Slide 1

Administrivia

- Reminder: Homework 5 design due today, code Tuesday.

Slide 2

Recursion — Overview

- Basic approach:
  - Identify “base case” — something you can solve directly.
  - Figure out how to decompose non-base cases into “smaller” problems, and apply algorithm to smaller problems.
- How to think about “does it work?”
  - Does it work for base case(s)?
  - Assuming recursive calls work, does it work for other cases?
  - Does every recursive call get you at least one step closer to a base case?
- Implementation — conceptually (and usually in fact) involves a stack of calls-in-progress.
- Can be slower than iteration (though sometimes not), but can also be much easier to understand.

Slide 3

Recursion — Simple Examples

- Factorial function.
- Function to compute Fibonacci numbers (very slow!).

Slide 4

Recursion — More Examples

- Linked list implementation. (If time permits?)
- Quicksort — pick “pivot” element, split array into elements less than pivot and elements greater than pivot, and sort recursively. Why does this work?
- Mergesort — split array (or list) into two pieces of equal size, sort recursively, merge. Why does this work?
- (Other example(s) as time permits.)

Slide 5

Minute Essay

- Given the following function

```
// assume n >= 0, m >= 0
int foo(int n, int m) {
  if (n == 0) return m;
  else return 1 + foo(n-1, m);
}
```

- What is `foo(3, 5)`?
- What (in words) does `foo` do?

Slide 6

Minute Essay Answer

- `foo(3, 5)` is 8.
- `foo` adds its two arguments.