符

#### 1．位左移

左移运算的实质是将对应的数据的二进制值逐位左移若干位，并在空出的位置上填0，最高位溢出并舍弃。例如int a,b;

a=5;

b=a<<2;

则b=20，分析过程如下：

(a)10=(5)10=(0000 0000 0000 0101)2

b=a<<2;

b=(0000 0000 0001 0100)2=(20)10

从上例可以看出位运算可以实现二倍乘运算。由于位移操作的运算速度比乘法的运算速度高很多。因此在处理数据的乘法运算的时，采用位移运算可以获得较快的速度。

提示 将所有对2的乘法运算转换为位移运算，可提高程序的运行效率

#### 2．位右移

位右移运算的实质是将对应的数据的二进制值逐位右移若干位，并舍弃出界的数字。如果当前的数为无符号数，高位补零。例如：

int (a)10=(5)10=(0000 0000 0000 0101)2

b=a>>2;

b=(0000 0000 0000 0001)2=(1)10

如果当前的数据为有符号数，在进行右移的时候，根据符号位决定左边补0还是补1。如果符号位为0，则左边补0；但是如果符号位为1，则根据不同的计算机系统，可能有不同的处理方式。可以看出位右移运算，可以实现对除数为2的整除运算。

提示 将所有对2的整除运算转换为位移运算，可提高程序的运行效率

3．复合的位运算符

在C语言中还提供复合的位运算符，如下：

&=、!=、>>=、<<=和^=

例如：a&=0x11等价于 a= a&0x11，其他运算符以此类推。

不同类型的整数数据在进行混合类型的位运算时，按右端对齐原则进行处理，按数据长度大的数据进行处理，将数据长度小的数据左端补0或1。例如char a与int b进行位运算的时候，按int 进行处理，char a转化为整型数据，并在左端补0。

补位原则如下：

1) 对于有符号数据：如果a为正整数，则左端补0，如果a 为负数，则左端补1。

2) 对于无符号数据：在左端补0。