```
1  /* Return y = Ax */
2  int *matvec(int **A, int *x, int n)
3  {
4      int i, j;
5
6      int *y = (int *)Malloc(n * sizeof(int));
7
8      for (i = 0; i < n; i++)
9          for (j = 0; j < n; j++)
10             y[i] += A[i][j] * x[j];
11     return y;
12 }
```

在这个示例中，程序员不正确地假设向量 y 被初始化为零。正确的实现方式是显式地将 y[i]设置为零，或者使用 calloc。

9.11.3　允许栈缓冲区溢出

正如我们在 3.10.3 节中看到的，如果一个程序不检查输入串的大小就写入栈中的目标缓冲区，那么这个程序就会有*缓冲区溢出错误*(buffer overflow bug)。例如，下面的函数就有缓冲区溢出错误，因为 gets 函数复制一个任意长度的串到缓冲区。为了纠正这个错误，我们必须使用 fgets 函数，这个函数限制了输入串的大小：

```
1  void bufoverflow()
2  {
3      char buf[64];
4
5      gets(buf); /* Here is the stack buffer overflow bug */
6      return;
7  }
```

9.11.4　假设指针和它们指向的对象是相同大小的

一种常见的错误是假设指向对象的指针和它们所指向的对象是相同大小的：

```
1  /* Create an nxm array */
2  int **makeArray1(int n, int m)
3  {
4      int i;
5      int **A = (int **)Malloc(n * sizeof(int));
6
7      for (i = 0; i < n; i++)
8          A[i] = (int *)Malloc(m * sizeof(int));
9      return A;
10 }
```

这里的目的是创建一个由 n 个指针组成的数组，每个指针都指向一个包含 m 个 int 的数组。然而，因为程序员在第 5 行将 sizeof(int *)写成了 sizeof(int)，代码实际上创建的是一个 int 的数组。

这段代码只有在 int 和指向 int 的指针大小相同的机器上运行良好。但是，如果我们在像 Core i7 这样的机器上运行这段代码，其中指针大于 int，那么第 7 行和第 8 行的循环将