CSE143 Summer 2008 Final Exam — Part B KEY
August 22, 2008

Name : _____

Section (eg. AA) : _____ TA : _____

This is an open-book/open-note exam. Space is provided for your answers. Use the backs of pages if necessary. The exam is divided into 4 questions with the following points:

Question	Points	Score
Tree Traversal	6	
Binary Search Tree	4	
Programming with Inheritance	20	
More Binary Trees	20	
Total:	50	

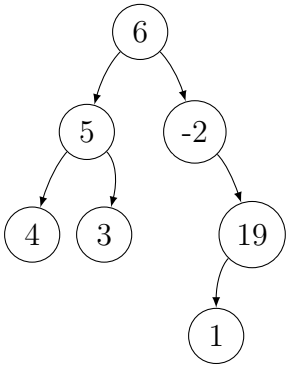
Do not begin work on this exam until instructed to do so. Any student who starts early or who continues to work after time is called will receive a 10 point penalty.

The exam is not, in general, graded on style and you do not need to include comments.

Please turn off an cell phones or other devices that might disturb others during the exam. You are NOT to use any electronic devices while taking the test, including calculators. Anyone caught using an electronic device will receive a 10 point penalty.

If you finish the exam early, please hand your exam to the instructor and exit quietly through the front door.

1. (6 points) **Tree Traversal**
   Show the order the nodes of this tree would be evaluated using the following traversals:



Preorder : `6,5,4,3,-2,19,1`

Inorder : `4,5,3,6,-2,1,19`

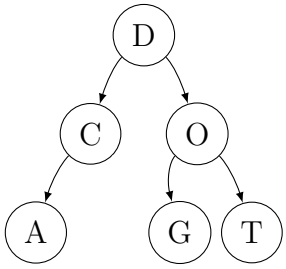
Postorder : `4,3,5,1,19,-2.6`

2. (4 points) **Binary Search Tree**
   Draw a picture below of the search tree that would result from inserting the following strings into an empty binary search tree in the following order:

   `"D" "O" "G" "C" "A" "T"`

   Assume the search tree uses alphabetical ordering to compare strings.

Solution:



3. (20 points) **Programming with Inheritance**

Imagine you have a class called MinMaxIntList that implements the following methods:

int size() - returns the number of elements in the list
int get(int index) - returns the element at the given index
String toString() - converts list to a string
void add(int val) - adds the value at the end of the list
void remove(int index) - removes the value at the index
int getMax() - returns the maximum value in the list, throws an IllegalStateException if the list is empty
int getMin() - returns the minimum value in the list, throws an IllegalStateException if the list is empty

The getMax() and getMin() methods are computationally expensive because they have to iterate over the whole list. Write a subclass of MinMaxIntList called FastMinMaxIntList that stores the min and max as fields that are updated when the list changes. Your new class should keep calls to MinMaxIntList's getMin and getMax to a minimum. Override getMax() and getMin() to use your variables, plus any additional methods necessary to keep your variables updated. The FastMinMaxIntList should always return the same results as the MinMaxIntList (although the speed of the various operations may be different).

Do not create any additional structured objects (arrays, ArrayLists, etc.) when solving this problem.

Solution:

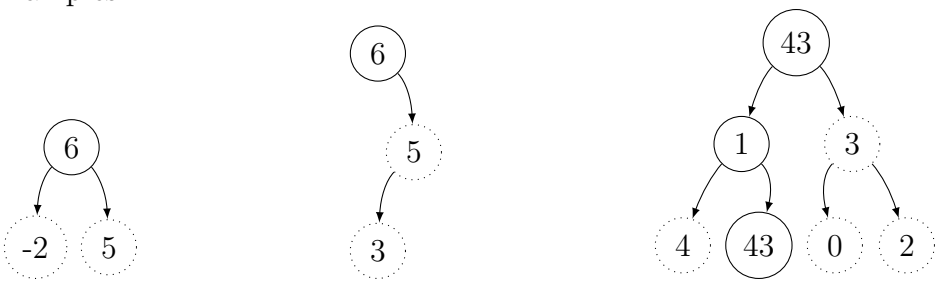
```
public class FastMinMaxIntList extends MinMaxIntList
{
  private int max,min;
  public int getMin() {
    if(size() == 0) throw new IllegalStateException("Empty List");
    return min;
  }
  public int getMax() {
    if(size() == 0) throw new IllegalStateException("Empty List");
    return max;
  }
  public void add(int val) {
    if(size() == 0) {
      max = min = val;
    } else {
      if(val > max) {
        max = val;
      } else if(val < min) {
        min = val;
      }
    }
    super.add(val);
  }
  public void remove(int index) {
    int val = get(index);
    super.remove(index);
    if(size() == 0) return;
    if(val == max) max = getMax();
    if(val == min) min = getMin();
  }
}
```

4. (20 points) **More Binary Trees**

Write a method removeSmallLeaves for the IntTree class we have discussed in lecture. This method should take an integer parameter and removes all leaf nodes in the tree less than the parameter. Note that when nodes are removed its possible that something that was a branch node might become a leaf – this new leaf should be removed as well if it is less than the minimum.

You may write helper methods to help you solve this problem, but you should not call any other methods of IntTree. Do not create any additional structured objects to solve this problem.

Examples:



Nodes with dotted lines are removed by removeSmallLeaves(6)

Solution:

```
public void removeSmallLeaves(int min) {
  overallRoot = removeSmallLeaves(min, overallRoot);
}

private IntTreeNode removeSmallLeaves(int min, IntTreeNode root) {
  if(root == null) return null;
  root.left = removeSmallLeaves(min,root.left);
  root.right = removeSmallLeaves(min, root.right);
  if(root.left == null && root.right == null) {
    if(root.data < min) {
      return null;
    }
  }
  return root;
}
```

```
// this is an alternative solution that does not use x = change(x)
// as a result it is a good deal longer
public void removeSmallLeaves(int min) {
  removeSmallLeaves(min, overallRoot);
  if(isLeaf(overallRoot) && overallRoot.data < min)
    overallRoot = null;
}

private boolean isLeaf(IntTreeNode node) {
  if(node == null) return false;
  return node.left == null && node.right == null;
}

private void removeSmallLeaves(int min, IntTreeNode root) {
  if(root == null) return;
  removeSmallLeaves(min, root.left);
  removeSmallLeaves(min, root.right);
  if(isLeaf(root.left) && root.left.data < min)
    root.left = null;
  if(isLeaf(root.right) && root.right.data < min)
    root.right = null;
}
```