### Announcements

- **Prelim 2 conflicts**: Today (April 2) is two weeks before the prelim, and the deadline for submitting prelim conflicts.
- **Instructor travel**: This week and the next two weeks, Profs. Lee and Marschner will be traveling on and off. Instructor office hours are unaffected, though there will sometimes be just one of us available.

### Some Important Terminology

- **assertion**: true-false statement placed in a program to assert that it is true at that point
  - Can either be a comment, or an `assert` command
- **precondition**: assertion placed before a statement
  - Same idea as function precondition, but more general
- **postcondition**: assertion placed after a statement
- **loop invariant**: assertion supposed to be true before and after each iteration of the loop
  - Distinct from attribute invariant
- **iteration of a loop**: one execution of its repetend

### Assertions versus Asserts

- **Assertions prevent bugs**
  - Help you keep track of what you are doing
  - Also track down bugs
- **Precondition**: assertion placed before a segment
- **Postcondition**: assertion placed after a segment

### Preconditions & Postconditions

- **Precondition**: assertion placed before a segment
- **Postcondition**: assertion placed after a segment

### Invariants: Assertions That Do Not Change

- **Loop Invariant**: an assertion that is true before and after each iteration (execution of repetend)

### Solving a Problem

- What statement do you put here to make the postcondition true?

### Relationships Between Two

- If precondition is true, then postcondition will be true

---

**CS 1110 Spring 2013: Loop Invariants**
Invariants: Assertions That Do Not Change

x = 0; i = 2
# Inv: x = sum of squares of 2..i-1
while i <= 5:
    x = x + i * i
    i = i + 1
# Post: x = sum of squares of 2..5

Integers that have been processed: 2, 3, 4, 5
Range 2..5

Invariant was always true just before test of loop condition. So it's true when loop terminates

Designing Integer while-loops

1. Recognize that a range of integers b..c has to be processed
2. Write the command and equivalent postcondition
3. Write the basic part of the for-loop
4. Write loop invariant
5. Figure out any initialization
6. Implement the repetend (process k)

# Process b..c
Initialize variables (if necessary) to make invariant true
# Invariant: range b..k-1 has been processed
while k <= c:
    # Process k
    k = k + 1
# post: range b..c has been processed

Finding an Invariant

# Make b True if no int in 2..n-1 divides n, False otherwise
b = True
k = 2
# invariant: b is True if no int in 2..k-1 divides n, False otherwise
while k < n:
    # Process k; if n % k == 0:
    b = False
    k = k + 1
# b is True if no int in 2..n-1 divides n, False otherwise

What is the invariant?

A: k = 0; c = s[0]
B: k = 1; c = s[0]
C: k = 1; c = s[1]
D: k = 0; c = s[1]
E: I don’t know

Reason carefully about initialization

1. What is the invariant?
2. How do we initialize c and k?

A: k = 0; c = s[0]
B: k = 1; c = s[0]
C: k = 1; c = s[1]
D: k = 0; c = s[1]
E: None of the above

An empty set of characters or integers has no maximum. Therefore, be sure that 0..k-1 is not empty. You must start with k = 1.