The primary purpose of this lab is to become familiar with the operation of a binary search tree, and its representation in memory. This lab (Lab 21a) includes the **`IntBinaryTree`** class from the textbook, with several enhancements to show the internal data, and a few new member functions. The "starter code" program works, but the '**D**' and '**H**' commands are incomplete.

Optional Lab Exercise:

Because we are getting very close to the end of the course, this lab exercise is an **optional** lab exercise. If you complete, and demonstrate in class, the solution for this Lab Exercise, then it will help your grade slightly, but if you do NOT complete it, that will NOT reduce your overall course grade.

Lab 21a Starter Code

Download the ZIP file from *Moodle*.

Extract the files and add the **.cpp** and **.h** files to your C++ project.

This is a typical command-loop lab program, that creates an **`IntBinaryTree`** object. It also allows the user to add nodes, display the tree, and remove nodes by means of the following interactive commands:

- the '**C**' (upper-case 'C') removes ALL nodes from the tree ("Clobber").
- the '**d**' (lower-case 'd') command displays the binary tree nodes by **INorder** traversal
- the '**D**' (upper-case 'D') command implements an enhanced version of the tree display (**lab exercise**)
- the '**h**' command prints "help" text.
- the '**H**' (upper-case 'H') command displays the tree height (**lab exercise**).
- the '**i**' command allows the user to insert a node into the tree, with a user-specified value
- the '**n**' command displays the number of nodes in the tree.
- the '**post**' command displays the binary tree nodes by POSTorder traversal.
- the '**pre**' command displays the binary tree nodes by PREorder traversal.
- the '**q**' command exits the program.
- the '**r**' command first deletes any tree that is already loaded, and then opens an input (text) file and reads the contents into the binary tree.
- the '**R**' (upper-case 'R') command removes a selected node from the tree.
- the '**s**' command searches the tree for a user-specified value
- the '**v**' command sets or clears VERBOSE mode.

Lab Exercise

1. Compile and run the program. Verify that you can create a binary tree, by using the '**r**' command. (Several sample text files are provided: `tree01.txt`, `tree02.txt`, and `tree03.txt`.)
2. Use the'**d**' command to display the contents of the tree. Observe the screen output and visually trace your way through the tree by following the node addresses, beginning with the value of root, and following the left and right link values. (To assist reading the output, refer to the *tree diagrams* included on page 3 of this document.)
3. Also try the "**pre**" and "**post**" commands, to observe the difference between the different traversal methods.
4. Try adding one or more nodes to the tree, using the '**i**' command.

5. Use the '**n**' command to display the number of nodes present.
6. Use the '**R**' command to remove nodes from the tree, and verify the result with the '**d**' command. Be sure to try removing some "leaf" nodes as well as nodes with one and two "child" nodes.

Programming Exercise

To complete the programming portion of this lab exercise, most or all of your code changes need to be in the **IntBinaryTree.h** and **IntBinaryTree.cpp** files.

7. Complete the implementation of the '**H**' command (Tree Height) by filling in the missing code in the **`IntBinaryTree::calculateSubTreeHeight`** function. The '**H**' command should return the maximum tree height in the same way that the '**n**' command returns the number of nodes in the tree. (Recall the definition of *Tree Height*, that we discussed in class.)
   HINT: use the **`IntBinaryTree::calculateNumberOfNodes`** function as a guide.

8. Implement the '**D**' command by completing the **`IntBinaryTree::enhancedTreeDisplay`** function. This command must display the tree contents like the '**d**', '**pre**', and '**post**' commands do, but also add the current node depth (within the tree) of each node to the output. The display order ("INorder", "PREorder", or "POSTorder") is your choice. An example of the possible output is shown in the table below.

   HINT: You may want to implement a new version of the **`displayNode`** function. This new, overloaded function could have a function prototype similar to the example below:

```
IntBinaryTree::displayNode(TreeNode *nodePtr, int depth) const;
```

(In the sample output shown below, the text that the user types is shown in **`bold`** font. In actuality, all text will appear in the same font.)

Enhanced INorder display of binary tree (includes depth of each node at the left)

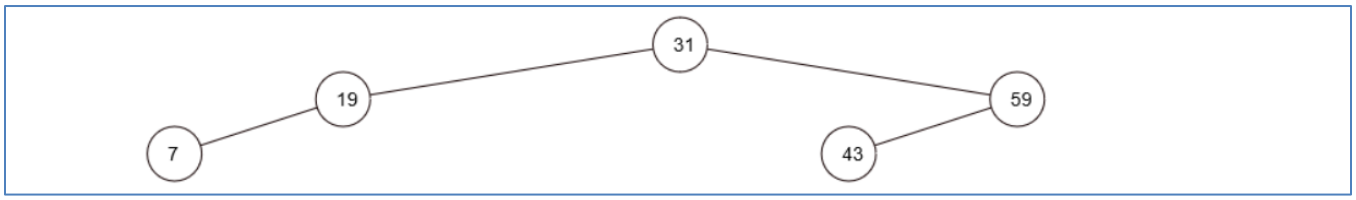
```
Command: r
Enter name of input file: tree01.txt
5 lines read from input file.

Command: D
root=000001FFD1CFFA50
( 2) node=000001FFD1CFFC90 value=   7 left=nullptr          right=nullptr
( 1) node=000001FFD1CFF870 value=  19 left=000001FFD1CFFC90 right=nullptr
( 0) node=000001FFD1CFFA50 value=  31 left=000001FFD1CFF870 right=000001FFD1CFFAB0
( 2) node=000001FFD1CFF030 value=  43 left=nullptr          right=nullptr
( 1) node=000001FFD1CFFAB0 value=  59 left=000001FFD1CFF030 right=nullptr

Command:
```

Sample Data Files

Several sample data files are provided with the "starter code" source files. When these files are loaded using the "**r**" command, they produce binary trees that resemble the following illustrations.[1]

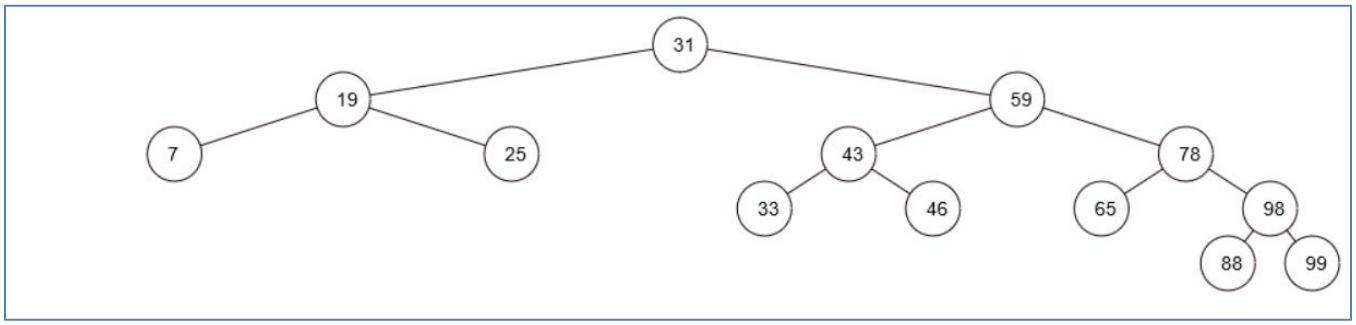
tree01.txt:

```
31 19 7 59 43
```



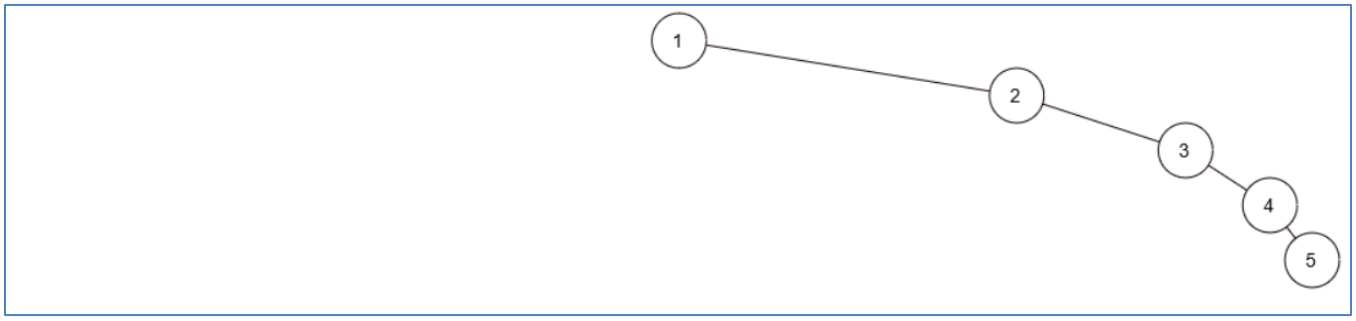
tree02.txt:

```
31 19 7 59 43 78 98 33 46 88 99 65 25
```



tree03.txt:

```
1 2 3 4 5
```



[1] These diagrams were created using this web page:
https://liveexample.pearsoncmg.com/dsanimation13ejava/BSTeBook.html

Sample Input / Output

In the sample session shown below, the text that the user types is shown in **bold** font. In actuality, all text appears in the same font.

Sample Input / Output

```
Command: h
Supported commands:
        C       Remove ALL nodes from binary tree (clobber)
        d       display binary tree by INorder traversal.
        D       Enhanced display of binary tree (lab exercise).
        h       print this help text.
        H       display Maximum tree Height (lab exercise).
        i       Insert node to binary tree
        n or N  display number of nodes in the tree.
        post    display binary tree by POSTorder traversal.
        pre     display binary tree by PREorder traversal.
        q       quit (end the program).
        r       open input file, read into binary tree.
        R       Remove node from binary tree
        s       Search for node in binary tree
        v       Set or Clear VERBOSE mode


Command: r
Enter name of input file: tree01.txt
5 lines read from input file.

Command: d
     INorder traversal:
root=00000217DE76FBB0
node=00000217DE76FD90 value=       7 left=nullptr          right=nullptr
node=00000217DE76FA90 value=      19 left=00000217DE76FD90 right=nullptr
node=00000217DE76FBB0 value=      31 left=00000217DE76FA90 right=00000217DE76F2B0
node=00000217DE76F190 value=      43 left=nullptr          right=nullptr
node=00000217DE76F2B0 value=      59 left=00000217DE76F190 right=nullptr

Command: pre
     PREorder traversal:
root=00000217DE76FBB0
node=00000217DE76FBB0 value=      31 left=00000217DE76FA90 right=00000217DE76F2B0
node=00000217DE76FA90 value=      19 left=00000217DE76FD90 right=nullptr
node=00000217DE76FD90 value=       7 left=nullptr          right=nullptr
node=00000217DE76F2B0 value=      59 left=00000217DE76F190 right=nullptr
node=00000217DE76F190 value=      43 left=nullptr          right=nullptr

Command: post
     POSTorder traversal:
root=00000217DE76FBB0
node=00000217DE76FD90 value=       7 left=nullptr          right=nullptr
node=00000217DE76FA90 value=      19 left=00000217DE76FD90 right=nullptr
node=00000217DE76F190 value=      43 left=nullptr          right=nullptr
```

Sample Input / Output

```
node=00000217DE76F2B0 value=      59 left=00000217DE76F190 right=nullptr
node=00000217DE76FBB0 value=      31 left=00000217DE76FA90 right=00000217DE76F2B0

Command: H
Height of tree = 2
Command: D
root=00000217DE76FBB0
( 2) node=00000217DE76FD90 value=   7 left=nullptr  right=nullptr
( 1) node=00000217DE76FA90 value=  19 left=00000217DE76FD90 right=nullptr
( 0) node=00000217DE76FBB0 value=  31 left=00000217DE76FA90 right=00000217DE76F2B0
( 2) node=00000217DE76F190 value=  43 left=nullptr  right=nullptr
( 1) node=00000217DE76F2B0 value=  59 left=00000217DE76F190 right=nullptr

Command: n
Population of tree = 5
Command: i
Enter value to insert into tree: 75

Command: H
Height of tree = 2
Command: D
root=00000217DE76FBB0
( 2) node=00000217DE76FD90 value=   7 left=nullptr          right=nullptr
( 1) node=00000217DE76FA90 value=  19 left=00000217DE76FD90 right=nullptr
( 0) node=00000217DE76FBB0 value=  31 left=00000217DE76FA90 right=00000217DE76F2B0
( 2) node=00000217DE76F190 value=  43 left=nullptr          right=nullptr
( 1) node=00000217DE76F2B0 value=  59 left=00000217DE76F190 right=00000217DE76F910
( 2) node=00000217DE76F910 value=  75 left=nullptr          right=nullptr

Command: i
Enter value to insert into tree: 99

Command: H
Height of tree = 3
Command: D
root=00000217DE76FBB0
( 2) node=00000217DE76FD90 value=   7 left=nullptr          right=nullptr
( 1) node=00000217DE76FA90 value=  19 left=00000217DE76FD90 right=nullptr
( 0) node=00000217DE76FBB0 value=  31 left=00000217DE76FA90 right=00000217DE76F2B0
( 2) node=00000217DE76F190 value=  43 left=nullptr          right=nullptr
( 1) node=00000217DE76F2B0 value=  59 left=00000217DE76F190 right=00000217DE76F910
( 2) node=00000217DE76F910 value=  75 left=nullptr          right=00000217DE76F310
( 3) node=00000217DE76F310 value=  99 left=nullptr          right=nullptr

Command: R
Enter value to remove from in tree: 43
FOUND    node=00000217DE76F190 value=      43 left=nullptr          right=nullptr
DELETING node=00000217DE76F190 value=      43 left=nullptr          right=nullptr
AFTER    delete operation: nodePtr=0000000000000000
Command: H
```

Sample Input / Output

```
Height of tree = 3
Command: D
root=00000217DE76FBB0
( 2) node=00000217DE76FD90 value=   7 left=nullptr          right=nullptr
( 1) node=00000217DE76FA90 value=  19 left=00000217DE76FD90 right=nullptr
( 0) node=00000217DE76FBB0 value=  31 left=00000217DE76FA90 right=00000217DE76F2B0
( 1) node=00000217DE76F2B0 value=  59 left=nullptr          right=00000217DE76F910
( 2) node=00000217DE76F910 value=  75 left=nullptr          right=00000217DE76F310
( 3) node=00000217DE76F310 value=  99 left=nullptr          right=nullptr

Command: R
Enter value to remove from in tree: 31
FOUND    node=00000217DE76FBB0 value=    31 left=00000217DE76FA90 right=00000217DE76F2B0
DELETING node=00000217DE76FBB0 value=    31 left=00000217DE76FA90 right=00000217DE76F2B0
AFTER    delete operation: nodePtr=00000217DE76F2B0
Command: H
Height of tree = 2
Command: D
root=00000217DE76F2B0
( 2) node=00000217DE76FD90 value=   7 left=nullptr          right=nullptr
( 1) node=00000217DE76FA90 value=  19 left=00000217DE76FD90 right=nullptr
( 0) node=00000217DE76F2B0 value=  59 left=00000217DE76FA90 right=00000217DE76F910
( 1) node=00000217DE76F910 value=  75 left=nullptr          right=00000217DE76F310
( 2) node=00000217DE76F310 value=  99 left=nullptr          right=nullptr

Command: C
root=00000217DE76F2B0 (before clobber operation)
Are you sure you want to delete the entire tree? y
DELETING node=00000217DE76FD90 value=       7 left=nullptr          right=nullptr
DELETING node=00000217DE76FA90 value=      19 left=nullptr          right=nullptr
DELETING node=00000217DE76F310 value=      99 left=nullptr          right=nullptr
DELETING node=00000217DE76F910 value=      75 left=nullptr          right=nullptr
DELETING node=00000217DE76F2B0 value=      59 left=nullptr          right=nullptr
root=0000000000000000 (after clobber operation)

Command: r
Enter name of input file: tree02.txt
13 lines read from input file.

Command: d
     INorder traversal:
root=00000217DE76FDF0
node=00000217DE76FC70 value=       7 left=nullptr          right=nullptr
node=00000217DE76F190 value=      19 left=00000217DE76FC70 right=00000217DE76F1F0
node=00000217DE76F1F0 value=      25 left=nullptr          right=nullptr
node=00000217DE76FDF0 value=      31 left=00000217DE76F190 right=00000217DE76FD30
node=00000217DE76F970 value=      33 left=nullptr          right=nullptr
node=00000217DE76FC10 value=      43 left=00000217DE76F970 right=00000217DE76F610
node=00000217DE76F610 value=      46 left=nullptr          right=nullptr
node=00000217DE76FD30 value=      59 left=00000217DE76FC10 right=00000217DE76F910
node=00000217DE76FAF0 value=      65 left=nullptr          right=nullptr
```

Sample Input / Output

```
node=00000217DE76F910 value=      78 left=00000217DE76FAF0 right=00000217DE76F430
node=00000217DE76F7F0 value=      88 left=nullptr          right=nullptr
node=00000217DE76F430 value=      98 left=00000217DE76F7F0 right=00000217DE76F310
node=00000217DE76F310 value=      99 left=nullptr          right=nullptr

Command: pre
     PREorder traversal:
root=00000217DE76FDF0
node=00000217DE76FDF0 value=      31 left=00000217DE76F190 right=00000217DE76FD30
node=00000217DE76F190 value=      19 left=00000217DE76FC70 right=00000217DE76F1F0
node=00000217DE76FC70 value=       7 left=nullptr          right=nullptr
node=00000217DE76F1F0 value=      25 left=nullptr          right=nullptr
node=00000217DE76FD30 value=      59 left=00000217DE76FC10 right=00000217DE76F910
node=00000217DE76FC10 value=      43 left=00000217DE76F970 right=00000217DE76F610
node=00000217DE76F970 value=      33 left=nullptr          right=nullptr
node=00000217DE76F610 value=      46 left=nullptr          right=nullptr
node=00000217DE76F910 value=      78 left=00000217DE76FAF0 right=00000217DE76F430
node=00000217DE76FAF0 value=      65 left=nullptr          right=nullptr
node=00000217DE76F430 value=      98 left=00000217DE76F7F0 right=00000217DE76F310
node=00000217DE76F7F0 value=      88 left=nullptr          right=nullptr
node=00000217DE76F310 value=      99 left=nullptr          right=nullptr

Command: post
     POSTorder traversal:
root=00000217DE76FDF0
node=00000217DE76FC70 value=       7 left=nullptr          right=nullptr
node=00000217DE76F1F0 value=      25 left=nullptr          right=nullptr
node=00000217DE76F190 value=      19 left=00000217DE76FC70 right=00000217DE76F1F0
node=00000217DE76F970 value=      33 left=nullptr          right=nullptr
node=00000217DE76F610 value=      46 left=nullptr          right=nullptr
node=00000217DE76FC10 value=      43 left=00000217DE76F970 right=00000217DE76F610
node=00000217DE76FAF0 value=      65 left=nullptr          right=nullptr
node=00000217DE76F7F0 value=      88 left=nullptr          right=nullptr
node=00000217DE76F310 value=      99 left=nullptr          right=nullptr
node=00000217DE76F430 value=      98 left=00000217DE76F7F0 right=00000217DE76F310
node=00000217DE76F910 value=      78 left=00000217DE76FAF0 right=00000217DE76F430
node=00000217DE76FD30 value=      59 left=00000217DE76FC10 right=00000217DE76F910
node=00000217DE76FDF0 value=      31 left=00000217DE76F190 right=00000217DE76FD30

Command: H
Height of tree = 4
Command: D
root=00000217DE76FDF0
( 2) node=00000217DE76FC70 value=   7 left=nullptr          right=nullptr
( 1) node=00000217DE76F190 value=  19 left=00000217DE76FC70 right=00000217DE76F1F0
( 2) node=00000217DE76F1F0 value=  25 left=nullptr          right=nullptr
( 0) node=00000217DE76FDF0 value=  31 left=00000217DE76F190 right=00000217DE76FD30
( 3) node=00000217DE76F970 value=  33 left=nullptr          right=nullptr
( 2) node=00000217DE76FC10 value=  43 left=00000217DE76F970 right=00000217DE76F610
( 3) node=00000217DE76F610 value=  46 left=nullptr          right=nullptr
( 1) node=00000217DE76FD30 value=  59 left=00000217DE76FC10 right=00000217DE76F910
( 3) node=00000217DE76FAF0 value=  65 left=nullptr          right=nullptr
( 2) node=00000217DE76F910 value=  78 left=00000217DE76FAF0 right=00000217DE76F430
( 4) node=00000217DE76F7F0 value=  88 left=nullptr          right=nullptr
```

Sample Input / Output

```
( 3) node=00000217DE76F430 value=  98 left=00000217DE76F7F0 right=00000217DE76F310
( 4) node=00000217DE76F310 value=  99 left=nullptr          right=nullptr

Command: s
Enter value to search for in tree: 44
node is NOT present in tree

Command: s
Enter value to search for in tree: 88
node is present in tree

Command: C
root=00000217DE76FDF0 (before clobber operation)
Are you sure you want to delete the entire tree? y
DELETING node=00000217DE76FC70 value=       7 left=nullptr          right=nullptr
DELETING node=00000217DE76F1F0 value=      25 left=nullptr          right=nullptr
DELETING node=00000217DE76F190 value=      19 left=nullptr          right=nullptr
DELETING node=00000217DE76F970 value=      33 left=nullptr          right=nullptr
DELETING node=00000217DE76F610 value=      46 left=nullptr          right=nullptr
DELETING node=00000217DE76FC10 value=      43 left=nullptr          right=nullptr
DELETING node=00000217DE76FAF0 value=      65 left=nullptr          right=nullptr
DELETING node=00000217DE76F7F0 value=      88 left=nullptr          right=nullptr
DELETING node=00000217DE76F310 value=      99 left=nullptr          right=nullptr
DELETING node=00000217DE76F430 value=      98 left=nullptr          right=nullptr
DELETING node=00000217DE76F910 value=      78 left=nullptr          right=nullptr
DELETING node=00000217DE76FD30 value=      59 left=nullptr          right=nullptr
DELETING node=00000217DE76FDF0 value=      31 left=nullptr          right=nullptr
root=0000000000000000 (after clobber operation)

Command: r
Enter name of input file: tree03.txt
5 lines read from input file.

Command: d
     INorder traversal:
root=00000217DE76FB50
node=00000217DE76FB50 value=       1 left=nullptr          right=00000217DE76F190
node=00000217DE76F190 value=       2 left=nullptr          right=00000217DE76F490
node=00000217DE76F490 value=       3 left=nullptr          right=00000217DE76F370
node=00000217DE76F370 value=       4 left=nullptr          right=00000217DE76F0D0
node=00000217DE76F0D0 value=       5 left=nullptr          right=nullptr

Command: pre
     PREorder traversal:
root=00000217DE76FB50
node=00000217DE76FB50 value=       1 left=nullptr          right=00000217DE76F190
node=00000217DE76F190 value=       2 left=nullptr          right=00000217DE76F490
node=00000217DE76F490 value=       3 left=nullptr          right=00000217DE76F370
node=00000217DE76F370 value=       4 left=nullptr          right=00000217DE76F0D0
node=00000217DE76F0D0 value=       5 left=nullptr          right=nullptr
```

Sample Input / Output

```
Command: post
     POSTorder traversal:
root=00000217DE76FB50
node=00000217DE76F0D0 value=       5 left=nullptr          right=nullptr
node=00000217DE76F370 value=       4 left=nullptr          right=00000217DE76F0D0
node=00000217DE76F490 value=       3 left=nullptr          right=00000217DE76F370
node=00000217DE76F190 value=       2 left=nullptr          right=00000217DE76F490
node=00000217DE76FB50 value=       1 left=nullptr          right=00000217DE76F190

Command: H
Height of tree = 4
Command: D
root=00000217DE76FB50
( 0) node=00000217DE76FB50 value=   1 left=nullptr  right=00000217DE76F190
( 1) node=00000217DE76F190 value=   2 left=nullptr  right=00000217DE76F490
( 2) node=00000217DE76F490 value=   3 left=nullptr  right=00000217DE76F370
( 3) node=00000217DE76F370 value=   4 left=nullptr  right=00000217DE76F0D0
( 4) node=00000217DE76F0D0 value=   5 left=nullptr  right=nullptr

Command: q
Exiting program with status = 0
```