Alas, there just isn't any handy way to express 45 in C. There's no cool symbol. For example, you cannot write

```
answer=4♠5;
```

This line just doesn't work, even considering that ♠ isn't a symbol anywhere on the keyboard (even in Las Vegas). It would be nice, but it just isn't so.

You need to draw on the C language's vast library of supplemental math functions. A few dozen functions compute such things as power-of, square root, terrifying trigonometric functions, and so on (see Table 25-1, later in this chapter). To take a number to a higher power, you use the `pow()` function.

The `pow()` function is used to calculate one value taken to a certain power, such as 4 taken to the second power (42). Here's the format:

```
value = pow(n,p)
```

In this line, `value`, `n`, and `p` all must be double-precision variables (defined using the `double` declaration). The `pow()` function calculates `n` to the `p` power. The answer is stored in the `value` variable.

To prevent your compiler from going mad, you must include the MATH.H header file at the beginning of your source code. Stick the following line up there somewhere:

```
#include <math.h>
```

This line is required for the `pow` function as well as for other math functions you may dare to use.

Putting `pow()` *into use*

Using the `pow()` function is easy — it's just like using any other function. Just declare your variables as doubles, include the MATH.H thing, and you're all set. Why not put it forth in one of those endearing math-sentence problems? To wit:

Suppose that you have tasked Milton — your brilliant, Mensa-club-leader son — with decorating the house for Christmas. Because he's too smart to drive, he directs you to go to the store and pick up 28 ("two to the eighth power") twinkly lights. You quickly dash off to your computer and use the C language to decipher what he means: