CS163 Lab #10 – Programming Advanced Trees

Check-off each step as you proceed! Submit the code at the end of lab.

Collaboratively Implement Advanced Tree Algorithms

With this lab, we will be working with an existing class implementing both binary search trees and 2-3-4 of integers. You have access to the .h files to see what functions are available. *Assume that you have a BST of integers*

Implementing BST Algorithms – Collaborative Process:

A. Have each team member **implement each of the functions assigned in this lab**,

B. **Remember** to plan the code before writing it using these questions:
   i. What is the simple (base) case? (Also, known as the stopping condition)
   ii. What is the incremental step to get to the next smaller sub-problem?
   iii. What needs to get done before going to that next smaller sub-problem?
   iv. What needs to get done after returning from that smaller sub-problem?

C. Each team member should develop the test plan for their function

D. Each team member should evaluate their work with a pointer diagram, showing each invocation

E. Once implemented, **compare the code and test plan** with others in your group

F. **Evaluate** what worked and what didn't

G. **Evaluate** the test plan. Did it cover all the necessary cases?

H. **Summarize** what was learned.

Login to **cs163lab.cs.pdx.edu** using your assigned login and password

- Change into the CS163/Lab10 directory **cd CS163/Lab10**
- Use a **linux editor** such as **vi, vim, or emacs** to type in your recursive functions
- **Compile** and link to my object code on **linux**.

```
g++  *.cpp  *.o  -g  -Wall
```

- **Always** fix the **warnings** found by the -Wall.

Begin first with the binary search trees, **in cs163_tree.cpp** file. Examine the **cs163_tree.h** file for the correct prototypes. For each of these you will need to:

- Implement the wrapper function to call your recursive function.
- Modify main (**cs163_lab10.cpp**) to call your functions

Level 1 - Introductory

______Step 1. **Write the code** (recursively) to display the largest item in a BST

______a. What is the simple (base) case? ________________________

______b. What is the incremental step to get to the next smaller sub-problem

____

______c. What needs to get done before going to that next smaller sub-problem?

____

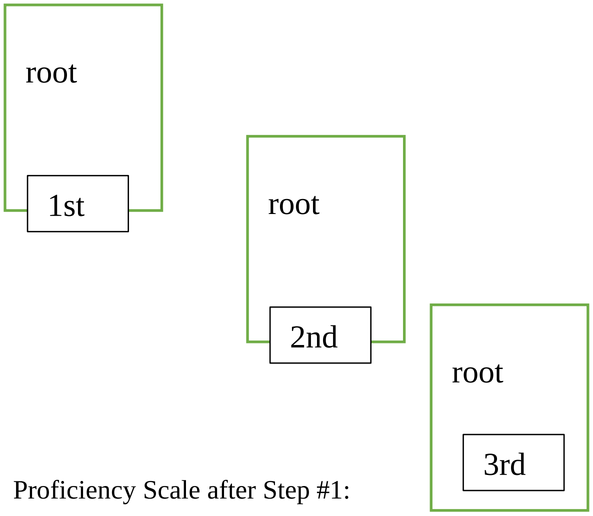
______d. What needs to get done after returning from that smaller sub-problem?

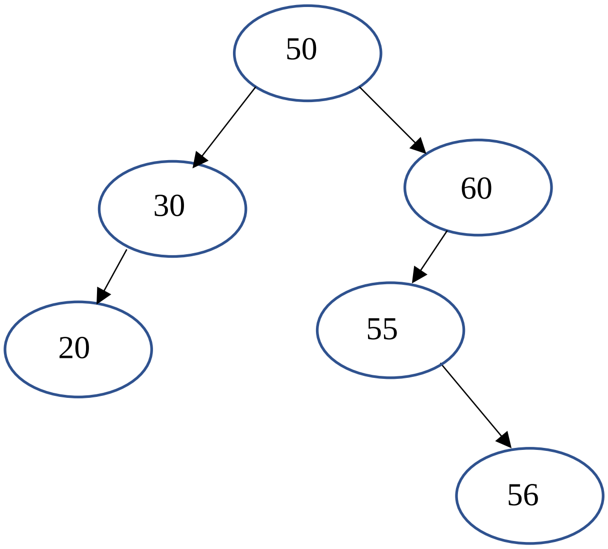
____

______e. **Develop the test plan:** Think about how to test the code, what flow of control exists in the member function, and how you will test all conditions:

Test Case(s)	**Expected Result**	**Verified? (yes/no)**

______f. Trace your code with a pointer diagram





Proficiency Scale after Step #1:

BST CODE

______Step 2. **Write the code** (recursively) to count the number of times the root's data appears in the list. Return the count and display it from main ______a. What is the simple (base) case? ________________________

______b. What is the incremental step to get to the next smaller sub-problem

____

______c. What needs to get done before going to that next smaller sub-problem?

____

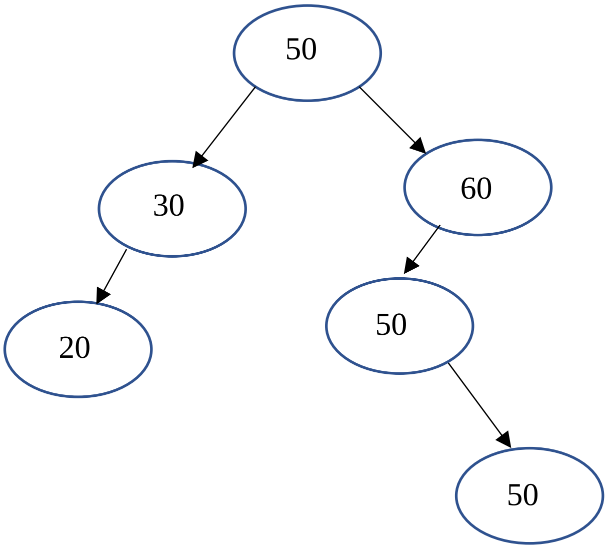
______d. What needs to get done after returning from that smaller sub-problem?

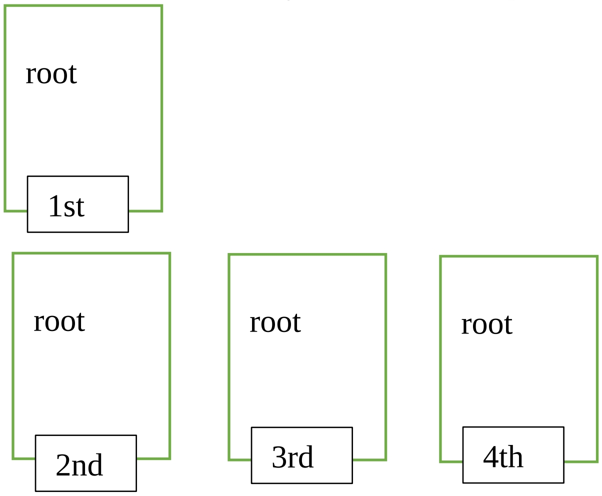
____

______e. **Develop the test plan:** Think about how to test the code, what flow of control exists in the member function, and how you will test all conditions:

Test Case(s)	**Expected Result**	**Verified? (yes/no)**

______f. Trace your code with a pointer diagram





Proficiency Scale after Step #2:

Level 2 - Intermediate

______Step 3. **Write the code** (recursively) to make a complete copy of a binary search tree, excluding the root of the original tree.

______a. What is the simple (base) case? ______________________

______b. What is the incremental step to get to the next smaller sub-problem

____

______c. What needs to get done before going to that next smaller sub-problem?

____

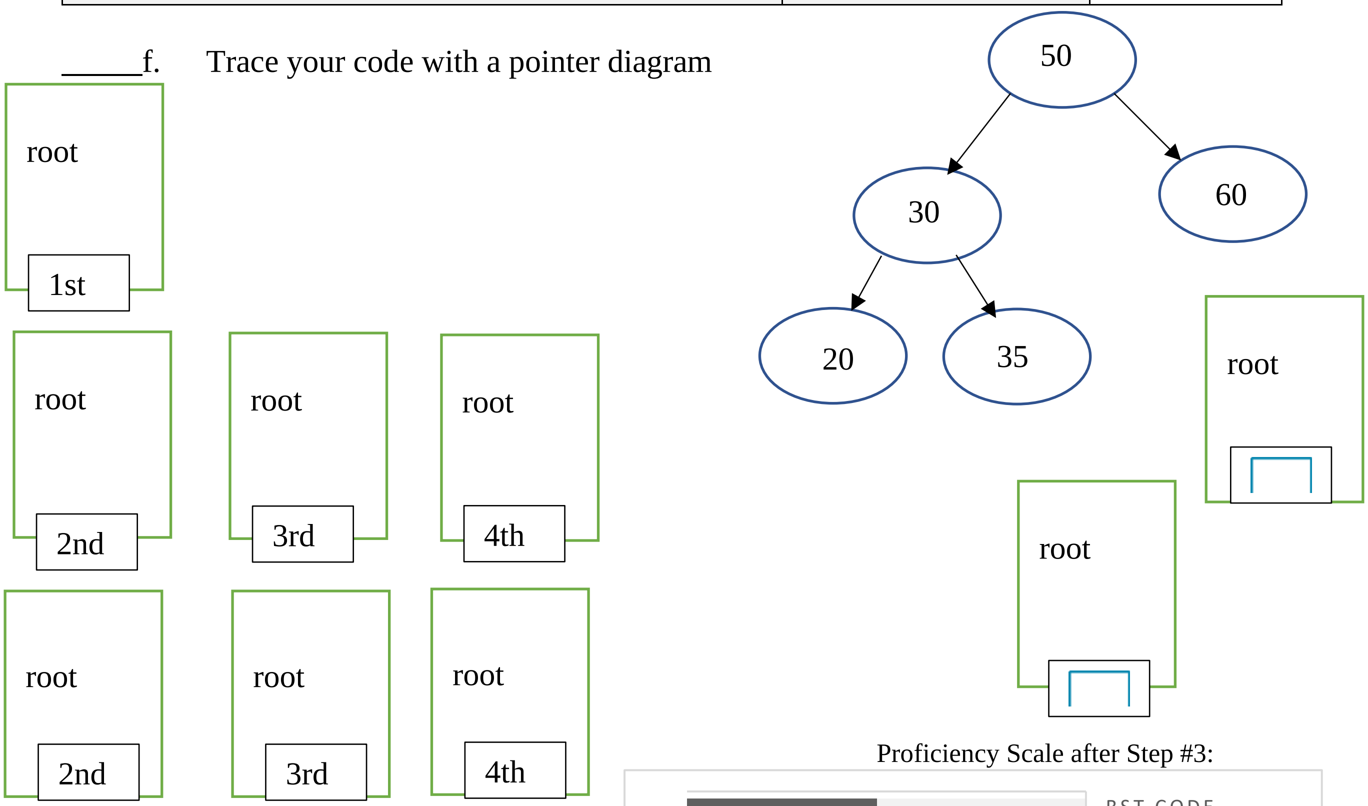
______d. What needs to get done after returning from that smaller sub-problem?

____

______e. **Develop the test plan:** Think about how to test the code, what flow of control exists in the member function, and how you will test all conditions:

Test Case(s)	**Expected Result**	**Verified? (yes/no)**

______f. Trace your code with a pointer diagram



Proficiency Scale after Step #3:

BST CODE

______Step 4. **Write the code** to count the number of nodes in a 2-3-4 tree that have 3 data items; return the count and display it from main _____a. What is the simple (base) case? ______________________

_____b. What is the incremental step to get to the next smaller sub-problem

____

_____c. What needs to get done before going to that next smaller sub-problem?

____

_____d. What needs to get done after returning from that smaller sub-problem?

____

_____e. **Develop the test plan:** Think about how to test the code, what flow of control exists in the member function, and how you will test all conditions:

Test Case(s)	**Expected Result**	**Verified? (yes/no)**

_____f. **Draw the pointer diagram:**

Proficiency Scale after Step #4:

Level 3 - Proficient

_____Step 5. **Challenge: Write the code** to determine the height of a 2-3-4 tree; return the value and display it from main

_____a. What is the simple (base) case? ______________________

_____b. What is the incremental step to get to the next smaller sub-problem

____

_____c. What needs to get done before going to that next smaller sub-problem?

____

_____d. What needs to get done after returning from that smaller sub-problem?

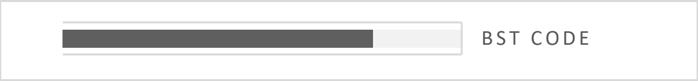
____

_____e. **Develop the test plan:** Think about how to test the code, what flow of control exists in the member function, and how you will test all conditions:

Test Case(s)	**Expected Result**	**Verified? (yes/no)**

_____f. **Draw the pointer diagram:**

Proficiency Scale after Step #5:



_____Step 6. **Challenge: Write the code to** (*recursively*) to copy a 2-3-4 tree; return zero if the original tree was empty

_____a. What is the simple (base) case? ______________________

_____b. What is the incremental step to get to the next smaller sub-problem

____

_____c. What needs to get done before going to that next smaller sub-problem?

____

_____d. What needs to get done after returning from that smaller sub-problem?

____

_____e. **Develop the test plan:** Think about how to test the code, what flow of control exists in the member function, and how you will test all conditions:

Test Case(s)	**Expected Result**	**Verified? (yes/no)**

_____f. **Draw the pointer diagram:**

Proficiency Scale after Step #6:

_____Step 7. **Tar the files** for the lab by typing:

```
tar   -cvf    CS163_Lab10.tar   *.cpp   *.h
```

_____Step 5. **List the files** in the directory to confirm that the .tar file exists

_____Step 6. **Submit** your program by typing **./submit** at the linux prompt