Slide 1

Administrivia

- Reminder: Homework 2 due today (11:59pm).
- Homework 3 will be on the Web tomorrow. Due in a week. I will send mail.
- ACM is offering tutoring, by appointment. Details by e-mail.
- We need to find out how many students are planning to take CSCI courses in the spring. Please reply to the e-mail about this.

Slide 2

Minute Essay From Last Lecture

- For many of you, access to a computer on which you can try things is problematical. We need to resolve this.
- Script on “sample programs” page that you can use to turn in homework from the command line. (Might be useful if you’re working remotely using PuTTY.)

Slide 3

Conditional Execution — Review/Recap

- Programs can select which path of execution to follow using `if`/`else`. Works with statements and in expressions.
- To select from multiple possibilities, can nest this construct, or chain with `else if` (as in "grades" example). Or . . .

Slide 4

One More Conditional Construct — Match

- There are situations in which we have a lot of if/else code 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 5

Match, Continued

- Simple example:

```
val c = readChar
c 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.

Slide 6

Functions in Programming — Motivation

- Writing the same, or similar, code over and over is tedious and error-prone. Can we somehow capture frequently-done computations in a way we can reuse?
- Non-trivial programs can be huge (as much as tens of millions of lines of code for an operating system, e.g.). Humans are not (usually?) very good at understanding large and complicated things — need some way to "manage complexity".

Slide 7

Functions in Programming

- Functions are one way to solve both problems (code reuse and managing complexity). Most if not all programming languages provide some way to do this (possibly under another name, e.g., procedures).
- Similar, but not identical, to functions in mathematics.

  In math, a function has a domain and a range, and maps elements of the domain to elements of the range.

  In programming, a function's domain is represented by the number and types of its *parameters* (a.k.a. arguments), and its range is represented by a *return type*.
- A key difference is that functions in programming can have "side effects" — effects other than mapping input(s) to output(s).

Slide 8

Functions in Scala

- To create a function in Scala, you use the keyword `def` and then give the function's name, parameters, return type, and some code. Return type can be omitted if the function will be used only for its side effects.
- Very simple examples:

```
def sum(x : Int, y : Int) : Int = x + y
def hello() { println("hello") }
```

Slide 9

Functions in Scala, Continued

- To use a function, give its name and values for parameters.
- Very simple examples:

  `sum(10, 2)`

  `hello()` (can omit parentheses)

Slide 10

Example — Grades Program Revisited

- Now look again at the grades program. It starts by prompting for several input values. It might be nice to include in the prompt a maximum value. So we would be writing similar code over and over. Let's make a function for that.
- (Also, let's give names to those maximum values, to make the code more understandable to humans.)

Slide 11

Example — Finding Roots of a Quadratic Equation

- As a rather math-y example, let's write a function to compute and print the roots of a quadratic equation

$$ax^2 + bx + c = 0$$

- We'll use the formula

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

  and try to account for as many cases as we can . . .

- (We will just write the function for now, and test it interactively — load it into the REPL with `:load`.)

Slide 12

Minute Essay

- None — quiz.