CS1110 18 October 2011

Read: Sec. 2.3.8 and chapter 7 on loops.
The lectures on the ProgramLive CD can be a big help.

Some anagrams

A decimal point	I'm a dot in place	Animosity	Is no amity
Debit card	Bad credit	Desperation	A rope ends it
Dormitory	Dirty room	Funeral	Real fun
Schoolmaster	The classroom	Slot machines	Cash lost in 'em
Statue of liberty	Built to stay free	Snooze alarms	Alas! No more Z's
The Morse code	Here come dots	Vacation times	I'm not as active
Western Union	No wire unsent	George Bush	He bugs Gore
Parishioners	I hire parsons	The earthquakes	That queen shake

Circumstantial evidence — Can ruin a selected victim
Victoria, England's queen — Governs a nice quiet land
Eleven plus two — Twelve plus one (and they have 13 letters!)

1

A4 last sem: mean: 94.7, median: 98, std dev: 6.3

this sem 125 graded: mean: 96.4, median: 99, std dev: 5.9

Time spent: mean: 6.4 median: 6 (84 people)

2-2.5:	4	08:	13
3-3.5:	8	09:	10
4-4.5:	10	10:	5
5-5.5:	18	12:	1
6-6.5:	09	16:	1
7-7.5:	08	20:	1

assertion: true-false statement, sometimes placed in a program to *assert* that it is true at that point.
precondition: assertion placed before a statement
postcondition: assertion placed after a statement
loop invariant: assertion supposed to be true before and after each iteration of the loop
iteration of a loop: one execution of its repetend
We describe a methodology for developing for-loops.

2

Assertion: true-false statement (comment) asserting a belief about (the current state of) your program.

// x is the sum of 1..n <- asserts a specific relationship between x and n

x ? n 1 x ? n 0

x ? n 3

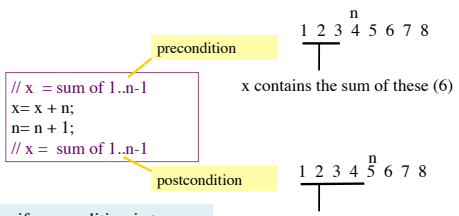
Assertions help prevent bugs by helping you keep track of what you're doing …
… and **they help track down bugs** by making it easier to check belief/code mismatches

assert <boolean expression> ;

Java assert statement. To execute: if the bool exp is false, stop with an error message

3

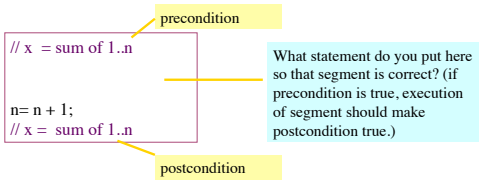
Precondition: assertion placed before a segment
Postcondition: assertion placed after a segment



Meaning: if precondition is true, then after executing the segment the postcondition will be true

4

Solving a problem

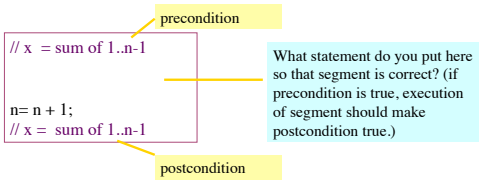


What statement do you put here so that segment is correct? (if precondition is true, execution of segment should make postcondition true.)

A. x= x + 1;
B. x= x + n;
C. x= x + n+1;
D. None of A, B, C
E. I can't figure it out

5

Solving a problem



What statement do you put here so that segment is correct? (if precondition is true, execution of segment should make postcondition true.)

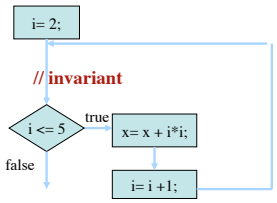
A. x= x + 1;
B. x= x + n;
C. x= x + n+1;
D. None of A, B, C
E. I can't figure it out

6

Invariants: another type of assertion

An invariant is an assertion about the variables that is true before and after each iteration (execution of the repetend).

```
x= 0;
for (int i= 2; i <= 5; i= i +1) {
   x= x + i*i;
}
// {R: x = sum of squares of 2..5 }
```



Invariant:
x = sum of squares of 2..i-1

in terms of the range of integers that have been processed so far

The loop processes the range 2..5


```
// Process integers in a..b
// inv: the integers in a..k-1 have been processed
for (int k= a;  k <= b;  k= k + 1) {
        Process integer k;
}
// post: the integers in a..b have been processed
```

Command to do something

equivalent post-condition

Methodology for developing a for-loop

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
for (int k= b;  k <= c;  k= k+1) {
   // Process k
}
// Postcondition: range b..c has been processed
```


Finding an invariant

```
// Store in b the value of:
     "no int in 2..n-1 divides n
b= true;
// invariant: b = no int in 2..k-1 divides n
for (int k= 2; k < n;  k= k +1) {
      // Process k;
      if (n%k == 0) b= false;
}
// b = "no int in 2..n-1 divides n"
```

Command to do something and

equivalent postcondition

What is the invariant? 1 2 3 … k-1 k k+1 … n

Finding an invariant

```
// set x to no. of adjacent equal pairs in s[0..s.length()-1]
```

for s = 'ebeee', x = 2.

```
// invariant:
for (int k= 0; k < s.length();  k= k +1) {
        Process k;
}
// x = no. of adjacent equal pairs in s[0..s.length()-1]
```

Command to do something and equivalent post-condition

k: next integer to process. Which ones have been processed?

A. 0..k C. a..k
B. 0..k–1 D. a..k–1

What is the invariant?

A. x = no. adj. equal pairs in s[1..k]
B. x = no. adj. equal pairs in s[0..k]
C. x = no. adj. equal pairs in s[1..k–1]
D. x = no. adj. equal pairs in s[0..k–1]

1. What is the invariant?

Being careful

```
// { String s has at least 1 char }
// Set c to largest char in String s

// inv: c is largest char in s[0..k–1]

for (int k= ;  k < s.length();  k= k + 1) {
      // Process k;

}
// c = largest char in s[0..s.length()–1]
```

Command

postcondition

2. How do we initialize c and k?

A. k= 0; c= s.charAt[0];
B. k= 1; c= s.charAt[0];
C. k= 1; c= s.charAt[1];
D. k= 0; c= s.charAt[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. Therefore, start with k = 1.