# CSCI-1200 Data Structures — Fall 2022
# Lab 11 — `ds_set` Implementation, part II

## Checkpoint 1

Checkpoint 1 will be available at the start of Wednesday's lab.

## Checkpoint 2

***estimate: 15-30 minutes***

Download these files:

http://www.cs.rpi.edu/academics/courses/fall22/csci1200/labs/11_trees_II/ds_set.h
http://www.cs.rpi.edu/academics/courses/fall22/csci1200/labs/11_trees_II/test_ds_set.cpp

Implement and test the decrement operator for `tree_iterator`. Determine an appropriate sequence to *insert* the numbers 1-15 such that the resulting tree is *exactly balanced*. After using the `print_sideways` function to confirm the construction of this tree, test your iterators on the structure. Similarly, create a couple unbalanced trees to demonstrate that both the increment and decrement operators for iterators are debugged. Your decrement operator should correctly decrement the `end()` iterator. You can use the same "trick" we used in Lab 6 to make this work for `ds_list` iterators. Ask a TA or mentor if you have any questions.

**To complete this checkpoint:** Show your TA or mentor your iterator decrement code and your test cases.

## Checkpoint 3

***estimate: 10-20 minutes***

Now let's monitor the "work" done by the increment and decrement functions. Add two global counter variables named `up_count` and `down_count` to your program. *Yes, global variables are bad in general! It's ok for this inspection task for the purpose of debugging & confirming our discussion of Big O Notation.*

Initialize these global counters to zero. When an iterator moves down one link in the tree, increment the `down_count` variable. When an iterator moves one link up in the tree, increment the `up_count` variable.

Now, iterate from `begin()` to `end()` over your tree, and print out these global counters after each step. Do the counts match the distance between the nodes in the diagram? The number of links followed up and down is a good measurement of the total cost / running time of the iteration functions. What is the worst case cost for a call to `operator++` on a tree with $n$ nodes? What is the total cost for iterating over all $n$ nodes in the whole tree? What is the average or *amortized* cost for a call to `operator++` on a tree? Does this match what we discussed in Lecture 19?

**To complete this checkpoint:** Show your TA or mentor your modified iterator increment code and your global counter inspection code. Be prepared to discuss your measurements of the cost of each step of forward iteration and the overall best/worst/average/total Big O Notation for forward iteration.

Use the remainder of the lab time to work on Homework 8. Ask the TAs for help! Make sure your code is making proper use of memory for Homework 8. Run your code with a memory debugger (Dr. Memory or Valgrind) on your own machine.