CS 1110, LAB 12: LOOP INVARIANTS

http://www.cs.cornell.edu/courses/cs1110/2017fa/labs/lab12/

First Name: _____ **Last Name**: _____ **NetID**: _____

Because of the way the semester falls, this is the second to last lab. We will have one more lab during the last week. However, remember that you are allowed to skip two labs and still get full credit. So if you have done all of the labs so far, technically you can skip this one.

With that said, these last two labs are important. They help you construct algorithms with while-loops. This is an important part of the final exam (and the only new topic not on Prelim 1 or 2). In fact, the next lab is even more important for the final than this one. Therefore, even if you do not turn them in for credit, we recommend that you do these two labs to give you more practice before the exam. In addition, we will post the last lab next week even though there is no lab the week of Thanksgiving.

Lab Materials. We have created several Python files for this lab. You can download all of them from the Labs section of the course web page.

http://www.cs.cornell.edu/courses/cs1110/2017fa/labs

When you are done, you should have the following two files.

- `lab12.py` (the primary module for the lab)
- `test12.py` (a unit test for `lab12.py`)

Create a *new* directory on your hard drive and download all of the files into that directory. You can also get the files bundled in a single ZIP file called `lab12.zip` from the course web page.

Getting Credit for the Lab. Once again, you have a choice between getting credit through the online system or your instructor. The online lab is available at the web page

http://www.cs.cornell.edu/courses/cs1110/2017fa/labs/lab12/

As with the last lab, you should be modifying the file `lab12.py` directly, and not typing all of your answers into the online system directly.

Because of the holidays, there is no lab the week of Thanksgiving. Therefore, you have until **until the beginning of lab the last week of class to finish it**. You should always do your best to finish during lab hours. Remember that labs are graded on effort, not correctness.

Exercise 1: Warm-Up Exercises

Completing Assertions. Each line below contains an assertion P guaranteed to be true. It 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 R is true. We have filled in the first one for you.

Course authors: D. Gries, L. Lee, S. Marschner, W. White

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 the product of k..n	x is the product of 1..n	
x is smallest element of the segment s[0..k−1]	x is smallest element of the segment s[0..len(s)−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..]	
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. In each problem below, you are given a precondition P, an assignment to a variable, and P rewritten as a postcondition. Where indicated, write a statement so that if P is true initially, it will also be true afterward. The statement can be in English, but make it a command to do something. In all of these exercises, v is a list of ints.

Note: You may not use any function calls in your answer.

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


    n = n + 1
    # P: x is the sum of 1..n

(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]
```

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


    h = h - 1
    # P: x is the sum of h..100

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


x
    h = h - 1
    # P: x is the minimum of v[h..100]
```

Exercise 2: Functions With Loop Invariants

In the file lab12.py are the stubs for two functions: num_space_runs and split. In addition to the specification, the stub provides you with both an invariant and a post-condition. For each function you should

- Assign the variables initial values to satisfy the invariant.
- Write the body of the while-loop to make progress while satisfying the invariant.
- Return one of the variables as the final answer, according to the specification.

When you have done this, you have finished the lab.