比较安全的两整数平均值算法实现

求两个整数的平均值这个问题相信大家都想过，大家肯定会很快的写出以下的算法：

```
public static int mean(int a, int b){
   return (a + b) / 2;
}
或者
public static int mean(int a, int b){
   return (a + b) >> 1;
}
或者
public static int mean(int a, int b){
   return (a + b) >>> 1;
}
```

不错，上面的函数是能够求出a和b的平均值，但是你当a=0xffffffff,b=0x00000001,上面函数求出的值都为0。等等，不对啊，两个正整数相加的平均值之和怎么可能为0呢？仔细分析你会发现a=0xffffffff,b=0x00000001相加之后会导致算出的结果溢出，使得算出来的结果出现了错误。那么怎么才能使得两
数的平均值不会出现上述的情况呢？关于>>和>>>的区别见《Java中>>和>>>移位操作符的区别》

我们都知道，每个十进制数可以转换为对应的二进制，并且可以分解为各个位与其权的乘积的和。比如：10 和 6 的平均值：10的二进制是：1010，6的二进制是：0110.

10 = 1010（二进制）= $1* 2^3+ 0 * 2^2 + 1 *2^1 + 0* 2^0$

6 = 0110 （二进制）= $0 * 2^3+ 1 * 2^2 + 1 *2^1 + 0* 2^0$

然后，把两个数的分解为这样的多项式进行相加。考虑两个数的二进制序列中相同的位与不同的位：

1、将相同的位进行相加，结果等于两数按位与的结果的两倍；

2、将不同的位进行相加，其结果等于按位异或的结果。

比如：10和6相同的部分为0010，不同的部分分别为1000,0100，所以：

```
(10 + 6) / 2 <=> ((0010+ 1000) + (0010+ 0100)) >> 1
<=> ((0010+ 0010) + (1000+ 0100)) >> 1
<=> (0010+ 0010) >> 1 + (1000+ 0100) >> 1
<=> 0100 >> 1 + 1100 >> 1
<=> 0010 + 1100 >> 1
<=> 0010 + 0110
<=> 1000
```

分析我们可以发现，上述计算步骤的 0010 + 1100 >> 1 步是由(10 & 6) + ((10 ^ 6) >> 1)得到的，所以，任何的整数的平均值都可以由(a & b) + ((a ^ b) >> 1)计算出，算法如下：

```
public static int mean(int a, int b){
   return (x & y) + ((x ^ y) >> 1);
}
```

（完）