Slide 1

Administrivia

- (None.)

Slide 2

Minute Essay From Last Lecture

- Many people found these problems significantly more difficult than the earlier ones. Probably so! (But I think this is not the start of a trend.) Remember that help is available — from me, or from the ACM tutors (though some of them do not know Scala well or at all).
- One person commented about the counting up/down problem that it was not trivial to think of all the possibilities. True! and this is one of the things that makes programming interesting but challenging.

Slide 3

Lists in Scala

- As with arrays, there are two basic ways to make lists in Scala.
- One is similar to how you create an array by listing elements:

  ```
  val l1 = List(1,2,3,4)
  ```

- Another is to build it up an element at a time with the "cons" operator (`::`):

  ```
  var ll = List[Int]()
  ll = 1::ll
  ```

  (You would likely not write exactly that code; it's meant only to illustrate use of the `::` operator.)

Slide 4

Lists in Scala, Continued

- You *can* get the length of a list and values of elements using the same syntax as with arrays, but that's not very idiomatic.
- Better is to use to use `head` and `tail` and recursion. Let's write something analogous to the array demo from last time . . .

Slide 5

Arrays and Lists and Recursion, More Examples

- Now we have functions to read numbers into an array or a list. What can we do with them? Lots of things . . .
- As an example, we could write some 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

Minute Essay

- None — quiz.