Slide 1

Administrivia

- Homework 4 on the Web; due next Monday.

Slide 2

Quotes of the Day/Week/?

- From a key figure in the early days of computing:

  “As soon as we started programming, we found to our surprise that it wasn’t as easy to get programs right as we had thought. Debugging had to be discovered. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent finding mistakes in my own programs.” (Maurice Wilkes: 1948)

- From someone in a discussion group for the Java programming language:

  “Compilers aren’t friendly to anybody. They are heartless nitpickers that enjoy telling you about all your mistakes. The best one can do is to satisfy their pedantry to keep them quiet :)”

Slide 3

Sidebar: Tracing Code

- Something I sometimes ask you to do on quizzes/exams is tell me what a program prints out (without just typing it in). To do this you need to “trace the program”, “play computer”, etc.
- (This can be a useful skill when your programs don’t give the answers you want.)
- To do this: Work through the program statements in the same order the computer does, writing down values for variables. (Example: Counting-change function.)

Slide 4

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 a function a complete program/script by just adding something to the end to get input from the user. Let’s do that for `countdown.scala` from last time.)

Slide 5

More Examples

- Another example of recursion from math — Fibonacci numbers.
- Another example of recursion — bank-balance program that repeatedly prompts for check, deposit, or “quit” to end. (Not maybe very interesting/practical now, but once we have arrays/lists maybe . . .)

Slide 6

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.
- 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.

Minute Essay

- None — quiz.