* 2.64　写出代码实现如下函数：

```
/* Return 1 when any odd bit of x equals 1; 0 otherwise.
   Assume w=32 */
int any_odd_one(unsigned x);
```

函数应该遵循位级整数编码规则，不过你可以假设数据类型 int 有 $w=32$ 位。

** 2.65　写出代码实现如下函数：

```
/* Return 1 when x contains an odd number of 1s; 0 otherwise.
   Assume w=32 */
int odd_ones(unsigned x);
```

函数应该遵循位级整数编码规则，不过你可以假设数据类型 int 有 $w=32$ 位。

你的代码最多只能包含 12 个算术运算、位运算和逻辑运算。

** 2.66　写出代码实现如下函数：

```
/*
 * Generate mask indicating leftmost 1 in x.  Assume w=32.
 * For example, 0xFF00 -> 0x8000, and 0x6600 --> 0x4000.
 * If x = 0, then return 0.
 */
int leftmost_one(unsigned x);
```

函数应该遵循位级整数编码规则，不过你可以假设数据类型 int 有 $w=32$ 位。

你的代码最多只能包含 15 个算术运算、位运算和逻辑运算。

提示：先将 x 转换成形如[0…011…1]的位向量。

** 2.67　给你一个任务，编写一个过程 int_size_is_32()，当在一个 int 是 32 位的机器上运行时，该程序产生 1，而其他情况则产生 0。不允许使用 sizeof 运算符。下面是开始时的尝试：

```
1   /* The following code does not run properly on some machines */
2   int bad_int_size_is_32() {
3       /* Set most significant bit (msb) of 32-bit machine */
4       int set_msb = 1 << 31;
5       /* Shift past msb of 32-bit word */
6       int beyond_msb = 1 << 32;
7
8       /* set_msb is nonzero when word size >= 32
9          beyond_msb is zero when word size <= 32  */
10      return set_msb && !beyond_msb;
11  }
```

当在 SUN SPARC 这样的 32 位机器上编译并运行时，这个过程返回的却是 0。下面的编译器信息给了我们一个问题的指示：

```
warning: left shift count >= width of type
```

A. 我们的代码在哪个方面没有遵守 C 语言标准？

B. 修改代码，使得它在 int 至少为 32 位的任何机器上都能正确地运行。

C. 修改代码，使得它在 int 至少为 16 位的任何机器上都能正确地运行。

** 2.68　写出具有如下原型的函数的代码：

```
/*
 * Mask with least signficant n bits set to 1
 * Examples: n = 6 --> 0x3F, n = 17 --> 0x1FFFF
 * Assume 1 <= n <= w
 */
int lower_one_mask(int n);
```

函数应该遵循位级整数编码规则。要注意 $n=w$ 的情况。

** 2.69　写出具有如下原型的函数的代码：