Slide 1

Administrivia

- Reminder: Homework 3 due today. Okay to turn in Monday if you need help since I have not been around.
- Midterm to be rescheduled for — October 21? October 23? (I will send mail.)

Slide 2

Functions — Recap

- Functions are useful for
  - Breaking up a non-trivial program into manageable pieces.
  - Avoiding duplication of code.
- Parameters are passed to function “by value”; to allow function to return more than one value, pass “pointers” (roughly, addresses) rather than values.

Slide 3

Functions and Recursion

- Something else we want to be able to do is repeat something some fixed number of times, or until some condition is true — for example, in the converter program, prompt again if we get invalid input.
- We'll talk soon about some new constructs to do that, but we can do it now, with *recursion* — having a function call itself.

Slide 4

Recursion — Concepts

- Recursive function is one that calls itself.
- Obviously to make this work we need a way to stop recursing — a *base case* — otherwise we have something akin to the in-joke definition of GNU ("GNU is Not Unix").
- Also we need to be sure that every recursive call brings us closer to a base case.

Slide 5

Recursion — Implementation

- How it works: When you call any function, the current "state" (values of variables) is preserved ("pushed onto a stack"), and space is reserved for the called function's local variables (including parameters). When the function returns, this space is freed up again. So if we stack up recursive calls to the same function, each has its own copy of all local variables.
- Simple examples — factorial, Fibonacci numbers, counting.

Slide 6

Minute Essay

- Anything noteworthy to say about Homework 3?