CS 1110, LAB 11: LOOPS AND INVARIANTS

Name: ____________________ **Net-ID**: __________

There is an online version of these instructions at

http://www.cs.cornell.edu/courses/cs1110/2012sp/labs/lab11.php

You may wish to use that version of the instructions.

The purpose of this lab is to give you practice with developing loops from invariants. It also introduces some of the important array algorithms that you should know.

Note: You do not need to "memorize" the algorithms in this lab for the prelim on April 17th. But you will need to know them for the final. The practice this lab gives you should help you understand the use of loop invariants drawn as array diagrams.

Requirements For This Lab. There is no code to turn in 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. When you are done, you should show your instructor the contents of the sheet. If you finish the **first three exercises**, the instructor will mark you down has having completed the assignment.

If you do not finish that much, you should the first three exercises **by the beginning of lab next week** and show them to your instructor at that time. In any case, you should try to do all the exercises (including the remaining two) because that will help you understand and gain skill in solving such problems.

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 DrJava, you are welcome to do so. In this case, you may find it useful to print out the contents of an array. The static ethod java.util.Arrays.toString(Object[]) prints out the contents of its array parameter.

Algorithm Shortcuts. In writing your algorithms in the space below, we want the algorithms to be as close to Java as possible, and not high-level "pseudo-code". However, if you want to swap two elements of an array, you are permitted to write

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

instead of the the three assignments that perform the swap.

Exercise 1: Warm-Up Exercises

Counting the values in `b[x..y-1]`. You are to give a formula (in Java code) for the number of values in the segment `b[x..y-1]`, which you should write in the box below:

If you do not know what the formula is, first try to figure it our by looking at segments of size 1 and 2. Otherwise, ask you 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 Array Diagrams. Draw an array `b` that satisfies these conditions

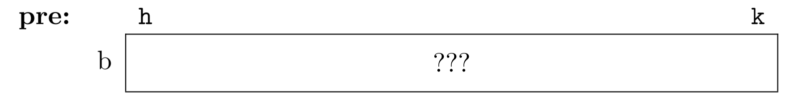
```
b[0..i] >= 5, b[i+1..j] = 5, b[j+1..] <= 5
```

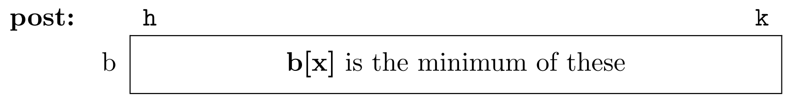
Exercise 2: Finding the Minimum of the Array

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

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

We represent these assertions as the following pictures (you may either use the text version or the pictures in your algoritms).





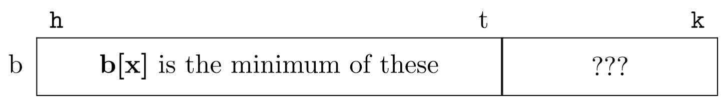
Given these assertions, we present four different loop invariants below. You are to write a loop (with initialization) for each one.

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:

P1:



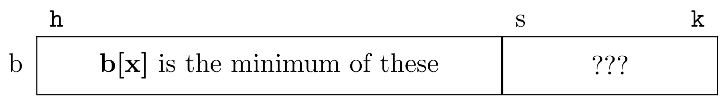
Write your loop for this invariant in the space below:

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:

P2:



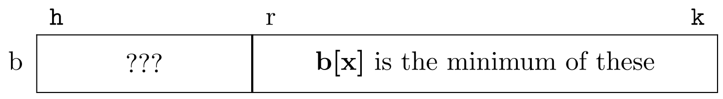
Write your loop for this invariant in the space below:

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:

P3:



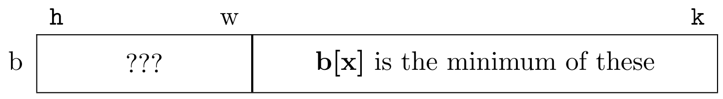
Write your loop for this invariant in the space below:

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:

P4:



Write your loop for this invariant in the space below:

Exercise 3: Partitioning on a Fixed Value

The purpose of the algorithms in this question is to swap the values of array b and to store a value in k so that the postcondition given below is true. Array b is not necessarily 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 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]` ≤ 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]` ≤ 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:

Exercise 4: General Partitioning Algorithm

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` $<$ `b.length`
(this is just so we can talk about b[h] initially; `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:

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

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

Postcondition: `b[0..b.length1` 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 repetend 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 repetend of your first loop). Instead, you should use one of the two methods you wrote above as a helper method.

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

P1: `b[0..k1]` is sorted and `b[0..k1]` $\leq$ `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]` $\leq$ `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..b.length-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: