A 水水的a+b>c

题目描述

输入三个数 a, b, c，判断 $a + b$ 是否大于 c 。

输入格式

多组数据输入；

第一行，一个正整数 n，表示数据组数；

接下来 n 行，每行 3 个以空格分隔的整数 a, b, c ；

请注意 a, b, c 的数据范围。

输出格式

对于每组数据，输出一行字符串；

若 $a + b > c$ ，输出 `true` ，否则输出 `false`

输入样例

```
2
1 2 3
-5 19 4
```

输出样例

```
false
true
```

数据范围

$0 < n < 1000$

$-2^{31} \leq a, b, c < 2^{31}$

HINT

int 可表示数的范围即是 $[-2^{31}, 2^{31})$ ，但两个 int 类型的数相加**可能**会溢出 int ，因此我们可以用 $long\ long$ 来读入数据，$long\ long$ 的范围是 $[-2^{63}, 2^{63})$ ；

```
long long big;          // 定义一个long long型变量
scanf("%lld", &big);    // 读入一个long long型变量
printf("%lld\n", big);  // 输出一个long long型变量
```

溢出 int 的含义是超过了 int 的表示范围，运行以下代码，两个正数相加却输出了一个负数，这就是溢出的含义。

```
#include <stdio.h>

int main() {
    int a = 2147483647;     // a = 2^31 - 1
    int b = 1;
    printf("%d\n", a + b);
    return 0;
}
```

之后做题一定要留意数据范围，不然很容易掉坑了！

仅用 int 读入数据似乎也能解决这道题，做完可以稍加思考。

Author: wqh

B 高斯求和

题目背景

（可直接看题目描述）

高斯求和的小故事相信大家都耳熟能详。1787年高斯10岁，他进入了学习数学的班次，这是一个首次创办的班，孩子们在这之前都没有听说过算术这么一门课程。数学教师是布特纳（Buttner），他对高斯的成长也起了一定作用。在全世界广为流传的一则故事说，高斯10岁时算出布特纳给学生们出的将1到100的所有整数加起来的算术题，布特纳刚叙述完题目，高斯就算出了正确答案。
不过，这很可能是一个不真实的传说。据对高斯素有研究的著名数学史家E·T·贝尔（E.T.Bell）考证，布特纳当时给孩子们出的是一道更难的加法题：$81297+81495+81693+\ldots+100899$。这也是一个等差数列的求和问题（公差为$198$，项数为$100$）。当布特纳刚一写完时，高斯也算完并把写有答案的小石板交了上去。E·T·贝尔写道，高斯晚年经常喜欢向人们谈论这件事，说当时只有他写的答案是正确的，而其他的孩子们都错了。高斯没有明确地讲过，他是用什么方法那么快就解决了这个问题。数学史家们倾向于认为，高斯当时已掌握了等差数列求和的方法。一位年仅10岁的孩子，能独立发现这一数学方法实属很不平常。现在，请你借助计算机的帮助，模仿高斯计算出等差数列的和吧。

题目描述

给出一个等差数列的首项 a_1 、公差 k 与项数 n ，请计算该等差数列这 n 项的和。

输入

多组数据输入；

第一行输入一个正整数 num ，表示一共有 num 组数据。

接下来 num 行，每行共有三个正整数，分别表示一个等差数列的首项 a_1 、公差 k 与项数 n 。

输出

共输出 num 行，每行一个整数；

第 i 行表示相应第 i 组数据对应的等差数列的 n 项的和。

输入样例1

```
1
124 24 35
```

输出样例1

```
18620
```

输入样例2

```
5
220 36 2
128 31 30
98 70 17
62 59 18
206 16 24
```

输出样例2

```
476
17325
11186
10143
9360
```

样例解释

对于**输入样例1**，第一行数字1代表接下来将输入1组数据。在接下来的1行代表着该等差数列项从124开始，以大小为24的公差增加35次，则等差数列和为：
$124+148+172+\ldots\ldots+940=18620$，故这一组数据输出$18620$。

数据范围

对输入数据，a_1,k 与等差数列项和 sum 均为正整数且在 int 范围内；

数据组数 num 满足 $num<200$，项数 n 满足 $n<100$。

AUTHOR：czy

C 一元二次方程进阶版

题目描述

小明同学有很多一元二次方程，你能帮他求解这些方程么?

方程形式为$ax^2+bx+c=0$，其中x为未知数，a、b、c为参数。

与之前不同的是，小明同学新学习了复数的概念，之前无解的方程都可以由复数表示根。

输入

输入共 $n+1$ 行;

第一行一个整数 n 表示有一共有n个待求解的方程。

接下来 n 行，每行三个整数，分别表示第 i个方程的参数 a_i、b_i、c_i

输出

输出共 n 行

第 i 行表示输入的第i个方程的求解结果:

- 如果此方程不是一元二次方程，输出这个一元一次方程的解**（输入保证此方程一定有解）**，结果保留到小数点后6位。
- 如果方程有实数解，输出一行两个浮点数，以一个空格分隔，表示方程的两个根，结果保留到小数点后6位；**值大的在前，值小的在后**，如果是重根，输出两个一样的数。
- 如果方程没有实数解，输出两个复数解 $p+qi$ 和 $p-qi$，以一个空格分开，其中$q>0$。p 和 q 保留到小数点后6位**（如果p是0也要输出）**。

输入样例

```
4
1 3 1
2 -4 2
1 2 8
0 3 4
```

输出样例

```
-0.381966 -2.618034
1.000000 1.000000
-1.000000+2.645751i -1.000000-2.645751i
-1.333333
```

数据范围

$-100 \leq a_i, b_i, c_i \leq 100$

$a_i, b_i, c_i \in Z$

$1 \leq n \leq 10$

HINT

参考第一次上机C题及课件P3 p15复数表示

1. $\Delta = b^2 - 4ac$
2. $\sqrt{-1} = i$
3. 如果结果有0需要输出为`0.000000`
4. 开根号可以利用`math.h`中的`sqrt()`函数

   C库函数-sqrt()
5. 求绝对值可以利用`math.h`中的`fabs()`函数，也可以自己手写

   C库函数-fabs()
6. 输出保留两位小数的示例代码为

```
double a = 2.3333333;
printf("%.2f",a);
```

AUTHOR:zym

D LJF的位运算加密

题目背景

LJF其实是一个害羞的小男生，而且还是一个科技文盲，平时只能靠传纸条的方式跟npy联系。众所周知，我航的第二校训为`上网不涉密，涉密不上网`，所以LJF和npy约定了每次都在纸条加上不同的加密信息。但是LJF的npy并不会位运算，所以需要大家来帮忙解密。

题目描述

纸条上原有的信息为1个 int 型32位正整数 a，还有3个加密码 x, y, z，只需要将 a 的二进制位的第 x, y, z 位全部置 0 便可解密（最低位认为是第0位），请你输出解密后的结果。

输入

共2行数据。

第1行为一个十进制正整数a。

第2行为3个加密码 x, y, z，用空格隔开。

输出

一个十进制整数。即解密后的数字。

输入样例

```
744
5 6 7
```

输出样例

```
520
```

样例解释

`744`的32位二进制为`0000 ... 0010 1110 1000`，将其第$5, 6, 7$位上的`1`置`0`（最右位为第 0 位），得到的二进制数为`0000 ... 0010 0000 1000`，转换为十进制即`520`。

数据范围

$$0 < a < 2147483648$$

$$0 \le x, y, z \le 31$$

x, y, z互不相等

HINT

本题是位运算操作中的关闭位（清空位）的用法，也就是说，在不影响其他位的情况下将指定位“置零”。假设要将变量flag的第1号位“置零”，我们可以利用一个变量MASK，MASK只有第1位是1，其他位都是0，然后进行如下操作

```
flag = flag & (~MASK)
```

由于MASK只有第1位是1，其他位都是0，所以~MASK只有第1位是0，其他位是1；然后与flag进行按位与 `&` 运算，得到的结果就是将flag的第1号位“置零”，其他位不变。

那么如何得到MASK呢？我们可以利用左移 `<<` 运算

```
MASK = 1 << n; //第n位是1，其他位是0的MASK
```

接下来问题就是如何获得一个3个特定位上是1，其他位上是0的数；我们可以按照上面的方法分别得到三个所需位上为1，其他位上为0的数MASK1，MASK2，MASK3，然后将这三个数进行按位或 `|` 运算，即

```
MASK = MASK1 | MASK2 | MASK3;
```

按照如上方法一步一步去写吧！（再提示就差把代码给你了）

Author:LJF

E 统计字母

题目描述

统计一个字符串里出现各个字母的次数，不考虑大小写

输入

一行，一个字符串，长度小于256。

输出

分多行输出,按A到Z的顺序输出相应字母在在输入的字符串中出现次数,**不考虑大小写**，如果出现次数为0则不要输出.

格式为: `字母大写[空格]出现的次数`

输入样例

```
#define cbj0 GARBAGE
```

输出样例

```
A 2
B 2
C 1
D 1
E 3
F 1
G 2
I 1
J 1
N 1
R 1
```

HINT

参考课件P3 例3-10

开一个数组存储A到Z的出现次数

author:cbj

F 小冰的烦恼

题目描述

二进制是以2为基数的记数系统，在计算机科学、电子技术等领域中具有极其重要的意义，位操作则是程序设计中对二进制数的一元和二元操作。

小冰学习了最基础的位运算知识，希望能借此实现交换某个整数二进制的任意两位，即给定正整数 x ，交换其二进制的第 m 位与第 n 位，你能帮帮他吗?

输入

一行输入，三个十进制整数 x, m, n

输出

一行输出，一个十进制正整数，表示正整数 x 交换两位后的结果

输入样例

```
100 2 4
```

输出样例

```
112
```

样例解释

100的二进制为1100100，第二位为1，第四位为0，交换两位后二进制为1110000，十进制表示为112

数据范围

$0 < x < 1000000$

$0 \leq m, n < 20$

Hint

可结合按位与、按位或和左移等运算实现取出二进制第n位，将第n位置1，第n位置0等操作。

```
x&(1<<n);//取出第n位
x|=(1<<n);//第n位置1
x&=~(1<<n);//第n位置0
```

Author:李南冰

G 进制转换plus

题目描述

给出十进制整数 a 和 k ，请把 a 转换为 k 进制后输出。当 $k > 10$ 的时候，$10 - 35$ 的数字用大写字母 `A`-`Z` 表示。

输入

两个整数 a 和 k，中间用空格隔开，含义如题目描述。

输出

一个数，表示 a 的 k 进制。

输入样例

```
47 16
```

输出样例

```
2F
```

样例解释

十进制整数47的16进制表示为2F。

数据范围

a 在 int 范围内且 $a \geq 0$，$2 \leq k \leq 36$。

HINT

参考课件P3 例3-1
如果不想打26个`if`的话，不妨利用一下ASCII码的知识吧。

```
int word=10;
printf("%c",word+55);//输出为字母A哦
printf("%c",word-10+'A');//这样也可以哦
```

AUTHOR：wwh

H 原码，反码，补码

题目描述

小林今天复习了原码，反码，补码的计算方法，于是他心血来潮决定出一道题考考你：

给定一个`signed char`类型的数，请依次输出它的原码，反码和补码。

输入

一个`signed char`类型的数x。

输出

共三行，第一行为x的原码；第二行为x的反码；第三行为x的补码；每行均由八个0或1组成。

输入样例1

```
7
```

输出样例1

```
00000111
00000111
00000111
```

输入样例2

```
-7
```

输出样例2

```
10000111
11111000
11111001
```

样例解释

7转化为8位二进制数得00000111，即原码，因为7大于0，所以反码和补码与原码相同；

-7转化为8位二进制数得10000111，最高位为符号位，1表示为负，因为-7小于0，所以反码为其绝对值的原码取反，即将00000111取反得到11111000，补码为反码+1，即11111001。

数据范围

数据保证x在`signed char`范围内且不会等于`-128`(因为`-128`没有原码和反码)，即$-127 \leq x \leq 127$;

为简化题目，保证数据中不会出现`-0`。

HINT

参考课件P3 p16原码反码补码

`signed char`可以简单的理解为长度为八位的带符号整型，可以直接用int类型来输入：

```
int x;
scanf("%d",&x);
```

当然，你也可以通过以下方式输入：

```
signed char x;
scanf("%hhd",&x);
```

如果忘记了原码，反码，补码的计算方式，可以复习一下课件P3-c2中的内容。

Author:ljh

送蚂蚁回家

脑筋急转弯

题目描述

有 n 只蚂蚁爬上了一个长度为 m 的很细很细的木棍（坐标 0 到 m ）。突然它们意识到了危险决定离开这条木棍，但是又没有人指挥，于是他们随便选了一个方向开始爬行。

已知蚂蚁的爬行速度是 1 ，爬行方向只有 $+1$ 和 -1 两种；由于木棍很细，如果两只蚂蚁相遇了，它们就会碰头然后各自掉头返回。当蚂蚁走到了 0 或 m 的位置，它就会掉下木棍，当爬行方向是 $+1$ 时，每个单位时间蚂蚁的坐标将会 $+1$ (如果没相遇的话)。

请问所有蚂蚁都掉下木棍需要多长时间?

输入格式

输入共 $n+1$ 行，第一行两个整数 n, m ，含义如题。

以下每一行，两个整数 x_i 和 a_i ，表示一只蚂蚁的状态，其中 x_i 表示这只蚂蚁的坐标， a_i 表示这只蚂蚁初次选择的行进方向，保证 $a_i = 1$ 或 $a_i = -1$ 。

输出格式

输出一个整数，表示所有蚂蚁掉下木棍所需要的时间。

输入样例

```
2 3
1 1
2 -1
```

输出样例

```
2
```

样例解释

对于样例，两只蚂蚁将会在 0.5 单位时间相遇，再花 0.5 单位时间回到自己最初的位置，但此时位于 1 的蚂蚁方向是 -1 ，位于 2 的蚂蚁方向是 $+1$ ，所以在下一时刻，它们分别到达了 0 和 3 ，掉下了木棍。

数据范围

$2 \leq n, m \leq 100000$ ，保证没有蚂蚁初始位置相同。

题面灵感来源于ksj

J 双重循环移位

题目描述

给予两个`int`型数据`a`和`b`，在C语言原有的左移以及右移运算的基础上，定义`a`和`b`的一次**向左（右）的**“双重循环移位”操作为：

- 第一步：
  - 对于**向左的**“双重循环移位”：将`a`**左移**一位，溢出的位记为`a0`；将`b`**左移**一位，溢出的位记为`b0`；
  - 对于**向右的**“双重循环移位”：将`a`**右移**一位，溢出的位记为`a0`；将`b`**右移**一位，溢出的位记为`b0`；
- 第二步：
  - 对于**向左的**“双重循环移位”：将`a`的**最低位**设置为`b0`；将`b`的**最低位**设置为`a0`；
  - 对于**向右的**“双重循环移位”：将`a`的**最高位**设置为`b0`；将`b`的**最高位**设置为`a0`。

（HINT中给出了具体的示例）

现在给予你两个`int`型数据`a`和`b`，请你输出经过n次向左或向右的“双重循环移位”后的`a`和`b`。

输入

多组数据输入。

每行数据有：两个整数，代表`int`型数据`a`和`b`；一个字母，只可能是`l`或`r`，`l`代表向左进行操作，`r`代表向右进行操作；以及一个正整数 n，代表执行操作的次数。每个数据之间用一个空格隔开。

其中`a`和`b`的数据大小不会超过`int`类型的数据范围，$0 \leq n < 2^{32}$.

数据组数不超过10^4组。

输出

对于每组数据，输出一行：经过n次“双重循环移位”操作的`int`型数据`a`和`b`，用一个空格隔开。

输入样例

```
0 1 l 1
0 0 r 0
1 2 r 64
2 1 l 32
9903 13390 l 29982
7971 -93323 r 90287
-56676 -25319 r 82517434
-27104 68213 l 363813242
```

输出样例

```
0 2
0 0
1 2
1 2
-1073738477 -2147481173
1044905981 652869632
-3627201 -1620353
-671089064 -2147482583
```

HINT

以8位二进制数举例说明一次向左的“双重循环移位“：

```
|    a     |    b     |   | a0 |    a     | b0 |    b     |   |    a     |    b     |
|----------|----------|-->|----|----------|----|----------|-->|----------|----------|
| 10000000 | 11111110 |   | 1  | 00000000 | 1  | 11111100 |   | 00000001 | 11111101 |

.------------>------------.
|      a          b       |  ===>     a          b
'<--10000000<--11111110<--'         00000001   11111101
```

以8位二进制数举例说明一次向右的“双重循环移位“：（右移操作采用逻辑右移）

```
|    a     |    b     |   | a0 |    a     | b0 |    b     |   |    a     |    b     |
|----------|----------|-->|----|----------|----|----------|-->|----------|----------|
| 00000001 | 01111111 |   | 1  | 00000000 | 1  | 00111111 |   | 10000000 | 10111111 |

.------------<------------.
|      a          b       |  ===>     a          b
'-->00000001-->01111111-->'         10000000   10111111
```

Author: Lucien Li