Slide 1

Administrivia

- Reminder: Quiz 3 Wednesday. Will cover material on proof techniques and proofs by induction.
- Homework 4 on Web. Due next Monday.

Slide 2

Program Correctness and Loops — Review

- For program P_1 as follows

  while B **do**
  P
  end while

  and Q an invariant of the loop in P_1, we can say that

  $$\{\ Q\ \}\ \ P_1\ \ \{\ Q\ \wedge\ B'\ \}$$

- The idea is to choose Q such that the postcondition in the above triple ($Q\ \wedge\ B'$) is useful — i.e., helps establish something we want to be true after the loop.

Slide 3

Trivial Example

- 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 4

Correctness of Loops, Continued

- The textbook isn't very explicit about this, but strictly speaking we have something else to prove — that the loop terminates!
- Can do this with a "metric" (think "measure") — integer function of program variables that decreases every time through the loop, and when it's less than or equal to zero the loop stops.
- In the silly example, we could use what?

Slide 5

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 6

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 7

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!

Slide 8

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?