- 禁止使用
  - 条件语句(if 或者?:)、循环、分支语句、函数调用和宏调用。
  - 除法、模运算和乘法。
  - 相对比较运算(<、>、<=和>=)。
- 允许的运算
  - 所有的位级和逻辑运算。
  - 左移和右移，但是位移量只能在 0 和 $w-1$ 之间。
  - 加法和减法。
  - 相等(==)和不相等(!=)测试。(在有些题目中，也不允许这些运算。)
  - 整型常数 INT_MIN 和 INT_MAX。
  - 对 int 和 unsigned 进行强制类型转换，无论是显式的还是隐式的。

即使有这些条件的限制，你仍然可以选择带有描述性的变量名，并且使用注释来描述你的解决方案的逻辑，尽量提高代码的可读性。例如，下面这段代码从整数参数 x 中抽取出最高有效字节：

```
/* Get most significant byte from x */
int get_msb(int x) {
    /* Shift by w-8 */
    int shift_val = (sizeof(int)-1)<<3;
    /* Arithmetic shift */
    int xright = x >> shift_val;
    /* Zero all but LSB */
    return xright & 0xFF;
}
```

**2.61　写一个 C 表达式，在下列描述的条件下产生 1，而在其他情况下得到 0。假设 x 是 int 类型。

A. x 的任何位都等于 1。

B. x 的任何位都等于 0。

C. x 的最低有效字节中的位都等于 1。

D. x 的最高有效字节中的位都等于 0。

代码应该遵循位级整数编码规则，另外还有一个限制，你不能使用相等(==)和不相等(!=)测试。

**2.62　编写一个函数 int_shifts_are_arithmetic()，在对 int 类型的数使用算术右移的机器上运行时这个函数生成 1，而其他情况下生成 0。你的代码应该可以运行在任何字长的机器上。在几种机器上测试你的代码。

**2.63　将下面的 C 函数代码补充完整。函数 srl 用算术右移(由值 xsra 给出)来完成逻辑右移，后面的其他操作不包括右移或者除法。函数 sra 用逻辑右移(由值 xsrl 给出)来完成算术右移，后面的其他操作不包括右移或者除法。可以通过计算 8*sizeof(int)来确定数据类型 int 中的位数 w。位移量 k 的取值范围为 $0 \sim w-1$。

```
unsigned srl(unsigned x, int k) {
    /* Perform shift arithmetically */
    unsigned xsra = (int) x >> k;
.
.
.
.
.
}

int sra(int x, int k) {
    /* Perform shift logically */
    int xsrl = (unsigned) x >> k;
.
.
.
.
.
}
```