CS 1110, LAB 9: LOOP EXERCISES

Name: ______________________________ **Net-ID**: ______________

There is an online version of these instructions at

http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab09.php

You may wish to use that version of the instructions.

The purpose of this lab is to give you some practice with assertions and with loops that process a range of integers. It is a mixture of coding exercises and answers to be written on paper.

Requirements For This Lab. The very first thing that you should do in this lab is to download the file `Loops.java` from the course web page:

http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab09/Loops.java

You will notice that this file has several (static) function stubs. Part of your lab will be to complete these function stubs. We expect you to test that your code is correct, but we will **not ask you to submit a JUnit test**. You are all experienced programmers now, and should understand the value of testing without having to turn in your tests each time.

For this lab you will show your instructor the contents of `Loops.java` and what you have written on this sheet. As always, you should try to finish the lab during your section. However, if you do not finish during section, you have **until the beginning of lab next week to finish it**. You should always do your best to finish during lab hours; remember that labs are graded on effort, not correctness.

Written Exercises

Questions on Ranges. How many values are in the following ranges? The last one requires a formula in terms of `h` and `k`. Remember that in the notation `h..k`, we require `k` $\geq$ `h`-1. For example, 5..4 is OK but 5..3 is not allowed.

Given Range	Contents	Given Range	Contents
5..7		h..h+1	
5..6		h..h	
5..5		h..h−1	
5..4		h..k	
4..4		h−1..$h+1$	

Assigning to Range Variables. Each line below asks you to write an assignment. We have done the first one for you to give you an idea of what we are looking for.

Range	Want	Assignment Statement
`h..k`	Assign to `k` so that the range has 1 element	`k = h;`
`h..k`	Assign to `h` so that the range has 1 element	
`h..k`	Assign to `k` so that the range has 0 elements	
`h..k`	Assign to `h` so that the range has 0 elements	
`0..n−1`	Assign to `n` so that the range has 1 element	
`0..n−1`	Assign to `n` so that the range has 0 elements	
`h−1..10`	Assign to `h` so that the range has 1 element	
`h+1..10`	Assign to `h` to that the range has 0 elements	

Completing Assertions. Each line below contains an assertion P that is guaranteed to be true. Each line also contains an assertion R, which we would like to be true. In the righthand column, put a boolean expression that, when true, allows us to conclude that R is true. We have filled in the first one for you.

Know P	**Want R**	**Additional Info Needed**
x is the sum of `1..n`	x is the sum of `1..100`	`n == 100`
x is the sum of `1..(n−1)`	x is the sum of `1..100`	
x is smallest char in `s[0..k−1]`	x is smallest char in the range `s[0..s.length()−1]`	
x is no. of blanks in `s[0..k-1]`	x is no. of blanks in `s[0..]`	
x is the smallest char in `s[h..]`	x is the smallest char in `s[0..]`	
x is the product of `k..n`	x is the product of `1..n`	
b = “nothing in `h..k` divides x”	b = “nothing in `m..k` divides x”	

Preserving Invariants. Below is a precondition P, an assignment to a variable, and the same assertion P as a postcondition. At the place indicated, place a statement so that if P is true initially, it will be true afterward (as indicated). The statement can be in English, if you are not sure how to write it in Java, but make it a command to do something. In the exercises below, `v` is a Vector¡Integer¿ (so you can assume that its elements are of class Integer).

```
(a) // {P: x is the sum of 1..n}
    // Put a statement here:


    n= n+1;
    // { P: x is the sum of 1..n }
```

```
(b) // { P: x is the sum of h..100 }
    // Put a statement here:


    h= h - 1;
    // { P: x is the sum of h..100 }
```

```
(c) // {P: x is the minimum of v[0..k-1] }
    // Put a statement here:


    k= k+1;
    // {P: x is the minimum of v[0..k-1] }
```

```
(d) // { P: x is the minimum of v[h..100] }
    // Put a statement here:


    h= h - 1;
    // { P: x is the minimum of v[h..100] }
```

Coding For-Loops

Start a new folder and download `Loops.java` into this folder. This class contains two partially completed functions and two functions stubbed in just with return statements so that they compile. You are to complete the functions in this class. Each must contain a loop. Write one at a time, implementing the specification that we give you.

Make sure each function is correct before proceeding to the next one. Do this by writing suitable calls in DrJava's Interactions pane or making up a Junit test class (though we do not ask that you turn in these tests). Use enough different test cases so that you really are sure that the function is correct. If the function uses a String value, make sure that it works on an empty string (one whose length is 0). File `Loops.java` contains additional comments.

You may not finish all four functions during the lab. Near the end of the lab, show an instructor or consultant what you have done. If necessary, complete the lab during the next week and show what you have done to the instructor or consultant then.

Below, we show you on paper the functions you will be writing. In addition, we ask several questions for you to write on this piece of paper.

Function 1. Complete the following function by filling in the underlined parts. Test your function to make sure it is right.

```
/** Yields:  "p is a prime."  i.e. p is at least 2 and
 *  is divisible by only 1 and itself */
public static boolean isPrime(int p) {
    if (p < 2) {
        return ______________;
    }
    if (p == 2) {
        return ______________;
    }
    // Return false if some integer in 2..p-1 divides p
    // inv:  p is not divisible by integers in 2..k-1
    for (int k = _____; k != p; k= k+1) {
        if (_____________________) {
            return false;
        }
    }
    // no integer in 2..p-1 divides p
    return ______________;
}
```

Function 2. Type char is a primitive type (see pp. 224-226 of the class text). Characters are represented by integers (called ASCII codes), and the cast `(int)c` yields the integer that represents `c`. So, some characters are represented by primes and others are not. Complete the function below.

```
/** Yields:  a String that contains each capital letter (in 'A'..'Z')
 *  whose representation is prime */
public static String primeChars() {
    String s = ______________;
    // inv:  s contains each capital in 'A'..c-1 whose representation is prime
    for (char c = _____; c <= _____; c= (char)(c+1)) {
        if ( ______________________ ) {
            s = s + ______________;
        }
    }
    // s contains each capital in 'A'..'Z' whose rep is a prime
    return s;
}
```

Questions: How can you check out the function? How do you know that it works? Explain your testing strategy in **comments in your code**.

Function 3. Write a function with the following specification.

```
/** Yields: number of times character c appears in String s */
public static int noOfTimes(char c, String s)
```

For example, since 'e' occurs twice in "where", the following statement stores 2 in x:

```
x = noOfTimes('e', "where");
```

Questions: What range of integers does the function process? What is the invariant for your loop? The answers to both should be clear from your code, so **your code should contain the invariant as a comment**.

Function 4. Write a function with the following specification.

```
/** Yields: number of times the characters in s1 appear in s2 */
public static int noOfTimes(String s1, String s2)
```

This function could be used as follows. The call

```
noOfTimes("aeiou", "Where is it?")
```

yields the number of times a vowel occurs in Where is it?. Note that duplicates in `s1` are to be counted as many times as they occur in s1. For example,

```
noOfTimes("aaa", "ac")
```

returns 3.

The body of this function should contain a for-loop whose repetend calls the previous function that you wrote (the other function noOfTimes) once for each character in `s1`.

Questions: What range of integers does the function process? What is the invariant for your loop? The answers to both should be clear from your code, so **your code should contain the invariant as a comment**.