Your five-digit number: _____

Problem 0 (2 points)

Part a

Put your secret number on each page. Also make sure you have provided the information requested on the first page.

Part b

Complete the final survey in the activities for August 12 and 13.

Problem 1 (3 points)

Suppose a value in a max heap were changed. To recreate the heap to reflect the modified value, would you bubble the value up, bubble it down, both, or neither? Briefly defend your answer.

Your five-digit number: _____

Problem 2 (3 points)

Consider an initially empty AVL tree. Give a sequence of insertions for which the *fourth* insertion produces a rotation, or explain why no such sequence exists.

Your five-digit number: _____

Problem 3 (12 points)

Given below is a solution to one of the homework exercises, namely a constructor that builds a binary tree out of the preorder and inorder traversals of the tree. (Documentation of List methods appears on the last page of this exam.)

```
        public BinaryTree (List preorder, List inorder) {
1          if (preorder.size ( ) == 0) {
2             myRoot = null;
3          } else {
4             myRoot = helper (preorder, inorder);
5          }
        }

        private TreeNode helper (List preorder, List inorder) {
6          Object rootItem = preorder.get (0);
7          TreeNode t = new TreeNode (rootItem);
8          if (inorder.size ( ) == 1) {
9             return t;
10         }
11         int pos = inorder.indexOf (rootItem);
12         if (pos > 0) { // left children exist
13            t.myLeft = helper (
14                     preorder.subList (1, pos+1),
15                     inorder.subList (0, pos) );
           }
16         if (pos < inorder.size ( ) - 1) {       // right children exist
17            t.myRight = helper (
18                     preorder.subList (pos+1, preorder.size ( )),
19                     inorder.subList (pos+1, inorder.size ( )) );
           }
20         return t;
        }
```

Part a

Draw the tree that results from supplying the arguments [A, A, B] and [A, A, B] to the constructor, and briefly justify your answer.

Problem 3, continued

Part b

The code on the previous page assumes that its arguments are legal, that is, that they actually represent the preorder and inorder traversals of some tree. Your task is to bulletproof the code, providing tests that detect invalid input and cause an IllegalArgumentException to be thrown with an informative error message. (A tree with duplicate elements, by the way, is legal.)

We don't want JUnit tests. Instead, we want you to add expressions of the following type to the code:

```
if ( _____ ) {
    throw new IllegalArgumentException (an appropriate message goes here);
}
```

In the table below, indicate what tests to add and where in the code above they should be made to ensure that the constructor never crashes on its own, but instead either builds the appropriate tree or throws IllegalArgumentException.

You will not be penalized for redundant tests. We think there are five places in the code that need bulletproofing.

Test location or line number	*Test to make at that location*

Your five-digit number: _____

Problem 4 (12 points)

This problem involves designing an efficient algorithm to arrange elements of one ArrayList, which we'll call toSort, in the same sequence as they appear in another, which we'll call allElements. For example, suppose allElements contains the strings

now	is	the	time	for	all	good	people

and toSort contains the strings

good	is	for	now

The algorithm would return an ArrayList containing the strings

now	is	for	good

The algorithm to be designed has two steps:

1. Preprocess allElements in such a way as to make step 2 as efficient as possible.
2. Rearrange the elements of toSort as just described.

Step 1 need not be efficient at all. There are no memory constraints on the algorithm.

Part a

Describe, in terms easily understandable by another CS 61BL student, a structure in which you would store the elements of allElements so you can rearrange the elements of toSort as quickly as possible. Name this structure processedElements. Don't worry about the time needed to build it, or how much memory it takes.

Problem 4, continued

Part b

Describe, in terms easily understandable by another CS 61BL student, a procedure for using processedElements to arrange the values in toSort. Assume that the values in toSort are all also in allElements, and that allElements is much bigger than toSort. Your algorithm should run as quickly as possible.

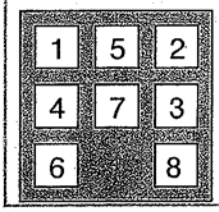
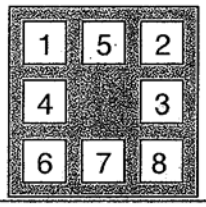
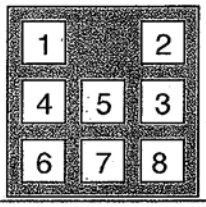
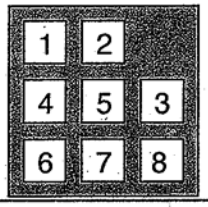
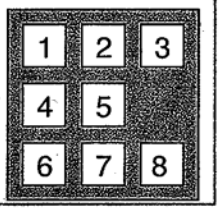
Part c

Provide as good an estimate as possible of the running time of your answer to part b, in terms of T, the number of objects in toSort, and A, the number of objects in allElements. Also explain how you determined your estimate.

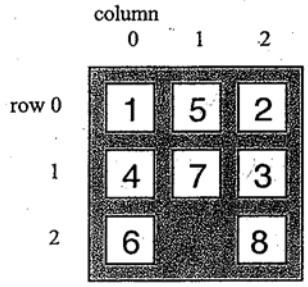
Your five-digit number: _____

Problem 5 (16 points)

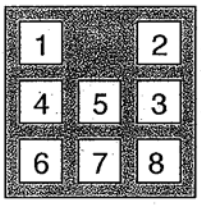
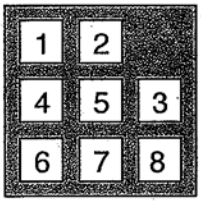
The "8 puzzle" uses eight square blocks numbered 1 to 8 in a three-by-three tray. The goal is to slide the blocks around in the tray until the blocks are in numerical order when read row by row. For example, the four moves below solve the problem for the given initial configuration.

initial configuration	*after moving 7 down*	*after moving 5 down*	*after moving 2 left*	*after moving 3 up*
				

A separate document accompanying this exam contains the framework of a program to solve this puzzle. Its primary class, named Tray, represents the tray and its blocks with an array of the integers 0 through 8; 0 represents the blank space in the tray (the place without a block). Rows and columns are numbered from the top left corner of the tray:



On the next page, supply a pseudocode body for the processLeftMove method, in detail sufficient for another CS 61BL student to translate it immediately into Java code (essentially one pseudocode statement for each Java statement in processLeftMove).

To clarify: a left move from the tray  results in the tray 

Problem 5, continued

```
// Possibly extend the fringe with the tray (if any) that results from
// moving a block left in the given current tray.
public void processLeftMove (Tray current) {
```

Your five-digit number: _____

List methods

E	`get(int index)` Returns the element at the specified position in this list.
int	`indexOf(Object o)` Returns the index in this list of the *first* occurrence of the specified element, or –1 if this list does not contain this element.
int	`size()` Returns the number of elements in this list.
List<E>	`subList(int fromIndex, int toIndex)` Returns a view of the portion of this list between the specified fromIndex, inclusive, and toIndex, exclusive.

HashSet methods

boolean	**`add`**`(E o)` Adds the specified element to this set if it is not already present.
boolean	**`contains`**`(Object o)` Returns `true` if this set contains the specified element.
boolean	**`isEmpty`**`()` Returns `true` if this set contains no elements.
boolean	**`remove`**`(Object o)` Removes the specified element from this set if it is present.
int	**`size`**`()` Returns the number of elements in this set (its cardinality).