CS100J Spring 2005 Exercises on loops

Many of these exercises ask you to write a loop (with initialization), given the task to be performed, a postcondition, and loop invariant. Develop the loop using the four loopy questions. This allows you to separate your concerns. For example, when writing the initialization, you don't worry about the loop condition or repetend, you just ask yourself what needs to be done to truthify the invariant. When you are finished writing the loop, test it in your IDE! That is the only way to be sure you did it properly.

If a question asks you to write a loop to calculate a value but does not express the postcondition as a true-false statement, then you should first write the postcondition and then write a suitable invariant (we won't ask you to write a suitable invariant on a test). For example, question 17 is to:

> Write a loop to count how many times the vowel "a" occurs in a string s.

So, you first write the postcondition:

> x is the number of times "a" occurs in s (for example)

and then write the invariant:

> x is the number of times "a" occurs in s[0..k-1] (for example)

Then, write the loop, using the four loopy questions.

Do not write return statements in the bodies of your loops. You may (should) write a method in Java to test your answers, but the purpose of these exercises is *not* to write methods but to write sequences of statements to truthify some postconditions.

E1. Write four loops (with initialization) to store in x the product of the integers in the range 2..10. The postcondition R is: x is the product of 2..10.

(a) Use this invariant P1, which was created by replacing constant 10 in R by k:

> P1: $2 \le k \le 10$ and x is the product of 2..k

(b) Use this invariant P2, which was created by replacing constant 10 in R by k – 1:

> P2: $2 \le k \le 11$ and x is the product of 2..(k – 1)

(c) Use this invariant P3, which was created by replacing constant 2 in R by k:

> P3: $2 \le k \le 10$ and x is the product of k..10

(d) Use this invariant P4, which was created by replacing constant 2 in R by k + 1:

> P4: $1 \le k \le 10$ and x is the product of (k + 1)..10

E2. Write four loops (with initialization) to determine whether an integer n is divisible by an integer in the range first..last, where first ≤ last. The answer is stored in a boolean variable b: the postcondition R is:

> R: b = "n is divisible by an integer in first..last"

(a) Use this invariant P1, which was created by replacing last in R by k:

> P1: b = "n is divisible by an integer in first..k"

(b) Use this invariant P2, which was created by replacing constant last in R by k – 1:

> P2: b = "n is divisible by an integer in first..(k – 1)"

(c) Use this invariant P3, which was created by replacing constant first in R by k:

> P3: b = "n is divisible by an integer in k..last"

(d) Use this invariant P4, which was created by replacing constant 2 in R by k + 1:

> P4: b = "n is divisible by an integer in k+1..last"

E3. Given is n > 0. Write a loop (with initialization) to store in k the largest power of 2 that is at most n. Note that 2**0 = 1. The obvious way to calculate k is to successively set k to 1, 2, 4, 8, ... until the right

power of 2 is reached. Use the postcondition R and invariant P shown below. Note how P is R with the last constraint n < 2**(k+1) removed and with the addition of variable b You may not use the notation ** or a Math function for exponentiation.

R: $1 \le 2^{**}k \le n < 2^{**}(k+1)$
P: $1 \le 2^{**}k \le n$ and $b = 2^{**}k$

E4. Write a loop to calculate the quotient q and remainder r when x (≥ 0) is divided by y (> 0), using just addition and subtraction (no multiplication or division). The four variables are related by this formula:

$x / y = q + r / y$ where $0 \le r < y$

i.e. $x = y * q + r$ where $0 \le r < y$

Use the loop invariant P:

P: $x = y * q + r$* and $0 \le r$

which arises from the formula by deleting the constraint r < y.

E5. Given is x > 0 and y > 0, both integers. Find the greatest common divisor of x and y, written as x gcd y. This is the largest integer that divides both. Use these properties of gcd:

x gcd y = (x-y) gcd y
x gcd y = x gcd (y-x)
x gcd x = x

Use two fresh variables b and c the following postcondition and invariant:

R: b = x gcd y
P: b gcd c = x gcd y

E6. Write a program segment to delete all the vowels in String t. Here is the outline:

Answer the four loopy questions to develop the loop using the following invariant and postcondition.

invariant: s[0..(k – 1)] contains no vowels
postcondition R: s[0..(s.length() – 1)] contains no vowels

Be wary of your increment: you can make progress in two different ways.

E7. Deoxyribonucleic acid, or DNA for short, is the building block of all life. Each strand of DNA consists of two strings of bases twisted together to form a double helix. There are 4 bases, which are represented by the letters G, A, T and C. In a double helix, the letters A and T bond together, as do the letters C and G. The two sequences in a helix, then, are complements of each other. For example, these two sequences are complements of each other:

sequence 1: ACGTTAC
sequence 2: TGCAATG

Notice how the A's and T's line up with each other, as do the C's and G's. Write a loop to determine if two Strings s1 and s2 representing DNA sequences are complements of each other. What do you need to assume about the lengths of those Strings?

E8. Write a loop to produce the DNA complement of a String s.

E9. The Fibonacci numbers are the numbers 0, 1, 1, 2, 3, 5, 8, …. Each number is the sum of the previous two. Here is a recurrence relation that describes the sequence:

$f_0 = 0$
$f_1 = 1$
$f_n = f_{n-1} + f_{n-2}$ for $n > 1$

Write a code fragment that finds Fibonacci number n, where n > 1. Use this invariant:

invariant: a = f_i and b = f_{i-1}

postcondition: i = n (and, therefore, a = f_n)

E10. Compound interest on an account is computed as follows: if an account has balance balance, and the annual interest rate is rate, then the next year's balance is this:

balance + balance * rate

Suppose variables balance, rate, and n (>= 0) contain values. Write a program segment that changes balance to the balance in the account after n years.

E11. Write a loop to count how many pairs of adjacent equal characters are in a string s. The string "bbbccd" contains 3 pairs of adjacent equal values.

E12. Write a loop that finds the length of the prefix of string s that is in descending order. Example, for s = "dcbaz", the answer is 4, since the string "dcba" is in descending order but "dcbaz" is not. For s = "", the answer is 0. For s = "z", the answer is 1. For s = "za", the answer is 2.

E13. Write a loop that finds the number of blanks at the end of string s.

E14. Write a loop that finds out whether string s is a palindrome. s is a palindrome if it reads the same backwards and forwards. For example, these are palindromes: "", "b", "bcb", "bccb". These are not palindromes: "bbbc", "bc". Use this postcondition and invariant for the loop:

invariant: s[0..h-1] is the reverse of s[k+1..s.length()-1]

postcondition: invariant and
either h >= k or s[h] != s[k]

Thus, the loop iteratively looks at the beginning characters s[0], s[1], ... and compares them to the end characters s[s.length()-1], s[s.length()-2], ...

E15. Write a loop and other statements that tells whether every character in string s has the same character next to it. For example, for the strings "", "aa", "bbbdd", "eeff66gg", the answer is yes, but for the string "aabbc" and abcbs" the answer is no. Your algorithm may include conditional statements and assignments, besides the loop. Please try to write a postcondition and invariant for your loop before writing the loop.

A simple way to do this exercise is to check each character s[k] in turn and, within the repetend, to check whether that s[k] is equal to either the preceding or the following character. It is good if your loop terminates as soon as it is detected that the answer is false.

E16. Write a loop that stores true in b if the characters in positions 1, 2, 4, 8, 16, ... of a string s are digits and false otherwise. For example, for s = "0123a5678", b is false, while for s = "1111111111", b is true. Please try to write a postcondition and invariant for your loop before writing the loop.

E17. Given are two strings s and t. Write a loop (and other statements) that determines whether they are equal. You could, of course, use the expression s.equals(t), but that is not a loop. The loop should terminate as soon as it is discovered that s and t are not equal.

E18. Think of a string of characters as consisting of pairs of equal adjacent characters, called twins. For example, in "aabbcc" we have three pairs of twins while in "aaaaab" the fifth "a" is missing its twin and the b is missing its twin. Write a loop that, given a string s, creates a string t that is the same as s except that characters that don't have twins in s have twins in t.

Here's another way to think about it. If in s a run of equal characters has an odd length, then add one more of those characters in t. Examples:

s: "a"	t: "aa"
s: "aabcc"	t: "aabbcc"
s: "aaabbccccc"	t: "aaaabbcccccc"
s: "aba"	t: "aabbaa"

You can use this invariant:

inv: t is s[0..k-1] but with twins added and if s[k-1] has a twin, it is s[k-2]