Type the source code for IQ.C into your editor. The new deal here is the `getval()` function, which returns a value by using the `return` keyword. (Does that look familiar? I tell you more in a second!)

Save the source code file to disk as IQ.C.

Compile. Run.

Here's what the sample output may look like, using fictitious figures for myself:

```
Enter your age:33
Enter your weight:175
Enter your area code:208
The computer estimates your IQ to be 27.000000.
```

Of course. I knew my IQ was that high. I'm not boasting or anything. It's only an estimate, after all.

- By using this formula, only old, fat people living in low-numbered area codes can get into Mensa.
- This program has some problems. For example, the IQ value that's calculated should be a floating-point number, and it's not (unless your age, weight, and area code are very special). This problem is fixed in the nearby sidebar, "Fixing IQ.C by using the old type-casting trick."
- Note how `getval()` is defined as an integer function. Inside `getval()`, an integer value is produced by the `atoi()` function. It's saved in the *x* variable, which is then returned to the main function by using the `return(x);` statement. Everything is an integer, so the function is of that type as well.
- In the `main()` function, `getval()` is used three times. The values it produces (what it functs) is saved in the `age`, `weight`, and `height` integer variables, respectively.
- Yeah, you probably lied too when you entered your weight. Don't! The more tumid you are, the smarter the program makes you.

Return to sender with the `return` keyword

Functions that return values need some type of mechanism to send those values back. Information just can't fall off the edge, with the compiler assuming that the last curly brace means "Hey, I must return the variable, uh, *x*. Yeah. That's it. Send *x* back. Now I get it."