This time, the test is greater than *or* equal to: Is the number that is entered 5 or more than 5? If the number is greater than or equal to 5, it must be more than 4, and the `printf()` statement goes on to display that important info on the screen.

The following modification to the GENIE1.C program doesn't change the `if` comparison, as in the previous examples. Instead, it shows you that more than one statement can belong to `if`:

```
if(number<5)
{
    printf("That number is less than 5!\n");
    printf("By goodness, aren't I smart?\n");
}
```

Everything between the curly braces is executed when the comparison is true. Advanced C programs may have lots of stuff in there; as long as it's between the curly braces, it's executed only `if` the comparison is true. (That's why it's indented — so that you know that it all belongs to the `if` statement.)

- The comparison that `if` makes is usually between a variable and a value. It can be a numeric or single-character variable.
- `if` cannot compare strings. For information on comparing strings, refer to my book *C All-in-One Desk Reference For Dummies* (Wiley).
- Less than and greater than and their ilk should be familiar to you from basic math. If not, you should know that you read the symbols from left to right: The `>` symbol is *greater than* because the big side comes first; the `<` is *less than* because the lesser side comes first.

- The symbols for less than or equal to and greater than or equal to always appear that way: `<=` and `>=`. Switching them the other way generates an error.
- The symbol for "not" in C is the exclamation point. So, `!=` means "not equal." What is !TRUE (not-true) is FALSE. "If you think that it's butter, but it's !." No, I do ! want to eat those soggy zucchini chips.

- When you're making a comparison to see whether two things are equal, you use *two* equal signs. I think of it this way: When you build an `if` statement to see whether two things are equal, you think in your head "is equal" rather than "equals." For example:

  ```
  if(x==5)
  ```

  Read this statement as "If the value of the *x* variable *is equal* to `5`, then. . . ." If you think "equals," you have a tendency to use only one equal sign — which is very wrong.