More Pitfalls for `scanf` than for `printf`

`scanf` has the same pitfalls as `printf`

- Be sure to **match format specifiers (and ordering) to variable types**.
- Be sure to **match number of specifiers to number of addresses** given.

And more!

- **Don't forget to write "&" before each variable.** (Behavior is again undefined, but can be quite difficult to find the bug.)

`printf` Returns the Number of Characters Printed

Function calls are expressions.

Both `printf` and `scanf` return `int` (the calls evaluate to values of type `int`).

`printf` returns the number of characters printed to the display (or < 0 on error).

Writing a `printf` followed by a semicolon

- evaluates the expression (calls `printf`),
- then discards the return value.

The return value of `printf` is rarely used.

`scanf` Returns the Number of Conversions

`scanf` returns the number of conversions performed successfully, or -1 for no conversions.

The return value is **important for checking user input**.

For example,

```
if (2 != scanf ("%d%d", &A, &B)) {
    printf ("Bad input!\n");
    A = 42; B = 10; /* defaults */
}
```

