CS 1110, LAB 12: LOOPS AND LOOP INVARIANTS

http://www.cs.cornell.edu/courses/cs1110/2017sp/labs/lab13.pdf

First Name: ____________________ **Last Name**: ____________________ **NetID**: ______________

Getting Credit: Deadline: **in the first 10 minutes of (your) lab next week (Tues May 9/Wed May 10)**. The checking-off procedure is the same as before.[1]

Exercise 1: Warm-Up Exercises

Completing Assertions; practice with integer-range notation. Each row below contains an assertion P and an assertion R that we would like to be true. In the right-hand column, write an assignment statement for variable `n` that allows us to conclude that R is true if P was true beforehand. We have filled in the first one for you.

Know P	**Want** R	**Assignment to `n` that makes** R **true**
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 `n..k`	x is the product of `1..k`	
x is smallest element of the segment `s[0..n−1]`	x is smallest element of the segment `s[0..len(s)−1]`	
x is no. of blanks in `s[0..n-1]`	x is no. of blanks in `s[0..]`	
x is the smallest element of the segment `s[n..]`	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 `n..k` divides x; False otherwise	

Preserving Invariants. In each of (a)-(d) below, you are given an assertion labeled `P` and an assignment statement. Where indicated, write code so that if the assertion `P` is true initially, it is *also* true after your code and the assignment statement. We have done the first one for you.

In all of these exercises, `v` is a list of `int`s.

Course authors: E. Andersen, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White

[1] In case you've forgotten, here's a reminder: Show this handout and/or your code to a staff member either (a) during your lab 13 session, (b) in non-professorial consulting hours listed at http://www.cs.cornell.edu/courses/cs1110/2017sp/about/staff.php up to the day **before** your next scheduled lab section, or (c) in the first 10 minutes of (your) next scheduled lab (Tues May 9/Wed May 10). Beyond that time, the staff have been instructed not to give you credit.
Labs are graded on effort, not correctness. We just want to see that you tried all the exercises, and to clarify any misunderstandings or questions you have.

(a)
```
# P: x is the sum of 1..n
# STUDENTS: Put 1-3 lines here:

x = x + (n+1)

n = n + 1
# x is (once again) the sum of 1..n
```

(b)
```
# P: x is the sum of h..100
# STUDENTS: Put 1-3 lines here:


h = h - 1
# x is (once again) the sum of h..100
```

(c)
```
# P: x is the minimum of v[0..k-1]
# STUDENTS: Put 1-3 lines here:



k = k + 1
# x is (again) the min of v[0..k-1]
```

(d)
```
# P: x is the minimum of v[h..100]
# STUDENTS: Put 1-3 lines here:



h = h - 1
# x is (again) the min of v[h..100]
```

Exercise 2: Functions With Loop Invariants

The following pages provides "stub" (skeletons) for several functions, all of which have the same specification, but are to be implemented based on different invariants. Complete **two of the three** skeletons with while-loops that constitute effective use of the given invariants.

If you want to code up and test your solutions, or see test cases for the function to be implemented, get the Lab 13 files packaged in a single zip file from the Labs section of the course web page, http://www.cs.cornell.edu/courses/cs1110/2017sp/labs .

```python
def num_space_runs1(s):
    """Returns: The number of runs of spaces in the string s.

    A run is a collection of adjacent spaces.  We need a non-space character
    in between to break up runs.

    Example: num_space_runs('  a  f   g    ') returns 4
             num_space_runs('a  f   g') returns 2
             num_space_runs('  a  bc   d') returns 3

    Parameter s: The string to parse
    Precondition: s is a nonempty string with letters and spaces"""

    # STUDENTS: The invariant for you to work with is:
    #     s[0..i-1] has n runs of spaces, AND:
    #     in_a_run is a boolean:
    #          True if i-1 is a valid index and s[i-1] is a space
    #          False otherwise
    #
    # In other words, s[i..len(s)-1] still needs to be checked;
    # and in_a_run tells us whether a new space would be part of an old run.


    # REPLACE THE FOLLOWING WITH CORRECT INITIALIZATION CODE:
    i = None
    n = None
    in_a_run = None

    # PUT YOUR WHILE LOOP HERE
    # Hint1: you only need to increment n when you find a space and you are
    # not currently in a run.
    # Hint2: you need to change in_a_run when:
    #   (a) you have found a space and you are not currently in a run, or
    #   (b) you found a non-space and you currently in a run
    # Hint3: don't forget to increment your loop variable, if you have one!















    # post: s[0..len(s)-1] contains n runs of spaces
    # PUT THE RETURN STATEMENT HERE
```

```
def num_space_runs2(s):
    """Same spec as above"""

    # invariant: s[0..i] contains n runs of spaces. So if i+1 is a legal index,
    # s[i+1] is the next thing to check, or the unknowns are s[i+1..len(s)-1].


    # WE ARE GIVING YOU THE FOLLOWING INITIALIZATION. DON'T CHANGE IT.
    i = 0
    if s[0] == ' ': # this initialization "peeks" at the data to see
                    # whether s[0] starts a run or not.
        n = 1
    else:
        n = 0

    # PUT YOUR WHILE LOOP HERE.
    # Hint: you only need to increment n when you have a space following a
    # non-space.











    # post: s[0..len(s)-1] contains n runs of spaces

    # PUT THE RETURN STATEMENT HERE
```

```
def num_space_runs3(s):
    """Same spec as above"""

    # The invariant for you to work with is:
    #     s[0..i] has n runs of spaces
    #
    # In other words, s[i+1..len(s)-1] still needs to be checked.

    # WE ARE GIVING YOU THE FOLLOWING INITIALIZATION. DON'T CHANGE IT.
    i = -1
    n = 0

    # PUT YOUR WHILE LOOP HERE
    # Hint: you only need to increment n when i is a valid index and s[i] is
    # not a space but s[i+1] is a space,
    # OR when i is -1 and the very first character in s is a space















    # post: s[0..len(s)-1] contains n runs of spaces
    # PUT THE RETURN STATEMENT HERE
```