CS 1110, LAB 12: SEQUENCE ALGORITHMS

http://www.cs.cornell.edu/courses/cs1110/2015fa/labs/lab12.pdf

First Name: ____________ **Last Name**: ____________ **NetID**: ____________

This last lab is extremely important. It helps you construct complex algorithms on sequences. This is a important part of the final exam (and the only new topic not on Prelim 1 or 2). Even if you have completed enough labs to "skip one," we recommend that you do not skip this lab.

This lab gives you some practice using invariants to implement algorithms on sequences, as covered in Lecture 24. If you are having difficulty with this part of the lab, you might want to review the additional reading for that lecture.

Getting Credit for the Lab. This is a *long* lab composed entirely of written exercises. While it is okay if you do not finish the lab during class time, remember that there is *no official lab next week*. We will hold lab on Tuesday as optional office hourse for students who want the help, but there is no lab on Wednesday. Therefore, this lab is not due until the final week of class. With that said, this is an important enough lab that we would prefer for you to do it before Thanksgiving.

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.

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 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. On the next page, 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 `int`s.

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

```
(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]
```

Drawing Horizontal Diagrams. Draw a diagram for list `b` that satisfies these conditions

```
b[0..i] >= 5, b[i+1..j] = 5, b[j+1..] <= 5
```

Exercise 2: Sequence Algorithms

When you write your algorithms, we want them to be as close to Python as possible, and not high-level "pseudo-code". However, if you want to swap two elements of an list, you may write

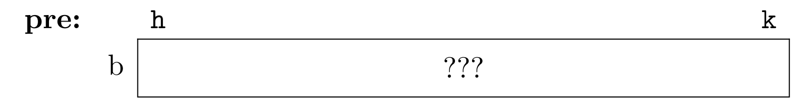
```
swap b[i] and b[j]
```

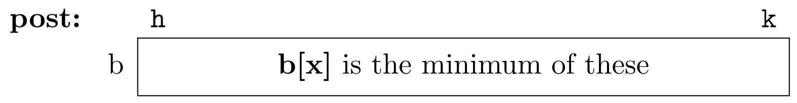
instead of the three assignments that perform the swap. This is the only shortcut that we allow.

Finding the Minimum of a List. The following statements are assertions in an algorithm to find the minimum element of a list list **b[h..k]**:

> **Precondition**: `h <= k < len(b)`
>
> **Postcondition**: **b[x]** is the minimum of **b[h..k]**

We represent each of these assertions as the pictures shown below.



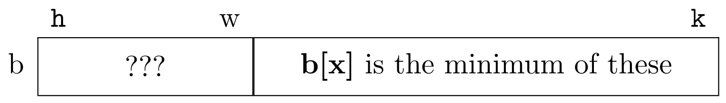


There any many, many different invariants we could construct that would be compatible with these assertions. Consider the following:

invariant: `b[x]` is the minimum of `b[w+1..k]`

Pictorially, we represent it as follows:

P4:



Write your loop for this invariant in the space below:

Partitioning on a Fixed Value. The algorithms below swap the values of list b and store a value in k to make the postcondition is true. List b is not sorted initially. The precondition and postcondition are as follows:

Precondition: `b[0..]` = ? (i.e. nothing is known about the values in b)

Postcondition: `b[0..k]` ≤ 6 and `b[k+1..]` > 6

Below are two different invariants. You should write a loop (with initialization) for each one. You are also to draw pictorial representation of the invariant in each case.

Invariant P1. Written in text form, this invariant is as follows:

P2: `b[0..k]` ≤ 6 and `b[t..]` > 6

Draw the pictorial representation of this invariant below.

Write your loop for this invariant in the space below:

Invariant P3. Written in text form, this invariant is as follows:

P3: `b[0..s-1]` ≤ 6 and `b[k+1..]` > 6

Draw the pictorial representation of this invariant below

Write your loop for this invariant in the space below: