CS 1110, LAB 10: LOOP EXERCISES

Name: _____ **Net-ID**: _____

There is an online version of these instructions at

http://www.cs.cornell.edu/courses/cs1110/2012fa/labs/lab10

You may wish to use that version of the instructions.

The purpose of this lab is to give you some practice with assertions (e.g. preconditions, postconditions, and invariants) 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 `lab10.py` from the course web page:

http://www.cs.cornell.edu/courses/cs1110/2012fa/labs/lab10/lab10.py

You will notice that this file has several 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 unit 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 `lab10.py` 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 element of the segment `s[0..k`−1`]`	x is smallest element of the segment `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 element of the segment `s[h..]`	x is the smallest element of the segment `s[0..]`	
x is the product of `k..n`	x is the product of 1`..n`	
b is True if nothing in `h..k` divides x; False otherwise	b is True if nothing in `m..k` divides x; False otherwise	

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 Python, but make it a command to do something. In the exercises below, `v` is a list of `int`s.

(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 `lab10.py` into this folder. This class contains two partially completed functions and two functions stubbed in just with return statements so that you can call them safely. You are to complete **the first three of the functions in this module**. The last function is optional.

Each implementation must contain a while-loop. Write one at a time, implementing the specification that we give you. When a loop invariant has not been given (such as with the last function), write your own.

Make sure each function is correct before proceeding to the next one. Do this by writing suitable calls in the interactive prompt or making a unit test (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 `lab10.py` contains additional comments.

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