


2) 说明 p 可以写成这样的形式: $p = x \cdot q + r$, 其中 $|r| < |x|$ 。

3) 说明 $q = y$ 当且仅当 $r = t = 0$ 。

 **练习题 2.36** 对于数据类型 `int` 为 32 位的情况, 设计一个版本的 `tmult_ok` 函数(练习题 2.35), 使用 64 位精度的数据类型 `int64_t`, 而不使用除法。

旁注 XDR 库中的安全漏洞

2002 年, 人们发现 Sun Microsystems 公司提供的实现 XDR 库的代码有安全漏洞, XDR 库是一个广泛使用的、程序间共享数据结构的工具, 造成这个安全漏洞的原因是程序会在毫无察觉的情况下产生乘法溢出。

包含安全漏洞的代码与下面所示类似:

```
1  /* Illustration of code vulnerability similar to that found in
2   * Sun's XDR library.
3   */
4  void* copy_elements(void *ele_src[], int ele_cnt, size_t ele_size) {
5      /*
6       * Allocate buffer for ele_cnt objects, each of ele_size bytes
7       * and copy from locations designated by ele_src
8       */
9      void *result = malloc(ele_cnt * ele_size);
10     if (result == NULL)
11         /* malloc failed */
12         return NULL;
13     void *next = result;
14     int i;
15     for (i = 0; i < ele_cnt; i++) {
16         /* Copy object i to destination */
17         memcpy(next, ele_src[i], ele_size);
18         /* Move pointer to next memory region */
19         next += ele_size;
20     }
21     return result;
22 }
```

函数 `copy_elements` 设计用来将 `ele_cnt` 个数据结构复制到第 9 行的函数分配的缓冲区中, 每个数据结构包含 `ele_size` 个字节。需要的字节数是通过计算 `ele_cnt * ele_size` 得到的。

想象一下, 一个怀有恶意的程序员在被编译为 32 位的程序中用参数 `ele_cnt` 等于 $1\,048\,577(2^{20}+1)$ 、`ele_size` 等于 $4096(2^{12})$ 来调用这个函数。然后第 9 行上的乘法会溢出, 导致只会分配 4096 个字节, 而不是装下这些数据所需要的 $4\,294\,971\,392$ 个字节。从第 15 行开始的循环会试图复制所有的字节, 超越已分配的缓冲区的界限, 因而破坏了其他的数据结构。这会导致程序崩溃或者行为异常。

几乎每个操作系统都使用了这段 Sun 的代码, 像 Internet Explorer 和 Kerberos 验证系统这样使用广泛的程序都用到了它。计算机紧急响应组 (Computer Emergency Response Team, CERT), 由卡内基-梅隆软件工程协会 (Carnegie Mellon Software Engineering Institute) 运作的一个追踪安全漏洞或失效的组织, 发布了建议 “CA-2002-25”, 于是许多公司急忙对它们的代码打补丁。幸运的是, 还没有由于这个漏洞引起的安全失效的报告。

库函数 `calloc` 的实现中存在着类似的漏洞。这些已经被修补过了。遗憾的是, 许