Slide 1

Administrivia

- All homework should be turned in ASAP. Solutions are available for all (hardcopy only).

  "Due date" for Homework 6 (for the purpose of assessing late penalties) will be noon tomorrow.

  You can still turn in old assignments, for very reduced credit, up to the point at which you pick up solution(s).
- Office hours this week slightly different (as noted in e-mail yesterday). I will probably be around from 3:30pm to 4:20pm today as well.
- (Briefly review policy on collaboration.)

Slide 2

Proving Things About Loops — Example Revisited

- Suppose we have

  while $x > 0$ **do**

  $x := x - 1$

  end while

  with x an integer variable.
- Show that after the loop $x = 0$.

Slide 3

Things To Notice About Loop Invariants — Recap

- They're not unique — could come up with many "invariants" for a given loop. (This is true about preconditions in general.)
- The goal is to find one that's "useful" — if true at end of the loop with loop test false, helps us prove desired postcondition.
- Sometimes helps to think in terms of "what do the variables mean?" (E.g., `count` and `sum` from silly multiplication-by-addition example from last time.)
- Writing down a loop invariant can help (e.g., to avoid off-by-one errors) even if you don't do a complete formal proof.

Slide 4

Proofs of Program Correctness — Recap/Evangelism

- Many examples we looked at are trivial — mostly because they're all we can do in the time we have. (Textbook's proof that Euclid's algorithm works is a notable exception.) Keep in mind, though:
  - How to make this practical, and/or how to have it done by a smart program, are subjects of ongoing research.
  - In my opinion/experience, applying these ideas informally helps you "reason about programs". ("What do you know about the program variables at this point?" "What is this variable supposed to represent, and does the code support that?")
  - Similar ideas are very useful in reasoning about concurrent algorithms, which otherwise can be *very* tricky!

Slide 5

Propositional and Predicate Logic — Things to Review

- Translating (relatively simple) English into formulas.
- Evaluating whether a formula is true given a particular assignment of values to variables (propositional logic) or a particular interpretation (predicate logic).
- Proving formulas are always true / always valid using proof rules.

Slide 6

Proof Techniques — Things to Review

- Setting up proof / "proof obligations". Examples in minute essay for February 10.
- Proving something true versus finding a counterexample.
- Proofs by induction. Recall that there are two versions, "first principle" and "second principle". Review last problem-to-turn-in on Homework 3 for example of when the latter is useful.

Slide 7

Recursion and Recurrence Relations — Things to Review

- Recursive definitions. Note the overall idea — define larger cases in terms of smaller ones.
- Defining recurrence relations — e.g., problem about bats in homework, some of algorithm-analysis problems.
- Solving recurrence relations. Two methods discussed:

  Expand/guess/verify — can work for all, but requires proof by induction ("verify").

  Using formulas — easier in some ways, but only works for problems that fit. Which one to use, how to apply. (Tricky part of the latter seems to be plugging $g(n)$ into formula.)

Slide 8

Analysis of Algorithms — Things to Review

- General idea — point is to estimate time by counting some sort of basic operations.
- Defining and solving recurrence relations for recursive algorithms. Focus on the "code" for the algorithm, as described in sample solution for Homework 5.

Slide 9

Proofs of Program Correctness — Things to Review

- What it means to write $\{ Q \}\ P\ \{ R \}$.
- "Proof obligations" again — to prove that an if/then/else or loop is correct, what do you have to do?

Slide 10

Minute Essay

- None — sign in.