Run the program.

```
Please enter the number to start
the countdown (1 to 100):
```

Be a traditionalist and type **10**. Press Enter:

```
T-minus 10
T-minus 9
et cetera . . .
T-minus 1
Zero!
Blast off!
```

- The `do while` loop executes once, no matter what.
- In a `do-while` loop, a semicolon is required at the end of the monster, after `while`'s condition (see Line 21 in the program).
- Don't forget the ampersand (&), required in front of the variable used in a `scanf` statement. Without that `&` there, the program *really* screws up.

`do-while` *details*

A `do-while` loop has only one advantage over the traditional `while` loop: It always works through once. It's as though `do` is an order: "Do this loop once, no matter what." It's guaranteed to repeat itself. If it doesn't, you're entitled to a full refund; write to your congressman for the details.

The `do` keyword belongs to `while` when the traditional `while` loop stands on its head. The only bonus is that the `while` loop always works through once, even when the condition it examines is false. It's as though the `while` doesn't even notice the condition until after the loop has wended its way through one time.

Here's the standard format thing:

```
do
{
      statement(s);
}
while(condition);
```