多程序员调用分配函数(如 malloc)时，使用算术表达式作为参数，并且不对这些表达式进行溢出检查。编写 calloc 的可靠版本留作一道练习题(家庭作业 2.76)。

练习题 2.37　现在你有一个任务，当数据类型 int 和 size_t 都是 32 位的，修补上述旁注给出的 XDR 代码中的漏洞。你决定将待分配字节数设置为数据类型 uint64_t，来消除乘法溢出的可能性。你把原来对 malloc 函数的调用(第 9 行)替换如下：

```
uint64_t asize =
    ele_cnt * (uint64_t) ele_size;
void *result = malloc(asize);
```

提醒一下，malloc 的参数类型是 size_t。

A. 这段代码对原始的代码有了哪些改进？

B. 你该如何修改代码来消除这个漏洞？

2.3.6　乘以常数

以往，在大多数机器上，整数乘法指令相当慢，需要 10 个或者更多的时钟周期，然而其他整数运算(例如加法、减法、位级运算和移位)只需要 1 个时钟周期。即使在我们的参考机器 Intel Core i7 Haswell 上，其整数乘法也需要 3 个时钟周期。因此，编译器使用了一项重要的优化，试着用移位和加法运算的组合来代替乘以常数因子的乘法。首先，我们会考虑乘以 2 的幂的情况，然后再概括成乘以任意常数。

原理：乘以 2 的幂

设 x 为位模式$[x_{w-1}, x_{w-2}, \cdots, x_0]$表示的无符号整数。那么，对于任何 $k \geqslant 0$，我们都认为$[x_{w-1}, x_{w-2}, \cdots, x_0, 0, \cdots, 0]$给出了 $x2^k$ 的 $w+k$ 位的无符号表示，这里右边增加了 k 个 0。

因此，比如，当 $w=4$ 时，11 可以被表示为[1011]。$k=2$ 时将其左移得到 6 位向量[101100]，即可编码为无符号数 $11 \cdot 4 = 44$。

推导：乘以 2 的幂

这个属性可以通过等式(2.1)推导出来：

$$\begin{aligned} B2U_{w+k}([x_{w-1}, x_{w-2}, \cdots, x_0, 0, \cdots, 0]) &= \sum_{i=0}^{w-1} x_i 2^{i+k} \\ &= \left[\sum_{i=0}^{w-1} x_i 2^i\right] \cdot 2^k \\ &= x2^k \end{aligned}$$

■

当对固定字长左移 k 位时，其高 k 位被丢弃，得到

$$[x_{w-k-1}, x_{w-k-2}, \cdots, x_0, 0, \cdots, 0]$$

而执行固定字长的乘法也是这种情况。因此，我们可以看出左移一个数值等价于执行一个与 2 的幂相乘的无符号乘法。

原理：与 2 的幂相乘的无符号乘法

C 变量 x 和 k 有无符号数值 x 和 k，且 $0 \leqslant k < w$，则 C 表达式 x<<k 产生数值 $x * {}^{u}_{w} 2^k$。

由于固定大小的补码算术运算的位级操作与其无符号运算等价，我们就可以对补码运算的 2 的幂的乘法与左移之间的关系进行类似的表述：

原理：与 2 的幂相乘的补码乘法

C 变量 x 和 k 有补码值 x 和无符号数值 k，且 $0 \leqslant k < w$，则 C 表达式 x<<k 产生数值 $x * {}^{t}_{w} 2^k$。