

CS 1110, LAB 12: SEQUENCE ALGORITHMS

<http://www.cs.cornell.edu/courses/cs1110/2013fa/labs/lab12.pdf>

First Name: _____ Last Name: _____ NetID: _____

This last lab is extremely important. It helps you understand how to construct complex algorithms on sequences, such as we have done in class. This is a important part of the final exam. Therefore, even though this is the last week of class, this lab is not optional. You do not need to complete the lab in class section, but you do need to show it to an instructor or consultant to (eventually) check it off. See the instructions below on how to get credit for this lab.

Getting Credit for the Lab. There are no files to download for this lab. The purpose of this lab is to design loop algorithms, and you are to write the algorithms that you design (together with the diagrams for your pre and postconditions) on this sheet of paper. If you finish during class time, show this handout to your instructor. Your instructor will then swipe your ID card to record your success. You do not need to submit the handout.

This is a long handout, so you only need to finish **the first three exercises**. All other exercises are optional. If you do not finish during class, you need to show this lab to an instructor or consultant **before the final exam**. You can show it for credit during office hours or during consultant hours.

Note in our sample answers, that we do not draw all the pictures that are required. If an invariant is not drawn as a picture, you should draw it as a picture before continuing to work on the problem. If you need help, you should ask your TA or consultant.

You **do not need to implement these algorithms**. However, if you wish to test your programs in Python, you are welcome to do so. In this case, you may find it useful to print out the contents of the list after each time that you complete the algorithm.

Algorithm Shortcuts. In writing your algorithms in the space below, we want the algorithms 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 are permitted to write

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

instead of the the three assignments that perform the swap. This is the only algorithm shortcut that we are permitting.

EXERCISE 1: WARM-UP EXERCISES

Counting the values in $b[x..y-1]$. Without slicing the list b , write a Python expression that evaluates to the number of elements in the segment $b[x..y-1]$.

If you do not know what the formula is, first try to figure it out by looking at segments of size 1 and 2. Otherwise, ask your instructor and then memorize it. That formula is a useful tool. Knowing it will help you develop loops that deal with ranges of integers.

Drawing List Diagrams. Draw a list b that satisfies these conditions

$$b[0..i] \geq 5, b[i+1..j] = 5, b[j+1..] \leq 5$$

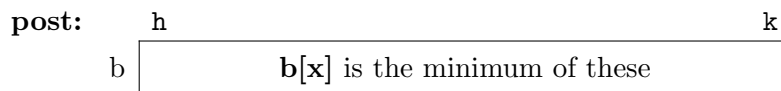
EXERCISE 2: FINDING THE MINIMUM OF THE LIST

The following are assertions for an algorithm to find the minimum of an list $b[h..k]$:

Precondition: $h \leq k < \text{len}(b)$

Postcondition: $b[x]$ is the minimum of $b[h..k]$

We represent each of these assertions as the following pictures.

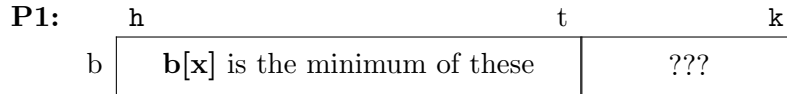


Given these assertions, we present four different loop invariants below. You are to write a loop (with initialization) for each one. The loop should be (mostly) Python code, though you are allowed to take the shortcut mentioned on the first page.

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

invariant: $b[x]$ is the minimum of $b[h..t]$

Pictorially, we represent it as follows:

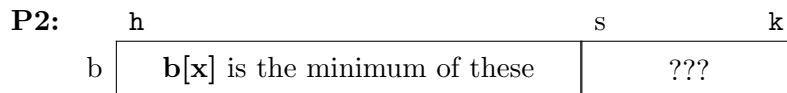


Write your loop for this invariant in the space below:

Invariant P2. Written in text form, this invariant is as follows:

invariant: $b[x]$ is the minimum of $b[h..s-1]$

Pictorially, we represent it as follows:

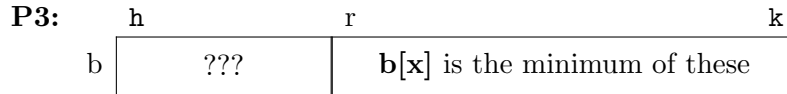


Write your loop for this invariant in the space below:

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

invariant: $b[x]$ is the minimum of $b[r..k]$

Pictorially, we represent it as follows:

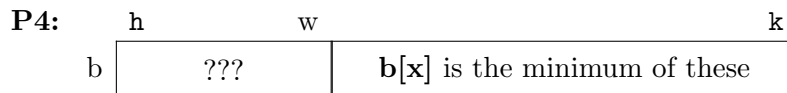


Write your loop for this invariant in the space below:

Invariant P4. Written in text form, this invariant is as follows:

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

Pictorially, we represent it as follows:



Write your loop for this invariant in the space below:

EXERCISE 3: PARTITIONING ON A FIXED VALUE

The algorithm below is 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: $b[0..k] \leq 6$ and $b[k+1..] > 6$

Below are three 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:

P1: $b[0..h] \leq 6$ and $b[k+1..] > 6$

Draw the pictorial representation of this invariant below.

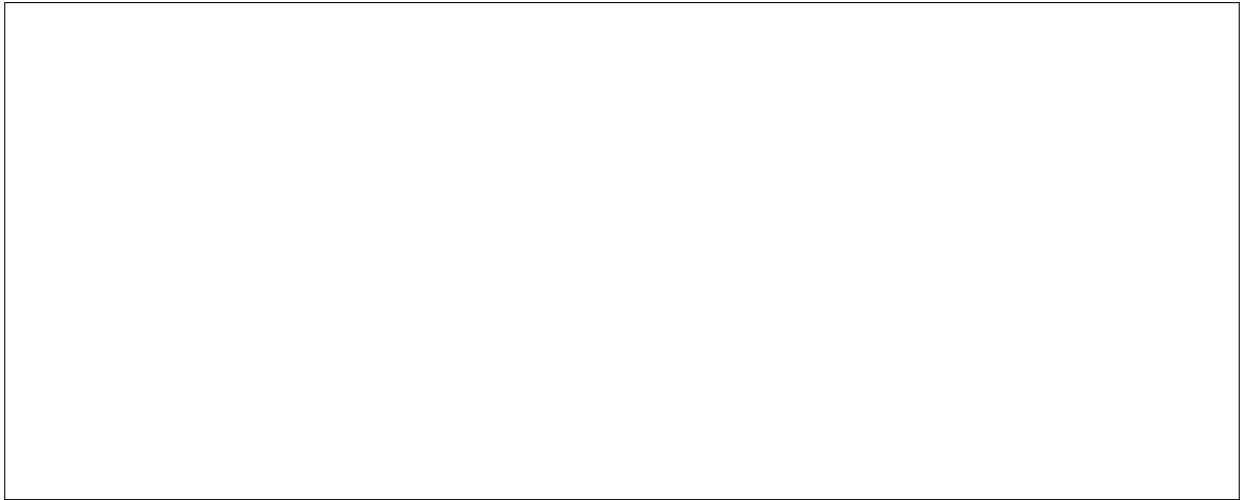
Write your loop for this invariant in the space below:

Invariant P2. Written in text form, this invariant is as follows:

P2: $b[0..k] \leq 6$ and $b[t..] > 6$

Draw the pictorial representation of this invariant below.

Write your loop for this invariant in the space below:



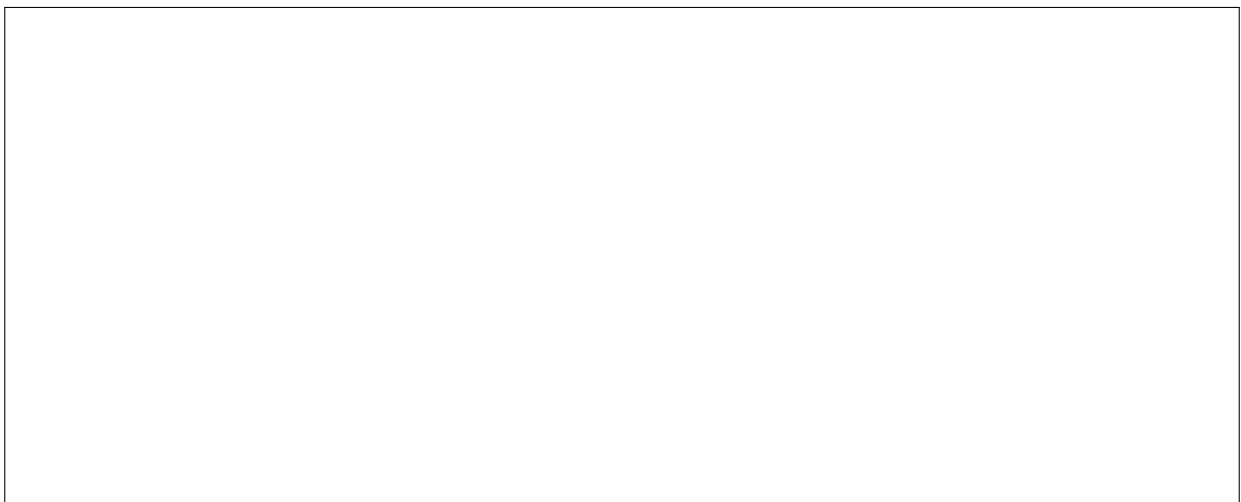
Invariant P3. Written in text form, this invariant is as follows:

$$\mathbf{P3: } b[0..s-1] \leq 6 \text{ and } b[k+1..] > 6$$

Draw the pictorial representation of this invariant below



Write your loop for this invariant in the space below:



EXERCISE 4: GENERAL PARTITIONING ALGORITHM (OPTIONAL)

This question is a generalization of the previous one. Again b is not necessarily sorted initially. Develop the partition algorithm (which uses only swap operations) to meet the assertions below:

Precondition: $b[h] = x$ for some x and $h \leq k < \text{len}(b)$
(this is so we can talk about $b[h]$; x is not a program variable.)

Postcondition: $b[h..j-1] \leq x = b[j] \leq b[j+1..k]$

Below are three 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:

P1: $b[h..j-1] \leq x = b[j] \leq b[t..k]$

Draw the pictorial representation of this invariant below

Write your loop for this invariant in the space below:

Invariant P2. Written in text form, this invariant is as follows:

P2: $b[h..j-1] \leq x = b[j] \leq b[q+1..k]$

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:

$$\mathbf{P3: } b[h..j-1] \leq x = b[j] \leq b[j+1..n-1]$$

Draw the pictorial representation of this invariant below.

Write your loop for this invariant in the space below:

EXERCISE 5: SELECTION SORT (OPTIONAL)

The last exercise is to write a selection sort, which sorts the contents of the list `b`. The postcondition for this problem is straightforward:

Postcondition: `b[0..len(b)-1]` is sorted (in ascending order)

We have provided several invariants for you below. Before you do each one, write the invariant as a picture. Then write the statement(s) you use to maintain the invariant in the body in English. You should state *what* it is you want to do, not *how* to do it. In particular, we do not want to see a nested loop (e.g. a loop within the body of your first loop).

Instead, you should reuse the results of one of the two previous exercises. Pretend that the algorithms from Exercise 2, 3, or 4 is inside a function (you can name the function whatever it is you want, so long as it is clear). Then use that function as a helper in this part of the lab.

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

P1: `b[0..k1]` is sorted and `b[0..k1] ≤ b[k..]`

Draw the pictorial representation of this invariant below.

Write your loop for this invariant in the space below:

Invariant P2. Written in text form, this invariant is as follows:

P2: `b[0..h]` is sorted and `b[0..h] ≤ b[h+1..]`

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: $b[s+1..len(b)-1]$ is sorted and $b[0..s] \leq b[s+1..]$

Draw the pictorial representation of this invariant below.

Write your loop for this invariant in the space below. Your code will require a helper function which is similar to, but not the same as one of the previous exercises. You do not have to implement the helper function so long as you clearly explain what it does.