Slide 1

Administrivia

- Midterm next Thursday. Review sheet, solutions to be available.
- Quiz solutions online. (Review Quiz 2 questions.)

Slide 2

Recursion — Review/Recap

- A function (or definition) is recursive if it calls/uses itself. Obviously(?) there needs to be at least one base case too.
- Can be somewhat tricky to think about whether/how recursive functions work — it involves nested calls to the same function, one "inside" the other in some sense. May be helpful to take what I call a "static" perspective, focusing on the code and one call to the function rather than the whole bunch of nested calls.
- To do that, first be clear on what the function does — "computes n factorial", or "computes the sum of array elements starting at this index". Then ask . . .

Slide 3

Recursive Functions — "(How) Does it Work"?

- (How) does it work for the base case(s)?
- (How) does it work for the non-base cases, *assuming that the recursive calls work*, meaning that they do what the function is supposed to do, based on the definition you came up with.
- (How) does each recursive call get us closer to a base case?
- (In some ways this is a mirror image of induction, as in proofs by induction, where we start with small cases and construct more complex ones.)

Slide 4

Arrays and Lists — Review/Recap

- Scala provides two basic types of "sequences", arrays and lists.
- Several ways to work with them. We start out by applying tools we already have (recursive functions), partly to get more practice with them. Also an opportunity to revisit "higher-order functions" (functions that use other functions as parameters) . . .

Slide 5

Arrays and Lists and Recursion, More Examples

- We started with functions to read numbers into an array or a list and print them out. What else can we do with them? Lots of things . . .
- We could write functions that take a collection of numbers and return a single number. Examples include sum, product, max, min, . . .
- But all of these functions basically do the same thing, right? the only thing that's different is how we combine two numbers into one. So maybe what we really want is a function one of whose parameters is a function . . .

Slide 6

Higher-Order Functions — Review/Recap

- "Higher-order functions" (first discussed in chapter 5) are functions that use other functions as parameters (or as return values). Very useful concept, supported in fairly different ways in different languages.
- As an example of how this is useful — summing all elements of an array versus computing their product, versus finding the smallest or largest element, etc. Basic computation (a *reduction*) involves combining elements pairwise with a binary operator, and by using a higher-order functions we don't have to repeat the parts that are the same.

Slide 7

Defining Higher-Order Functions in Scala

- Syntax illustrated by an example for our demo program(s):

```
def arrayCombine(a : Array[Int], startIndex : Int,
  combine : (Int, Int) => Int, identity : Int) : Int = { /* .... */ }
```

  where `combine` is a parameter that is itself a function(!).

  (I could have put all of that on one line, but it would have been long.)

- Within the body of the function (`arrayCombine` in the example) we can call the parameter function (`combine`) as we usually do, e.g., `combine(1, 2)` to call the function with parameters 1 and 2.

Slide 8

Using Higher-Order Functions in Scala

- One option for function parameters is a named function:

```
def add(x : Int, y : Int) : Int = { x + y }
arrayCombine(a, 0, add, 0)
```

- Another option is a function literal:

```
arrayCombine(a, 0, (x, y) => ( x + y ), 0)
```

- Yet another option is a special form of a function literal:

```
arrayCombine(a, 0, _ + _, 0)
```

Slide 9

Example(s) Revisited

- We could now revise our array demo program to do still more things with the array — find minimum and maximum elements, for example.
- We could add similar functionality to our list demo program.

Slide 10

Collection Methods — Overview

- As noted earlier, both arrays and lists provide a wide range of interesting(?) methods. ("Methods"? Briefly, special type of functions, described a bit in chapter 3.) The textbook lists some of them and is a good starting point. For full details, however . . . (To be continued.)

Slide 11

Minute Essay

- What are you finding interesting, difficult, noteworthy about the homeworks?