# CSCI-1200 Data Structures — Fall 2025
# Lab 10 — Binary Search Trees
# & `ds_set` Implementation, part I

## Checkpoint 1

Checkpoint 1 will be available at the start of Wednesday's lab.
*It will be a team-of-two paper & pencil worksheet to be completed with one other person from your lab section.*

## Checkpoint 2

***estimate: 15-25 minutes***

Now let's explore the implementation of the `ds_set` class, along with the use of recursive functions to manipulate binary search trees. Download and examine the files:

http://www.cs.rpi.edu/academics/courses/fall25/csci1200/labs/10_trees_I/ds_set.h
http://www.cs.rpi.edu/academics/courses/fall25/csci1200/labs/10_trees_I/test_ds_set.cpp

The implementation of `find` provided in `ds_set.h` is recursive. Re-implement and test a non-recursive replacement for this function.

The provided `test_ds_set.cpp` has basic tests of insert, find, and `print_sideways_tree`. Write additional test cases including: a type other than `string`; using insert to create a perfectly balanced tree, using insert to create an extremely unbalanced tree; using find on a tree with no elements, with 1 element, the element is at the root, the element is in the middle, the element is at the leaf, the tree is balanced, the tree is unbalanced; and anything else you can think of.

**To complete this checkpoint:** Show one of the TAs your new code and variety of tests. Be prepared to discuss the Big O Notation for running time for the two different versions of `find` for various inputs – what is the best case, worst case, and average case?

## Checkpoint 3

***estimate: 15-25 minutes***

The implementation of the copy constructor and the assignment operator is not yet complete because each depends on a private member function called `copy_tree`, the body of which has not yet been written. Write `copy_tree` and then test to see if it works by "uncommenting" the appropriate code from the main function. You'll also have to type up the implementation of `destroy_tree` from yesterday's lecture.

Run the code through the Dr Memory or Valgrind memory debuggers and make sure you have no memory errors or memory leaks. Write a few additional test cases to make sure the copy function works with an empty tree, a perfectly balanced tree, an extremely unbalanced tree, and anything else you can think of.

**To complete this checkpoint:** Show one of the TAs your new code. What is the Big O Notation of `copy_tree` for a tree with $n$ nodes and height $h$? Does it matter if the tree is balanced or unbalanced?

**OPTIONAL: Bring your finished or nearly finished Homework 7 Data Structure Diagram to lab discuss the design with a TA/mentor.**