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.

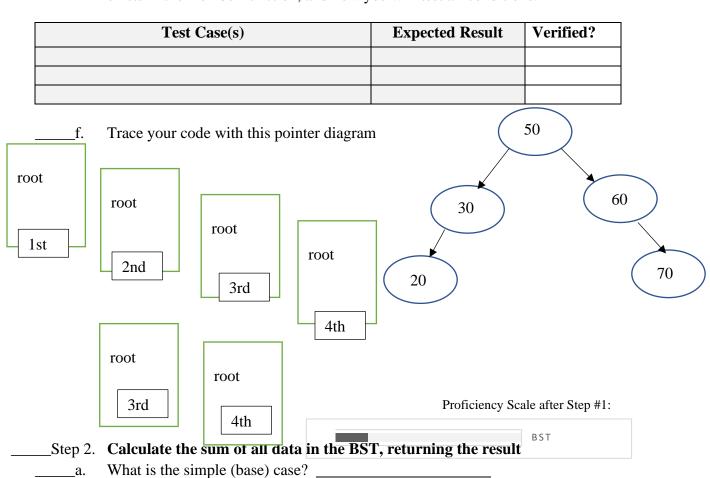
Begin implementing the member functions, in **cs163_bst.cpp** file. Examine the **cs163_bst.h** file for the correct prototypes. For each of these you will need to:

• 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-probler
	d. What needs to get done after returning from that smaller sub-problem
	Develop the test plan: Think about how to test the code, what flow of convists in the member function, and how you will test all conditions:



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

c.	What needs to get done before goi	ng to that next smaller sub-prob	olem?
d.	What needs to get done after return	ning from that smaller sub-prob	lem?
e. ex	Develop the test plan: Think ab ists in the member function, and ho		ow of control
	Test Case(s)	Expected Result	Verified?
f.	Trace your code with this pointer	diagram	50
root			60
		30	00
1st		20 35	
root	root root	20 33	root
2nd	3rd 4th	ro	oot
root	root root		
		L	
2nd	3rd 4th	Proficiency Scal	e after Step #2:
			BST

Level 2 - Intermediate

a.	What is the simple (bess) sees?	e height	
b.	What is the simple (base) case? What is the incremental step to get to	to the next smaller sub-problem	em
c.	What needs to get done before going	g to that next smaller sub-pr	oblem?
d.	What needs to get done after returning	ng from that smaller sub-pro	oblem?
e.	Develop the test plan: Think about tists in the member function, and how	you will test all conditions:	,
	Test Case(s)	Expected Result	Verified?
root	Trace your code with this pointer dia	agram 30	55
		20	33
root	root root		56
root 2nd	root root 4th	root	56
		root root	
2nd	3rd 4th	5th	root cale after Step #3:
root	root root 4th 4th	Sth Proficiency S	root cale after Step #3:

b.	What is the inc	remental step to get to	o the next smaller sub-prol	blem	
c.	What needs to g	get done before going	g to that next smaller sub-p	problem?	
d.	What needs to g	get done after returni	ng from that smaller sub-p	roblem?	
e.		er function, and how	t how to test the code, who you will test all conditions Expected Result		ol -
f.	Trace your cod-	e with this pointer dia	agram	50]
root 1st			30		60
root	root	root	20	35	root
2nd	3rd	4th		root	
root 2nd	root 3rd	root 4th			
			Proficiency	Scale after Step #4	4:
			BST		
Level 3 - Pro	<u>oficient</u>				

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

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 root r	a.	What is the simple (base) case?	
d. What needs to get done after returning from that smaller sub-problem? e. Trace your plan with pointer diagram for one subtree root roo	b.	What is the increment step to get to the next smaller sub-problem	
e. Trace your plan with pointer diagram for one subtree root	c.	What needs to get done before going to that next smaller sub-problem?	
root root	d.	What needs to get done after returning from that smaller sub-problem?	
root root root root root root	e.	Trace your plan with pointer diagram for one subtree 50	
root root root root root		30 60	
2nd 3rd 4th root	1st	20 35	7
root root root	root	root root root	
root root	2nd		
2nd 3rd 4th	root		
	2nd	3rd 4th	
Proficiency Scale after Step #5:		Proficiency Scale after Step #5:	
BST		BST	
Step 6. Tar the files for the lab by typing:	Step 6.	• • • •	
tar -cvf CS163_Lab7.tar *.cpp *.h Step 7. Submit your program by typing ./submit at the linux prompt	Step 7.		