Fixing IQ.C by using the old type-casting trick

In the IQ.C source code, the computer estimates your IQ based on this formula:

```
iq=(age*weight)/area;
```

That is, your IQ is equal to your age multiplied by your weight, with that total divided by your area code. All those variables are integers, and, incidentally, it's the exact formula used by my kids' school district.

Alarm! Whenever you divide any two values, the result is probably a `float`. No, count on it being a `float`. That's just the way math works. Decimals and fractions — it's messy stuff.

To make this function work, the variable `iq` is declared as a `float` — right up at the beginning of the source code, just as it should. But there's a problem: The value calculated by the equation is still stuffed into an integer. (Eh?) Even though the calculated result probably has a decimal part, all those variables are integers, and the result is an integer. (Hey, the compiler doesn't assume anything, remember?)

To fix the problem, you must do something known as *type casting,* where you tell the compiler to temporarily forget what type of variable is there and instead assume that it's something else.

Edit Line 19 in the IQ.C source code to read:

```
iq=(float)(age*weight)/area;
```

Insert the word `float` in parentheses right after the equal sign. Save the file back to disk. Compile and run. Notice that your IQ changes to a more floaty number in the output:

```
The computer estimates your IQ
   to be 27.764423.
```

Now the number is a true `float`.

No. To properly return a value, you need the proper `return` keyword.

The `return` keyword is used to send a value back from a function, to return a value from the function. Here's the format:

```
return(something);
```

The *`something`* is a value that the function must return. What kind of value? It depends on the type of function. It must be an integer for `int` functions, a character for a `char` function, a string (which is tricky), a floater for a `float` function, and so on. And, you can specify either a variable name or a constant value.

```
return(total);
```

The value of the variable `total` is returned from the function.

```
return(0);
```

The function returns a value of zero.