同正常的补码加法溢出的方式不同，当正溢出时，饱和加法返回 $TMax$，负溢出时，返回 $TMin$。饱和运算常常用在执行数字信号处理的程序中。

你的函数应该遵循位级整数编码规则。

** 2.74　写出具有如下原型的函数的代码：

```
/* Determine whether arguments can be subtracted without overflow */
int tsub_ok(int x, int y);
```

如果计算 x-y 不溢出，这个函数就返回 1。

**★ 2.75　假设我们想要计算 $x \cdot y$ 的完整的 $2w$ 位表示，其中，$x$ 和 $y$ 都是无符号数，并且运行在数据类型 unsigned 是 $w$ 位的机器上。乘积的低 $w$ 位能够用表达式 x*y 计算，所以，我们只需要一个具有下列原型的函数：

```
unsigned unsigned_high_prod(unsigned x, unsigned y);
```

这个函数计算无符号变量 $x \cdot y$ 的高 $w$ 位。

我们使用一个具有下面原型的库函数：

```
int signed_high_prod(int x, int y);
```

它计算在 $x$ 和 $y$ 采用补码形式的情况下，$x \cdot y$ 的高 $w$ 位。编写代码调用这个过程，以实现用无符号数为参数的函数。验证你的解答的正确性。

**提示**：看看等式(2.18)的推导中，有符号乘积 $x \cdot y$ 和无符号乘积 $x' \cdot y'$ 之间的关系。

* 2.76　库函数 calloc 有如下声明：

```
void *calloc(size_t nmemb, size_t size);
```

根据库文档："函数 calloc 为一个数组分配内存，该数组有 nmemb 个元素，每个元素为 size 字节。内存设置为 0。如果 nmemb 或 size 为 0，则 calloc 返回 NULL。"

编写 calloc 的实现，通过调用 malloc 执行分配，调用 memset 将内存设置为 0。你的代码应该没有任何由算术溢出引起的漏洞，且无论数据类型 size_t 用多少位表示，代码都应该正常工作。

作为参考，函数 malloc 和 memset 声明如下：

```
void *malloc(size_t size);
void *memset(void *s, int c, size_t n);
```

** 2.77　假设我们有一个任务：生成一段代码，将整数变量 x 乘以不同的常数因子 $K$。为了提高效率，我们想只使用＋、－和≪运算。对于下列 $K$ 的值，写出执行乘法运算的 C 表达式，每个表达式中最多使用 3 个运算。

A. $K=17$

B. $K=-7$

C. $K=60$

D. $K=-112$

** 2.78　写出具有如下原型的函数的代码：

```
/* Divide by power of 2. Assume 0 <= k < w-1 */
int divide_power2(int x, int k);
```

该函数要用正确的舍入方式计算 $x/2^k$，并且应该遵循位级整数编码规则。

** 2.79　写出函数 mul3div4 的代码，对于整数参数 x，计算 3*x/4，但是要遵循位级整数编码规则。你的代码计算 3*x 也会产生溢出。

**★ 2.80　写出函数 threefourths 的代码，对于整数参数 x，计算 3/4x 的值，向零舍入。它不会溢出。函数应该遵循位级整数编码规则。

** 2.81　编写 C 表达式产生如下位模式，其中 $a^k$ 表示符号 $a$ 重复 $k$ 次。假设一个 $w$ 位的数据类型。代码可以包含对参数 j 和 k 的引用，它们分别表示 $j$ 和 $k$ 的值，但是不能使用表示 $w$ 的参数。