2.61

```
A
   //x 所有位都为 1 等价于 x + 1 = 0
   !(x + 1)
B
   //x 所有位都为 0 等价于 x = 0
   !x
C
   //取出 x 最低字节,若都为 1 则是 0xFF,
   //减去 0xFF 再取反就符合要求了
   !((x & 0xFF) - 0xFF)
D
   //右移将最高字节移动到最低字节,再取出来
   !((x >> ((sizeof(int) - 1) << 3)) & 0xFF)
```

2.62

```
bool int_shifts_are_arithmetic(){
   //二进制全为 1 的数字
   int testint = -1;
   //这台机器上 int 的字长
   int w = sizeof(int) << 3;
   //右移 1 位,看符号位为 0 还是 1
   return (((testint >> 1) & (1 << (w - 1))) >> (w - 1));
}
```

2.65

```
int odd_ones(unsigned x){
   //异或会成对消去 1,不改变 1 的个数的奇偶性
   //用类似二分的方法,将前一半折到后面计算
   //之后不用考虑前面的位,缩小 x 有效的位数
   x ^= x >> 16;
   x ^= x >> 8;
   x ^= x >> 4;
   x ^= x >> 2;
   x ^= x >> 1;
   //只需看最后一位为 0 还是 1
   return x & 1;
}
```