Maybe a little more help in understanding conversion characters

To drive home how `printf()` uses its formatting string and arguments, bring up the source code for the GOODBYE.C program into your text editor. Change Line 5 to read:

```
printf("%s","Goodbye, cruel
    world!\n");
```

`printf()` has been modified to contain a formatting string and an argument.

The formatting string is `%s`, which is the *string* (for *s*) placeholder.

The argument is a string of text: `"Goodbye, cruel world\!n"`.

Save the source code under a new filename, BYE.C. Compile and run. The output is the same as the original; you have merely used the `%s` in the `printf()` function to "format" the output.

Try this modification of Line 5:

```
printf("%s, %s
    %s\n","Goodbye","cruel",
    "world!");
```

Carefully edit Line 5 to look like what's shown in the preceding line. It has three string placeholders, `%s`, and three strings in double quotes (with commas between them). Save. Compile. Run. The output should be the same.

(If you get a compiling error, you probably have put a comma *inside* the double quotes, rather than between them.)

The JUSTIFY.C program shows you only a hint of what the `printf()` function can do. `printf()` can also format numbers in a remarkable number of ways, which is a little overwhelming to present right now in this chapter.

- In the `printf()` function, the first item in quotes is a formatting string, though it can also contain text to be displayed right on the screen.
- The percent character holds special meaning to `printf()`. It identifies a *conversion character* — what I call a "placeholder" — that tells `printf` how to format its output.
- The conversion character `s` means string: `%s`.
- Any numbers between the `%` and the `s` are used to set the *width* of the text string displayed. So, `%15s` means to display a string of text using 15 characters. A minus sign before the `15` means to left-justify the string's output.
- Doesn't "left-justify" sound like a word processing term? Yup! It's formatting!
- `printf()` doesn't truncate or shorten strings longer than the width specified in the `%s` placeholder.