CS100J Lab 06. Loops Fall 2003

Name _____ Section time _____ Section instructor _____

This lab is purely a pencil-and-paper job. It gives you practice with loops, assertions, and understanding loops in terms of a precondition, postcondition, and loop invariant. The lab will being with a presentation by your lab instructor. Then, you will do some "simple" simple exercises. If you come across something you don't understand when doing the exercises, then ask a neighbor, a consultant, or your TA immediately.

Basics:

The while-loop has the form (syntax):

while (<*condition*>) <*repetend*>

where the <*condition*> is a boolean expression and the <*repetend*> is any statement.

The while loop is executed as follows: do the following repeatedly, until it can't be done because the <*condition*> is false:

Evaluate the <*condition*>, find it to be **true**, and execute the <*repetend*>.

The first execution of the repetend is iteration 0, the second is iteration 1, and so on.

Here is an example of a loop, which prints the squares of the numbers in the range 3..10:

```
int k= 3;
// { invariant: the squares of integers in the range 3..k-1 have been printed }
while (i != 11) {
     System.out.println(k * k);
     k= k + 1;
}
// { postcondition: the squares of integers in the range 3..11-1 have been printed }
```

Assertions

Remember that an assertion is a true-false statement about progam variables. It may be true or false, depending on the values of the variables. We place assertions in a program to "assert" they are true where we put them. The invariant of a loop is an assertion, and it is expected to be true before and after each iteration.

Four loop questions

In understanding a loop (and also in developing a loop), we ask four loopy questions:

1. How does it start? Initialization of variables has to make the invariant true --has to truthify it.
In the above example, the initialization sets k to 3. If we replace k by 3 in the invariant, we get

the squares of integers in the range `3..3-1` have been printed

or

the squares of integers in the range `3..2` have been printed

which is true because the range `3..2` is empty and no squares have been printed.

2. When does it stop? When it stops, the loop condition is false and the invariant is true. From these two together, we should be able to see that the postcondition is true.
In the above example, the loop stops when k = 11. If we replace k in the invariant by 11, we get

the squares of integers in the range `3..11-1` have been printed

This is the postcondition, so the loop terminates at the right time.

3. How does it make progress? Execution of the loop has to terminate by making the loop condition false.
In the above example, `k` starts out small (`3`) and it has to get up to `11`. Progress is made in the repetend by adding `1` to `k`.

4. How does it keep the invariant true? The loop invariant is supposed to be true before and after each iteration. That means that execution of the repetend has to keep it true.
In the above example, the invariant says that the squares of integers in the range `3..k-1` have been been printed. Therefore, the next integer to print is `k*k`, and that's what the repetend prints.

Exercises

We now give you some exercises. They are questions that you might have to ask when developing a loop, given a postcondition and invariant. However, we don't work with one complete loop with each question. Instead, we organize them in terms of the four loopy questions, giving you lots of practice with each loopy question in turn.

1. How does it start? Each case below consists of a relation and (perhaps) values for some of the variables used in the relation. Write assignments to the other variables that make the relation true.

Relation	Known variable	Assign to	Assignment(s)
(a) x * y = 5 * 4	x is 5	y	
(b) x * y = a * b	x is a	y	
(c) x = sum of 1..h	x is 1	h	
(d) x = sum of 1..h	h is 2	x	
(e) x = sum of 1..h	h is 1	x	
(f) x = sum of 1..h	h is 0	x	
(g) x = sum of h..10	h is 10	x	
(h) x = sum of h..10	h is 9	x	
(i) x = sum of h..10	x is 10	h	
(j) z + x * y = a * b	z is 0	x, y	
(k) z * xy = ab	z is 1	x, y	
(l) sum of h..n = sum h..k		k	

2. When does it stop? Each case below consists of an invariant and a postcondition. It is known that the invariant is true. What extra condition is needed to know that the postcondition is true?

invariant	postcondition	?
(a) x is sum of 1..k	x is sum of 1..10	
(b) x is sum of 1..k	x is sum of 1..n	
(c) x is sum of 0..k – 1	x is sum of 0..10	
(d) x is sum of 1..k – 1	x is sum of 1..n	
(e) x is sum of 0..k – 1	x is sum of 0..n – 1	
(f) x is sum of h..k	x is sum of 1..k	
(g) x is sum of h..k – 1	x is sum of 1..k – 1	
(h) m is the average of h..k – 1	m is the average of 1..k – 1	
(i) m is the average of 1..k – 1	m is the average of 1..n – 1	
(j) z + x * y = a * b	x * y = a * b	
(k) z + x * y = a * b	z = a * b	
(l) h..k – 1 has been processed	1..k – 1 has been processed	

3. How does it make progress? Each case below consists of an initial value for a variable and a final value. Write down a simple assignment that gets the variable closer to its final value.

initial value	final value	How to make progress
(a) h is 0	h is 10	
(b) h is 10	h is 0	
(c) h is n (where n > 0)	h is 0	
(d) h is n (where n < 0)	h is 0	

4. How does it fix the invariant? Each case below contains a relation that is assumed to be true and a statement that changes a variable. Write down a statement to execute **before** the given statement so that after both are executed, the relation is still true.

relation	statement	what to do before the statement
(a) s is the sum of 1..h	h= h + 1;	
(b) s is the sum of 1..(h – 1)	h= h + 1;	
(c) s is the sum of k..n	k= k – 1;	
(d) s is the sum of (k + 1)..n	k= k – 1;	
(e) s is the sum of 1..h	h= h – 1;	
(f) z + x * y = 100	y= y – 1;	
(g) z + x * y = 100 and y is even and y > 0	y= y / 2;	