Slide 1

Administrivia

- Midterm next Thursday. I will post a "review sheet" with a list of topics, and sample solutions for homeworks, probably this weekend but at least by class time Tuesday.

Slide 2

Arrays and Lists — Overview

- With what we've done so far we have enough tools to compute anything we want to compute. (And in combination with input/output redirection we could probably do some interesting things.)
- However, some things are awkward (repetition), and we don't yet have a convenient way to store many values — something similar to subscripted values in math.
- Most programming languages give you a way to represent *collections*. Exactly what you get depends on the language — e.g., C gives you only something quite primitive (but close to what the hardware can do), Java gives you something more abstract/useful, and Scala goes even further.

Slide 3

Arrays and Lists in Scala

- Scala provides two ways of representing a "sequence" (ordered list of elements), `Array`s and `List`s. From the outside they look very similar, but they behave fairly differently:

  An `Array` has a fixed number of elements, but the values of the elements can be changed.

  A `List` cannot be changed at all, but there are easy and efficient ways to build lists.

- Both are "parameterized types", which means you can specify the type of the elements.

Slide 4

Arrays in Scala

- Two syntaxes for creating an `Array`. Examples:

```
// four elements, initial values as given
val a1 = Array(1,2,3,4)
// ten elements, all zero
val a2 = new Array[Int](10)
```

- Syntax for referencing element uses name of array plus index in parentheses. Indexes range from 0 through length minus 1. Examples:

```
println(a1(1))
a2(2) = 20
```

Slide 5

Arrays in Scala, Continued

- Length of `Array` can be obtained with `.length` or `.size`.
- That gives us enough to write some simple functions using recursion . . .

Slide 6

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 7

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 . . .

Slide 8

Minute Essay

- None — quiz.