# COP 3538 – Data Structures with OOP

## Project 4 – Spring 2012

**Due: 2pm, Monday, 2 April 2012**

**Drop dead date/time: 2pm, Friday, April 6, 2012**

# Binary Trees

Using NetBeans 6.9 or later version, you are to write a Java program using OOP principles to accommodate the following functionality

## Assignment #4

**Objectives**:

Provide student with additional experiences with file input output.
Provide student with expanded exercises in UML
Provide student with exercises in Javadoc and its various formats
Provide student with exercises in building a binary tree with object nodes
Provide student opportunity to display binary trees in various traversals
Provide student with experience in inserting and deleting node contents of binary trees.

**Functionality:**

**Task 1: Build an array of String objects from the input file, ECountriesTrees.txt.**

Given a sequential file, ECountriesTrees.txt, on my web page, you are to build an array of strings, one string per input record (line). You are to use an InputOutputClass.

**Task 2: Build a Queue of Tree objects**

From the array of string objects, you are to build a queue of objects that will later be used to build a binary search tree. Each string object must be parsed into a tree-object format, such that each object in the queue is to contain the country name, the country's capital, the capital's population and also have room for two links: left and right child pointers that you will need when these objects are inserted into the binary search tree. (If you do not include room for these two pointers, you can add these two self-referential pointers to the queue object when you create the tree.) As you create these objects, insert() them into your queue . Remember, as you insert() each object into the queue, use the rear pointer to the queue. You will also need to investigate and use String Tokenizer to parse each string into an object, because the input file is comma-delimited.

**Task 3. Display the Queue of Tree Objects**. Proceeding from the front of the queue advancing to the rear, display the objects in the queue in a suitable, professional format (i.e. a header line and columns underneath; numeric fields with commas, etc.)

**Task 4: Build a Binary Search Tree from the Queue of Tree objects**

Using the queue of objects and starting from the front, build the binary search tree. Each node is to contain all the properties in each queue object plus a left and right

self-referential set of pointers. The binary search tree is to be built in the same order as the states appear in your tree queue.

**Task 5: Display the Binary Search Tree.** You are to display the tree using a recursive LNR scan. The format is to contain a header, left justified on the display line that contains the entries: **Recursive LNR Scan** followed by a second header underneath with text: **Country Name** **Capital** There is to be a blank line prior to the first header and all subsequent print lines are to be single spaced (no intervening blank lines) hereafter. Display the objects underneath the headers cited above.

**Task 6: Display the Binary Search Tree.** You are to display the tree using **an iterative LNR** scan.

Same formats as above. But your first header says: **Iterative LNR Scan**.

**Task 7: Display the binary Search tree** using an RNL **recursive** scan. Same directions for the headers.

**Task 8. Display the binary search tree** using an RNL **iterative** scan. (This is tricky)

**Task 8.** Given your binary search tree, **delete Belarus, Ukraine, and Latvia**. (each of these have no children)

**Task 9. Display an LNR recursive scan** with header citing deleted Belarus, Ukarine, and Latvia. Do: LNR Recursive Scan (sans Belarus, Ukraine, Latvia)

**Task 10.** Given your binary search tree, **delete Denmark and Norway.** (each of these have one child)

**Task 11. Display an LNR recursive scan** with header .

LNR Recursive Scan (sans Denmark and Norway)

**Task 12.** Given your binary search tree, **delete Sweden and Estonia**. (each of these have two children)

**Task 13. Display an LNR recursive scan** with header citing Minnesota was deleted.

LNR Recursive Scan (sans Sweden and Estonia)

**Design Suggestion:** Recommend you have a Tree Collection class that is called from Main that in turn invokes the IO routines in I/O Class object, builds the queue of objects from the array of string objects, builds the binary search tree, and then performs all maintenance (inserting, deleting, displaying) on the tree. Offhand for starters, you will need a Main class, I/O class, Tree collection class, queue class (for queue objects). An alternative design might split up the functionality in tree collection class to building the queue of tree objects and then separating the building of the tree and tree maintenance to another class.

**Deliverable:** Your zipped folder to me MUST include a copy of your input file. I will need these to run your program and to test it. Do not provide me with output your program generates. I will get your program to generate your outputs. As noted, your outputs will be displayed on the screen.

You are to zip all files in your P4 as expected and Send them to be via the assignment tab using the same naming conventions as in P0. **Be certain your zipped file runs! Please note: use zip files. If you use .rar files, I do not guarantee we can grade – and this is bad.** For UML, you may use Word, Power Point, Visio, or SmartDraw

Unzip it and run it locally before you send it to me. If it does not work, I cannot grade it.