Feel free to make the preceding modifications to your INSULT3.C program in your editor. Save the changes to disk as INSULT4.C. Compile. Run.

```
Name some jerk you know:
David
Yeah, I think
David
is a jerk, too.
```

The output looks funky, like one of those "you may be the first person on your block" sweepstakes junk mailers. But the program works the way it was intended.

- Rather than replace `printf()` with `puts()`, you have to rethink your program's strategy. For one, `puts()` automatically sticks a newline on the end of a string it displays. No more strings ending in `\n`! Second, `puts()` can display only one string variable at a time, all by itself, on its own line. And, last, the next bit of code shows the program the way it should be written by using only `puts()` and `gets()`.

- You must first "declare" a string variable in your program by using the `char` keyword. Then you must stick something in the variable, which you can do by using the `scanf()` or `gets` function. Only then does displaying the variable's contents by using `puts()` make any sense.

- Do not use `puts()` with a nonstring variable. The output is weird. (See Chapter 8 for the lowdown on variables.)

When to use `puts()`
When to use `printf()`

- Use `puts()` to display a single line of text — nothing fancy.
- Use `puts()` to display the contents of a string variable on a line by itself.
- Use `printf()` to display the contents of a variable nestled in the middle of another string.
- Use `printf()` to display the contents of more than one variable at a time.
- Use `printf()` when you don't want the newline (Enter) character to be displayed after every line, such as when you're prompting for input.
- Use `printf()` when fancy formatted output is required.