Slide 1

Administrivia

- Reminder: Homework 6 due Wednesday *at class time*. (This is so I can distribute a solution.)
- Reminder: Midterm Friday. Review sheet on Web. We will review more Wednesday.
- Midterm averages to be mailed later today.

Slide 2

Specifications and Correctness (Review)

- If we have a program P, and a specification consisting of precondition Q and postcondition R, we write

$$\{\, Q \,\} \;\; P \;\; \{\, R \,\}$$

to mean that if we start in a state where Q is true and run P, we end in a state where R is true. (Also, P terminates — no "infinite" loops.)

- Example:

$$\{\, x \geq 0 \,\} \;\; y := sqrt(x); \;\; \{\, y \geq 0 \,\wedge\, y^2 = x \,\}$$

Slide 3

Proofs of Program Correctness (Review)

- So far we have rules for
  - Assignment.
  - Sequential composition.
  - If/then/else.
  - Strengthening preconditions, weakening postconditions.
- What we still need is something to deal with loops . . .

Slide 4

Semi-Intermezzo: A Puzzle

- Suppose you have a jar containing white marbles and black marbles, plus an unlimited supply of extra black marbles, and you do the following:
  1. Select two marbles.
  2. If they're the same color, discard them both and put a black marble in the jar. If they're different colors, discard the black one and put the white one back in the jar.
  3. If there are at least two marbles in the jar, repeat.
- Does this end? If it does, what if anything can you say about the marble(s) in the jar when it ends?
- (Similar ideas behind "metric" for loop termination and "invariant" for loop correctness.)

Program Correctness and Loops

- We'll write loops in this form

```
**while** $B$ **do**
    $P$
**end while**
```

  After the loop terminates (assuming it does), what do we know about B? True or false?

- We also need the notion of a "loop invariant" — a predicate that, if true before we execute the loop body is true again after. More formally, Q is an invariant for the above loop if

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

- Now we can state the rule for loops . . .

Program Correctness and Loops

- 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

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

- We could prove this using induction (on the number of trips through the loop).
- 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.

Trivial Example

Slide 7

- Suppose we have

```
while $x > 0$ do
        $x := x - 1$
end while
```

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

Correctness of Loops, Continued

Slide 8

- 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 9

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 10

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):

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

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

- Do you feel relatively well-prepared for the midterm?
- (If you have requests for particular topics to review Wednesday — say so now, or send me e-mail.)