Slide 1

Administrivia

- Review sheet for final on Web. Also sample solutions for quizzes and all homeworks (soon). Sample solution for midterm in hardcopy today.
- Homework grades coming by e-mail when available.
- Due dates for all homeworks have passed, but I will accept late work for partial credit through Friday.
- Extra-credit problems — if there is still interest, I will put something on the Web by tomorrow sometime. To be due early next week.
- I will have office hours this week. Details by e-mail soon.

Slide 2

Course Recap

- Recall from first class — course is an "introduction to programming."
- Ideally, a first course would focus more on ideas of programming than details — except that, in the words of a colleague

  "Programming is not a spectator sport."

  so we have to choose a programming language, and an environment, and then it's difficult *not* to get caught up in the details.

Slide 3

Course Recap, Continued

- Course was originally designed to meet the needs of two audiences:
  - Prospective CS majors/minors.
  - Majors in other fields where programming is a useful skill (math, engineering, sciences).

  Later, course was added to common curriculum.
- Meeting the needs of all these diverse groups — may not even be possible, at best one must compromise.

Slide 4

Course Recap, Continued

- Many possible choices for a first programming language/environment.
- In this course, we use simple tools because we think this works best for our original target audience — helps you understand what's going on at a fairly low level. However, we're using a fairly high-level/abstract language (Scala) because you can solve more interesting problems sooner.
- Once you know *one* programming language, the next one is easier, and the next one easier still. (Caveat: "Easier" is relative to how similar language is to one you already know. Programming languages come in groups — procedural, functional, object-oriented, etc.)

Slide 5

What I Hope You Got From This Course

- A basic understanding of what programming is — expressing a problem and its solution as "an algorithm" and turning that into code.
- A good enough foundation in *some* programming language to let you write programs using whatever tools/environment are appropriate to your field.
- If you never write another program — now you know what "source code" is, and you've done something most people don't have any idea how to do, and you've done it using tools that are not especially novice-friendly!
- (If you found this course difficult — many people do! Subject is not easy, and a lot of material to cover.)
- (Look briefly at an example of code that does something more complicated / interesting?)

Slide 6

Minute Essay

- How did the course compare to your expectations/goals? Did you learn what you hoped to learn?