CS163 Lab #7 – BST Programming

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

Collaboratively Implement BST Algorithms

With this lab, we will be working with an existing class implementing a BST of integers. You have access to the .h files to see what functions are available. Your job will be to implement functions **recursively**. *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 of the necessary cases?

H. **Summarize** what was learned.

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

- Change into the CS163/Lab7 directory **cd CS163/Lab7**
- 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

Begin implementing the member functions, in **cs163_bst.cpp** file. Examine the **cs163_bst.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_lab7.cpp**) to call your functions

Level 1 - Introductory

_____Step 1. **Count the number of nodes in a BST with no children; return the count**

_____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?

_____

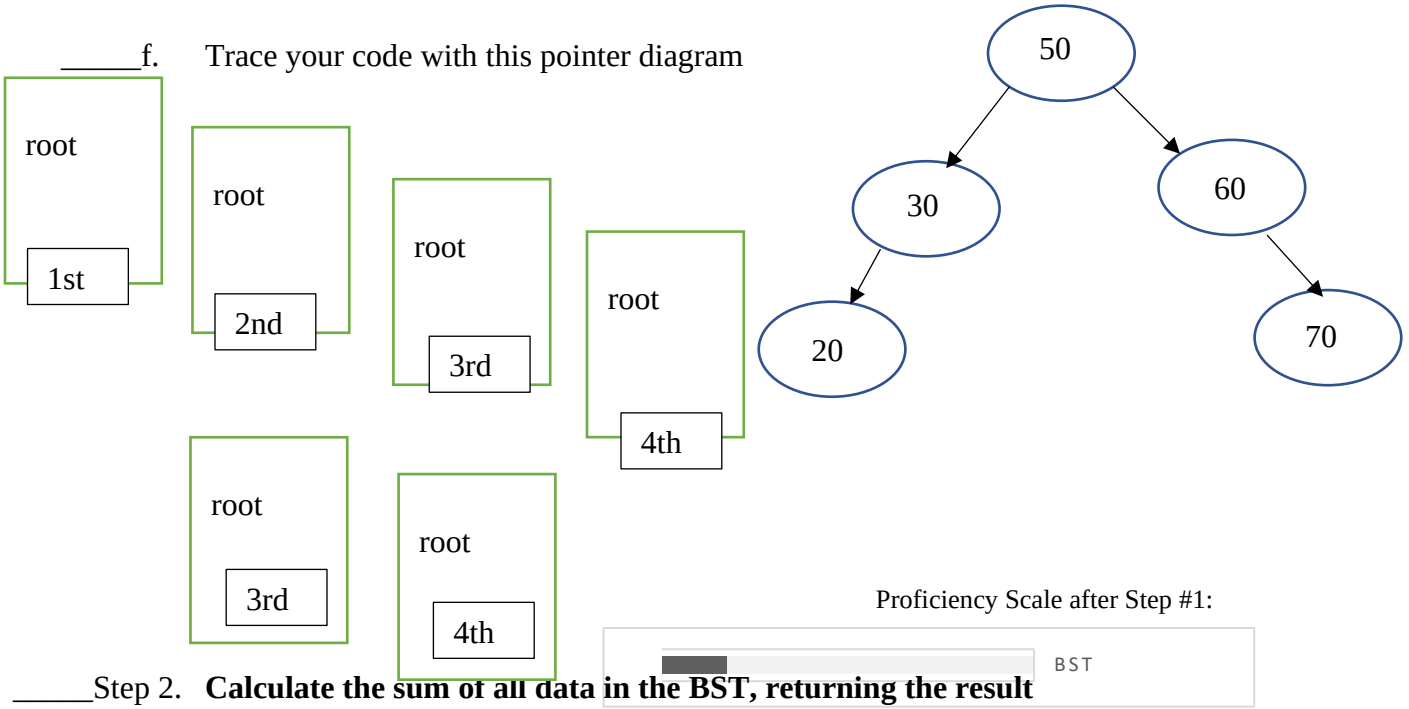
_____d. What needs to get done after returning from that smaller sub-problem?

_____

_____c. **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?

_____f. Trace your code with this pointer diagram



Proficiency Scale after Step #1:

BST

_____Step 2. **Calculate the sum of all data in the BST, returning the result**

_____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?

_____

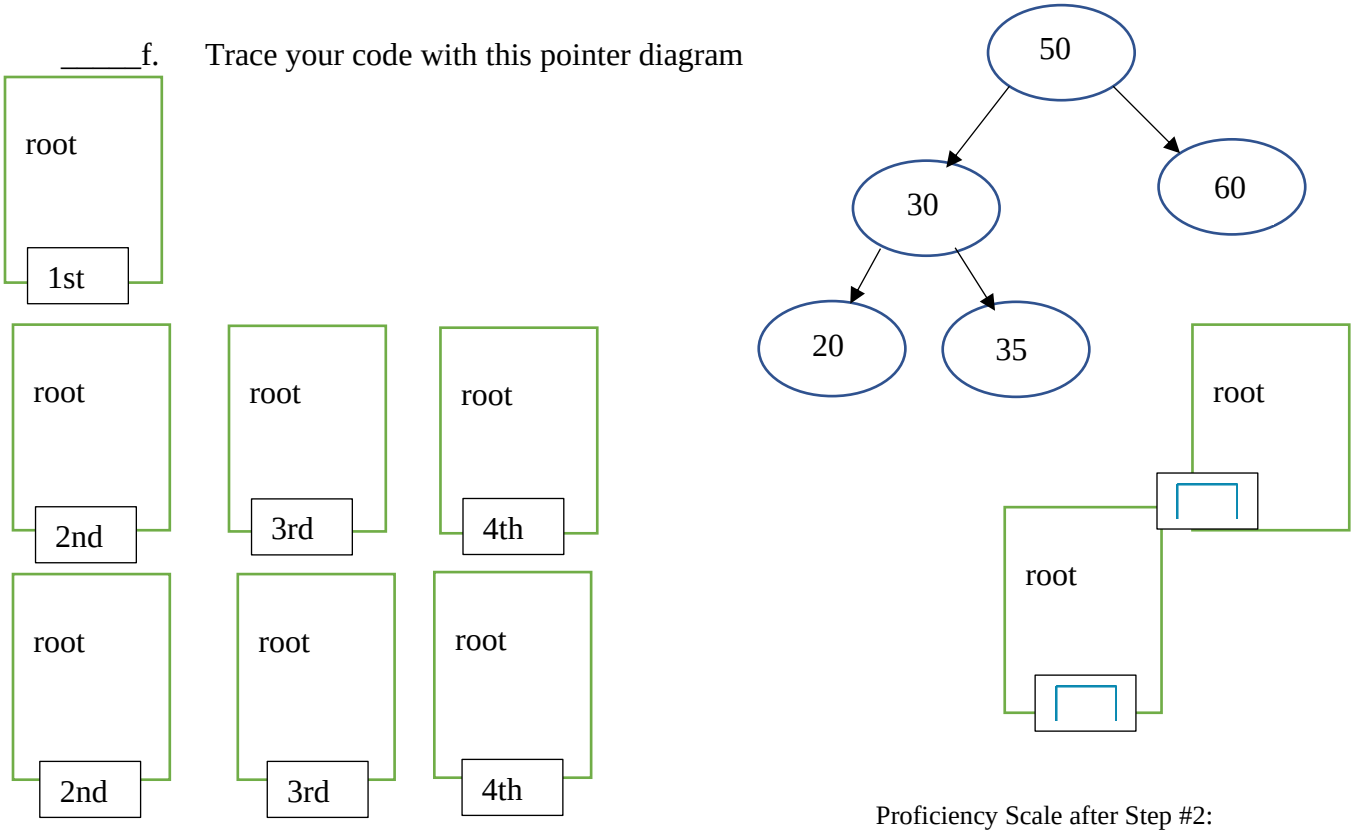
_____d. What needs to get done after returning from that smaller sub-problem?

_____

_____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?

_____f. Trace your code with this pointer diagram



Proficiency Scale after Step #2:

Level 2 - Intermediate

_____Step 3. **Traverse the tree to determine the height**

_____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?

_____

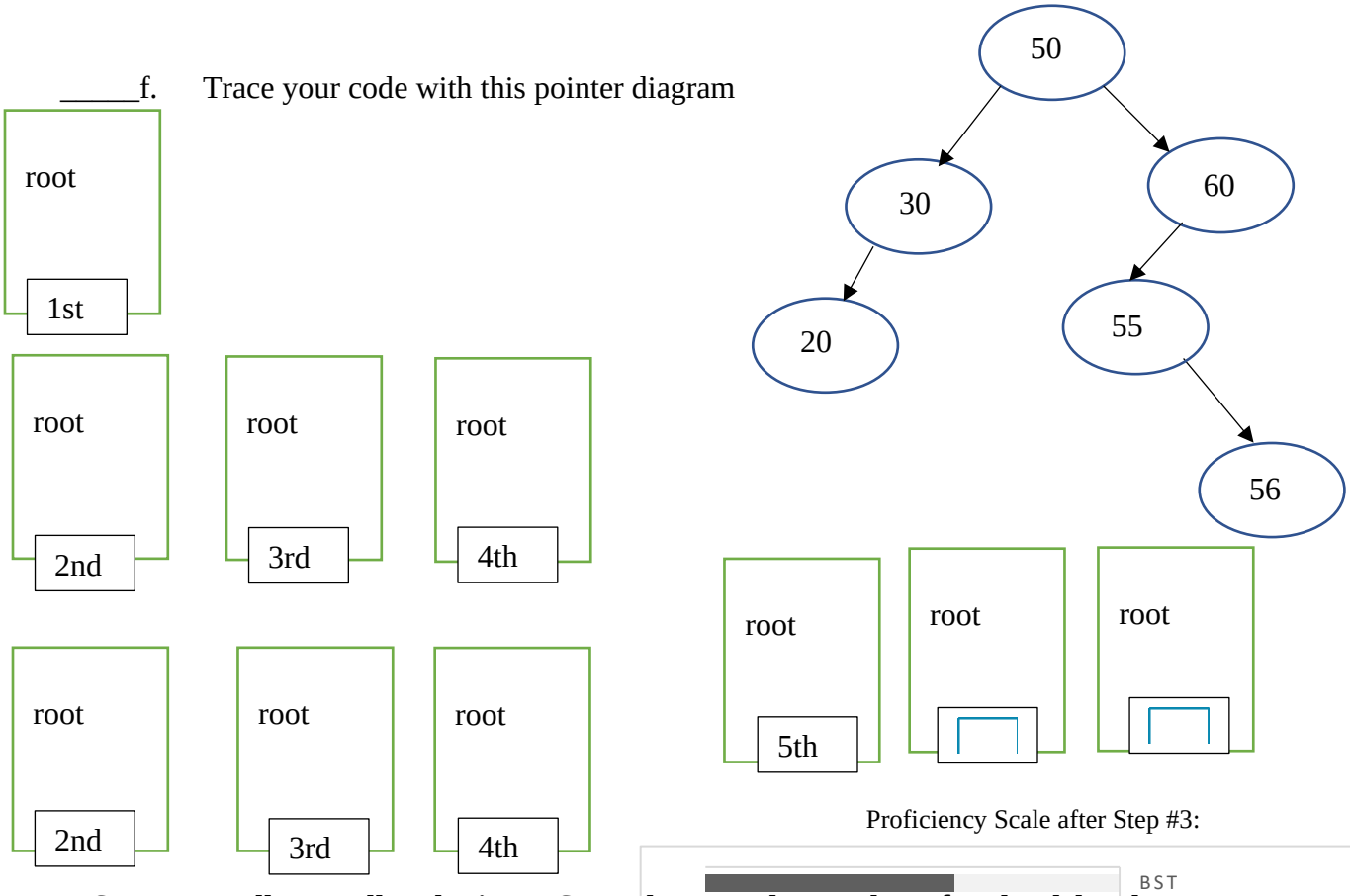
_____d. What needs to get done after returning from that smaller sub-problem?

_____

_____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?

_____f. Trace your code with this pointer diagram



Proficiency Scale after Step #3:

BST

_____Step 4. **Deallocate all nodes in a BST and return the number of nodes deleted**

_____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?

_____

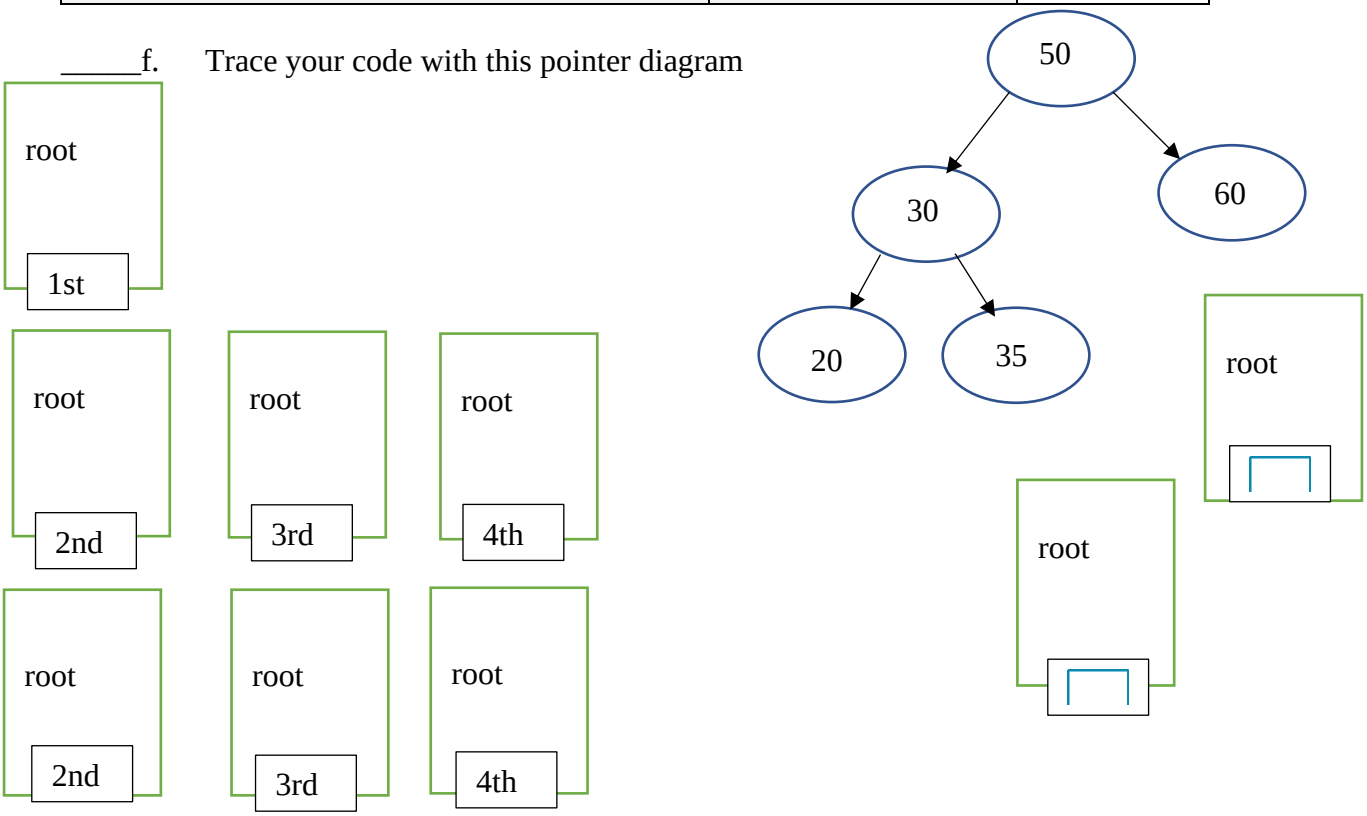
_____d. What needs to get done after returning from that smaller sub-problem?

_____

_____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?

_____f. Trace your code with this pointer diagram



Proficiency Scale after Step #4:

Level 3 - Proficient

_____Step 5. **IMPORTANT!** Make a complete copy of a BST, creating a new BST and return the number of nodes copied

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

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

_____

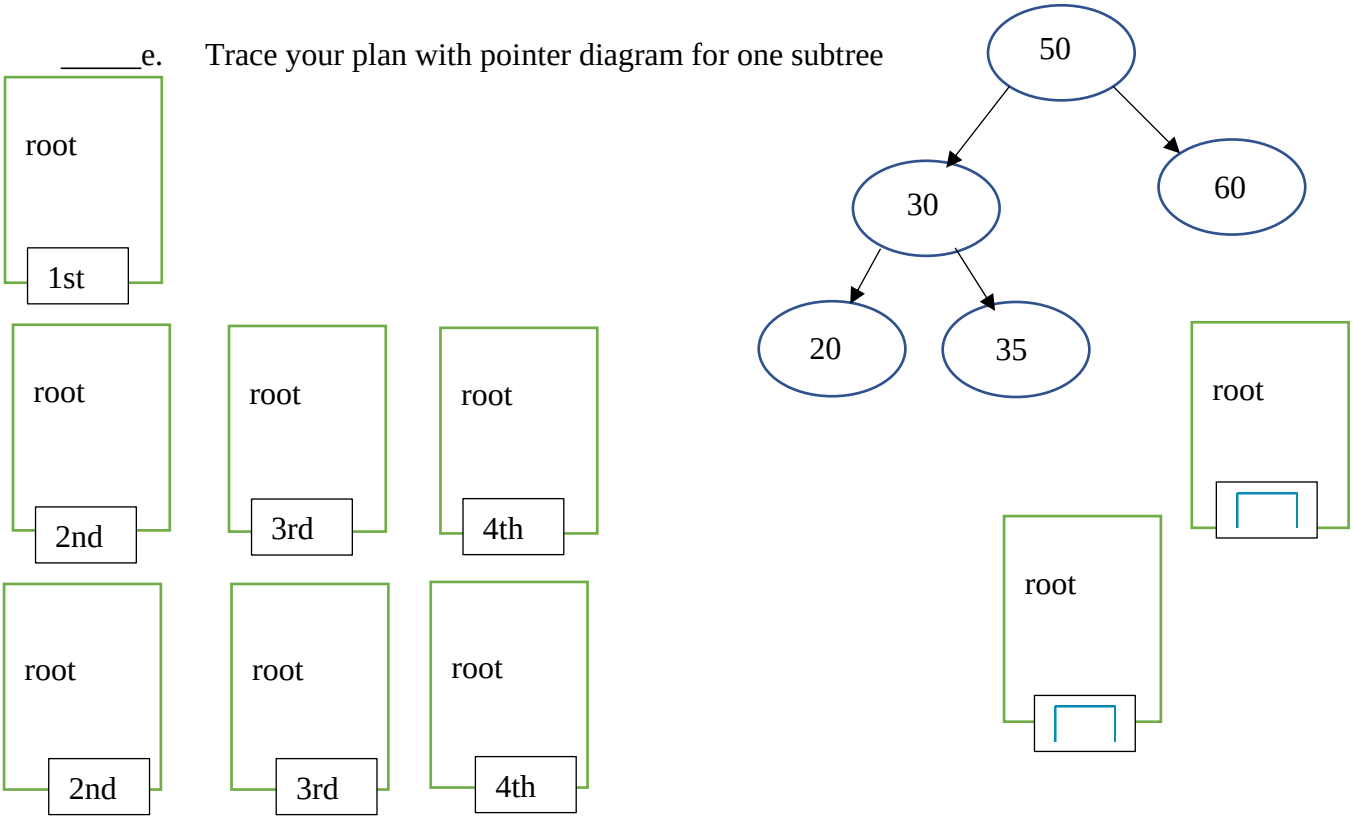
_____c. What needs to get done before going to that next smaller sub-problem?

_____

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

_____

_____e. Trace your plan with pointer diagram for one subtree



Proficiency Scale after Step #5:

_____Step 6. **Tar the files** for the lab by typing:

tar -cvf CS163_Lab7.tar *.cpp *.h

_____Step 7. **Submit** your program by typing **./submit** at the linux prompt