CS1110 21 October 2008

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

Announcements

1. Prelim 2 next Tuesday evening, 7:30PM, Uris Auditorium.
   Yes, for-loops are not on this prelim.

2. Please complete an online questionnaire concerning your TA.

http://www.engineering.cornell.edu/TAEval/menu.cfm

This is a midterm evaluation. It is important, because your constructive comments are used to help the TA improve, which may help you in this course.

You will probably receive an email about this. Please complete the survey this week!

2

Assertion: true-false statement placed in a program to assert that it is true at that place.

```
x = sum of 1..n
```

x	?	n	1
x	?	n	2
x	?	n	0

3

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

precondition

```
// x = sum of 1..n-1
x= x + n;
n= n + 1;
// x = sum of 1..n-1
```

postcondition

x	6	n	4

n n
1 2 3 4 5 6 7 8 9 10 11

x contains sum of these

x	15

4

Solving a problem

precondition

```
// x = sum of 1..n
x= x + n+1;
n= n + 1;
// x = sum of 1..n
```

postcondition

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

5

Solving a problem

precondition

```
// x = sum of 1..n-1
x= x + n;
n= n + 1;
// x = sum of 1..n-1
```

postcondition

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

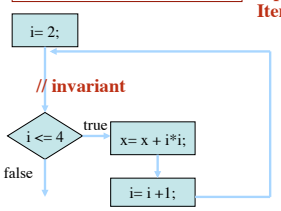
6

Execution of the for-loop

The for-loop:

```
for (int i= 2; i <= 4; i= i +1) {
    x= x + i*i;
}
```

loop counter: i
initialization: **int** i= 2;
loop condition: i <= 4;
increment: i= i + 1
repetend or body: { x= x + i; }
Iteration: 1 execution of repetend



To execute the for-loop.

1. Execute initialization.
2. If loop condition is false, terminate execution.
3. Execute the repetend.
4. Execute the increment and repeat from step 2.

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


```
// Process integers in a..b
// invariant: 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

a a+1 a+2 … k-1 k k+1 … b

Iteration: 1 execution of repetend
invariant: unchanging

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: something that is true before and after each iteration (execution of the repetend).

```
// Store in double variable v the sum
//   1/1 + 1/2 + 1/3 + 1/4 + 1/5 + … + 1/n
v= 0;
// invariant: v = 1/1 + 1/2 + … + 1/(k–1)
for (int k= 1; k <= n;  k= k +1) {
        Process k
}
// v =1/1 + 1/2 + … + 1/n
```

Command to do something and

equivalent postcondition

v = sum of 1/i for i in range 1..n

v = sum of 1/i for i in range 1..k-1

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

Find invariant: true before and after each iteration

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

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

for s = 'ebeee', x = 2.

k: next integer to process. Which ones have been processed?
A. 0..k C. a..k
B. 0..k–1 D. a..k–1
E. None of these

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]
E. None of these

Being careful

1. What is the invariant?

```
// { 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.