## Administrivia

- Notes from Friday updated to say more about minute essay answer.
- Reminder: Homework 4 due Wednesday.
- Another quiz next Monday.

**Slide 1**

## Minute Essay From Last Lecture

- Almost no one got this right! Answer discussed in class Friday. Worth reviewing if it wasn't clear at the time.
- A cultural(?) note: The name of the mystery function (`foobar`) is one used often in CS when one needs a more or less meaningless name for something, along with variants `foo`, `bar`, and so forth. Apparently based on WWII-era acronym FUBAR.

**Slide 2**

Slide 3

## Functions/Recursion Example, Continued

- Last time we(?) started writing a function to get an integer from stdin without using `scanf`, as a more complicated example of using functions and recursion.

  It's more difficult than I really expect you to do at this point, but I thought was also more interesting, and illustrates how functions can help you break a problem down into somewhat more manageable pieces (first skip blanks, then collect digits).

- Initially I thought it would be good to have the function return the integer (if it gets one) and use a pointer argument to also "return" error information. But on second thought I think it makes more sense for it to return the status (error or not) and use a pointer argument to "return" the value, as `scanf` does. Why? for convenience in checking error status.

- Finish example . . .

Slide 4

## Binary Numbers — Review

- Binary numbers are "the same as" base-10 numbers except for the base — i.e., where each digit of a base-10 number represents a quantity times a power of 10, in base 2 (binary) the digits represent quantities times powers of 2.

- Converting from binary to decimal (base 10) is straightforward based on this definition.

- To convert from decimal to binary, easiest way is to repeatedly divide by 2, recording quotients and remainders, until the quotient is zero. The "bits" of the result are the remainders, right to left. (Why this works.)

Slide 5

## Octall and Hexadecimal Numbers — Review

- Octal (base 8) and hexadecimal (base 16) numbers are similar, with the twist that for base 16 we need more than 10 different digits, so we use the familiar base-10 digits plus letters A through F (upper- or lower-case).
- Converting between either of these bases and base 2 is easy. For binary to octal, group bits in groups of three, right to left, adding zeros to the left if needed. Each group of three bits represents one octal digits. For binary to hexadecimal, proceed similarly, using groups of four bits. (Why this works.)

Slide 6

## Minute Essay

- None — quiz.