Slide 1

Administrivia

- Reminder: Homework 3 due Tuesday. Okay to turn in Wednesday since there was a delay in getting it on the Web.
- Notice that quiz solutions are on the Web (usually posted shortly after the quiz).

Slide 2

Recursion for Repetition — Review/Recap

- One way to repeat something a fixed number of times, or until some condition is true, is with recursion.
- Examples last time included factorial, "count down". (Notice that we can easily make the function a complete program/script by just adding something to the end to get input from the user. Sample program `roots.scala` shows an example, or `countdown.scala` from last time.)
- Example in book of using recursion to compute sum of numbers.
- Another example — make our rather sketchy all-purpose conversion program keep asking for input until the user says to quit, rather than doing only one conversion.

Slide 3

One More Conditional Construct — Match

- Notice in the conversion program that we have a lot of if/else's testing the same variable against various values. Somewhat repetitive, no? Some languages provide more-compact way to express this (e.g., `switch` in C and Java).
- Scala provides something more general — `match`. Allows matching specified variable with multiple conditions . . .

Slide 4

Match, Continued

- Simple example:

```
val c = readChar
n match {
  case 'a' => println("found a")
  case 'b' => println("found b")
  case _ => println("not a or b")
}
```

  This is already more powerful than what some languages provide, in that you can match strings. Much more is possible. Details later.

- We could use this to improve(?) the converter program . . .

Slide 5

Sidebar — Input/Output Redirection

- In beginning programming classes we often talk about getting input "from the keyboard". What if you want to read a lot of input, though, and maybe do it more than once (e.g., you want to confirm that after making a change to your program it still works for the inputs you tried before)?
- Strictly speaking, `readInt`, etc., do not read from the keyboard, but from "standard input". What's the difference? Many environments (including typical UNIX/Linux command shells) allow you to "redirect standard input" to indicate that it should come from something other than a human at a keyboard.
- Similarly, `println` doesn't write to "the terminal" but to "standard output".

Slide 6

Sidebar — Input/Output Redirection, Continued

- Programs can get input from files:

  `scala mypgm.scala < infile`

  where `infile` is a plain-text file containing input.
- Programs can put output in files:

  `scala mypgm.scala > outfile`

  (Use $>>$ to append rather than overwriting.)

Slide 7

Sidebar — Input/Output Redirection, Continued

- Or we can use the output of one program as the input of another ("pipe" the output of one to another):

  `scala pgm-to-make-nums.scala | scala sum.scala`

  One use for this — if a program produces so much output it scrolls off the screen, pipe it to `less`.

  Nitpick/caveat: Some output (usually error messages) is written to "standard error" rather than standard output, and it doesn't get redirected unless you ask for it to be. A syntax that works in our environment is to follow the $>$ or $|$ symbol with an ampersand (&).

- (This is one of the reasons some people like command-line environments — you can do a lot with them if you know how.)

Slide 8

Arrays and Lists — Preview

- With what we've done so far we have enough tools to compute anything we want to compute.
- 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. (Think about writing some sort of drawing program, one for which our bounding-box function might be useful. Probably you want to somehow store a lot of rectangles or more-general shapes. How?)
- 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 9

Minute Essay

- Can you think of (other) situations in which redirecting program input or output might be useful?