Slide 1

Administrivia

- Next homework coming soon (watch your mail).

Slide 2

Minute Essay From Last Lecture

- Question: Given program P as follows:

  if $x \geq 0$ **then**

  $x := x * 2$

  else

  $x := -x$

  end if

  We can show that $\{ \ x \neq 0 \ \} \ \ P \ \ \{ \ x \neq 0 \ \}$

  by showing that two other Hoare triples are true — what are they? (No need to say why they're true, just what they are.)

- Answer? (Also, how would you structure a full proof?)
- A little advice about writing up proofs: The idea is to present a logical argument that a human reader (a classmate, e.g.) can follow. Putting in some prose often helps!

Slide 3

Program Correctness and Loops, Review

- Our rule is this: For program P of the form

  while B **do**
  P_1
  end while

  if we also have a "loop invariant" Q, such that

  $$\{ \, Q \, \wedge \, B \, \} \;\; P_1 \;\; \{ \, Q \, \}$$

  then we can derive

  $$\{ \, Q \, \} \;\; P \;\; \{ \, Q \, \wedge \, B' \, \}$$

- Strictly speaking, we also have to prove that the loop does terminate — can do this by finding an integer function ("metric") that decreases every time through and when not positive means B is false.

Slide 4

Program Correctness and Loops, Continued

- Things to notice about loop invariants:
  - 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?"
  - 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.
- Example — silly program to compute $z = x \times y$ by repeated addition:

  $i := 0; z := 0;$
  while $i < x$ **do**
  $z := z + y; i := i + 1$
  end while

Slide 5

Program Correctness and Loops, Continued

- Another example — Euclid's algorithm for finding GCD (greatest common divisor, a.k.a. largest common factor) of a and b:

  $i := a; j := b;$
  while $j \neq 0$ **do**
  $q := i/j; r := i\%j;$
  $i := j; j := r;$
  end while

  At end, $i = gcd(a, b)$. It does?! Yes, and we can prove it, even if we don't quite understand *why*.

  Proposed invariant (using book's subscripting notation):

  $gcd(i_n, j_n) = gcd(a, b)$

Slide 6

Proofs of Program Correctness, Recap

- Many examples we looked at are trivial — mostly because they're all we can do in the time we have. 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!

Minute Essay

- How are you doing with the reading so far? Does the textbook explain things in a way that makes sense to you? Do you try working through some/all of the practice problems, and if so does that help?

Slide 7