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
Tin a dot in place
Animosity
Is no amity
Debit card
Bad credit
Desperation
A rope ends it
Dominoe
Dirty noon
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

Debit Card
Bad Credit
Desperation
A rope ends it

Dormitory
Dirty room
Funeral
Real fun

Schoolmaster
The classroom
Slot machines
Cash lost in ’em

Circumstantial evidence
Can ruin a selected victim

Parishioners
I hire parsons
The earthquakes
That queen shake

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

// x is the sum of 1..n-1
x = x + n;
// x is the sum of 1..n
x = x + n;
// x is the sum of 1..n+1

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

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

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

Solving a problem

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

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

Today’s terminology:
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
And we give you a methodology for developing for-loops.
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).

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

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

The loop processes the range 2..4

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

Finding an invariant

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

What is the invariant?

A. 1  2  3  …  k-1  k  k+1 … n

```
// 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]
```

Finding an invariant

```
// Find c is largest char in s[0..k-1]
for (int k= 0;  k < s.length();  k= k + 1) {
    // Process k;
}
```

```
// c is largest char in s[0..s.length()-1]
```

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= 0;  k < s.length();  k= k + 1) {
    // Process k;
}
```

```
// c is largest char in s[0..s.length()-1]
```

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.