Carefully edit your SPEED.C source code so that it matches what you see listed here. Save the file to disk again and then recompile. It has the same output because your only change was to make the value `55` a real, live constant rather than a value inside the program.

- Again, the key is that it takes only one, quick edit to change the speed limit. If you edit Line 3 to read

  ```
  #define SPEED 45
  ```

  you have effectively changed the constant value `45` in three other places in the program. This change saves some time for the SPEED.C program — but it saves you even more time for longer, more complex programs that also use constant values.
- *Symbolic constant* is C technospeak for a constant value created by the `#define` directive.

The `#define` directive

The `#define` construction (which is its official name, though I prefer to call it a directive) is used to set up what the C lords call a *symbolic constant* — a shortcut name for a value that appears over and over in your source code. Here's the format:

```
#define SHORTCUT value
```

`SHORTCUT` is the name of the constant you're defining. It's traditional to name it like a variable and use ALL CAPS (no spaces).

`value` is the value the *`SHORTCUT`* takes on. It's replaced by that value globally throughout the rest of your source code file. The *`value`* can be a number, equation, symbol, or string.

No semicolon is at the end of the line, but notice that the line absolutely must begin with a pound sign. This line appears at the beginning of your source code, before the `main()` function.

You should also tack a comment to the end of the `#define` line to remind you of what the value represents, as in this example:

```
#define SPEED 55  /* the speed limit */
```

Here, `SPEED` is defined to be the value `55`. You can then use `SPEED` anywhere else in your program to represent `55`.