```
/*
 * Do rotating left shift.  Assume 0 <= n < w
 * Examples when x = 0x12345678 and w = 32:
 *    n=4 -> 0x23456781, n=20 -> 0x67812345
 */
unsigned rotate_left(unsigned x, int n);
```

函数应该遵循位级整数编码规则。要注意 n=0 的情况。

**2.70　写出具有如下原型的函数的代码：

```
/*
 * Return 1 when x can be represented as an n-bit, 2's-complement
 * number; 0 otherwise
 * Assume 1 <= n <= w
 */
int fits_bits(int x, int n);
```

函数应该遵循位级整数编码规则。

*2.71　你刚刚开始在一家公司工作，他们要实现一组过程来操作一个数据结构，要将 4 个有符号字节封装成一个 32 位 unsigned。一个字中的字节从 0(最低有效字节)编号到 3(最高有效字节)。分配给你的任务是：为一个使用补码运算和算术右移的机器编写一个具有如下原型的函数：

```
/* Declaration of data type where 4 bytes are packed
   into an unsigned */
typedef unsigned packed_t;

/* Extract byte from word.  Return as signed integer */
int xbyte(packed_t word, int bytenum);
```

也就是说，函数会抽取出指定的字节，再把它符号扩展为一个 32 位 int。

你的前任(因为水平不够高而被解雇了)编写了下面的代码：

```
/* Failed attempt at xbyte */
int xbyte(packed_t word, int bytenum)
{
    return (word >> (bytenum << 3)) & 0xFF;
}
```

A. 这段代码错在哪里？

B. 给出函数的正确实现，只能使用左右移位和一个减法。

**2.72　给你一个任务，写一个函数，将整数 val 复制到缓冲区 buf 中，但是只有当缓冲区中有足够可用的空间时，才执行复制。

你写的代码如下：

```
/* Copy integer into buffer if space is available */
/* WARNING: The following code is buggy */
void copy_int(int val, void *buf, int maxbytes) {
    if (maxbytes-sizeof(val) >= 0)
            memcpy(buf, (void *) &val, sizeof(val));
}
```

这段代码使用了库函数 memcpy。虽然在这里用这个函数有点刻意，因为我们只是想复制一个 int，但是它说明了一种复制较大数据结构的常见方法。

你仔细地测试了这段代码后发现，哪怕 maxbytes 很小的时候，它也能把值复制到缓冲区中。

A. 解释为什么代码中的条件测试总是成功。**提示**：sizeof 运算符返回类型为 size_t 的值。

B. 你该如何重写这个条件测试，使之工作正确。

**2.73　写出具有如下原型的函数的代码：

```
/* Addition that saturates to TMin or TMax */
int saturating_add(int x, int y);
```