CS-212: LAB 8 – Binary Trees

Learning Objective:

- Creating a Binary Search Tree (BST)
- Traverse a BST with three algorithms (Pre / In / Post – Order)

This lab has three parts:

- Part-A: Creating your header file.
- Part-B: Writing the functions

Lab 8 Setup:

1. Open a new terminal window to SSH into your account on the [bingweb.binghamton.com] server. When logged in, make sure you are in your home directory by typing the following command if you are not already there. The "~" is just a shortcut symbol that means home directory:
   cd ~

2. Change your current directory to the "CS212" directory by typing:
   cd CS212

3. Confirm that you are in the "CS212" directory by typing:
   pwd

4. Create a directory for Lab8.
   mkdir Lab8

5. Move into the new directory
   cd Lab8

6. Create a new c file for you to write your lab.
   > <Last Name><FirstInitial>_Lab8.c
   > <Last Name><FirstInitial>_Lab8_Functions.c
   > <Last Name><FirstInitial>_Lab8.h

Introduction:

In this lab you will read integers in from an input file and create binary search tree with numbers. You have three output files. One output file for everything done with a PreOrder Traverse, one output file for everything done with an InOrder Traverse, and one output file for everything done with a PostOrder traverse.

NOTE: For this lab you will want to have reviewed the sample program on Brightspace and the Playbook Notes on how to delete a node.

PART A: CREATING YOUR HEADER FILE

Inside your header file you should include:

- Safeguards
- Include statements
- Constants / Macros
- Structure definitions (*see below for additional information*)
- Prototypes

Setting up your structures:

Create a structure recordType and include the following information:

- An integer

Create your node and include the following information:

- An instance of your recordType
- Node pointer to move left and right

PART B: WRITING THE FUNCTIONS

Function 1-3: PrintHeader, CenterString, PrintDivder

- These functions can be copied into your lab. All functions must be capable of sending their output to an output file or the screen.

Function 4 & 5: OpenFiles and CloseFiles

- These functions will open and close your input and output files.

Function 6: ReadDataFromInputFile

- This data will read one integer at a time. Each number will be used to create a new node and that node will be inserted into the binary search tree.

Function 7: CreateNode

- This function will initialize the member of a new node

Function 8: TreeIsEmpty

- This function will true if the tree is empty, otherwise it will return false

Function 9: InsertNode

- This function will insert your node into your tree

Function 10: PrintTreePreOrder

- This function will take a file pointer and the root of a tree

Function 11: PrintTreeInOrder

- This function will take a file pointer and the root of a tree

Function 12: PrintTreePostOrder

- This function will take a file pointer and the root of a tree

NOTE: For functions 13 and 14 you may assume that all numbers are between 1 and 100.

Function 13: FindMinValuePreOrder

- This function must use a PreOrder Traverse to find the minimum number in the tree

Function 14: FindMaxValueInOrder

- This function must use an InOrder Traverse to find the maximum number in the tree

Function 15: FindAverageValuePostOrder

- This function must us a PostOrder Traverse to find the sum of the numbers and the total number of nodes. The average will be calculated outside of function.

FUNCTION 16-18: Print the results from the previous three functions.

- Each one of these functions should take a char array that will be the header for the output
- When printing the average, only print 2 decimals.

NOTE: Deleting a node (functions 19 to 24) will be the most challenging part of this lab. Use the Playbook Notes on Brightspace.

FUNCTION 19: SearchTree

- This function will establish the location for a current pointer and a parent pointer

FUNCTION 20: DeleteLeafNode

- This function will delete a node that is a leaf node.

FUNCTION 21: DeleteNodeWithLeftChildOnly

- This function will delete a node that has only a left child.

FUNCTION 22: DeleteNodeWithRightChildOnly

- This function will delete a node that has only a right child

FUNCTION 23: DeleteNodeWithTwoChildren

- This function will delete a node that has two children

FUNCTION 24: DeleteNode

- This function will call SearchTree and use the pointers to determine which functions (functions 19-23) need to be called.

FUNCTION 25: FreeNodes

- Write a while-loop that will execute until your root is NULL. Inside the loop call DeleteNode.

PART C: WRITING THE MAIN

- In your main, call the functions:
- Declare your root node for your tree
- Open all your files
- Print your header to the screen and each output file.
- Read the data in from an input file.
- Print the list to your three output files. (Print the header for the output from the main function, use the PrintDivider and CenterString functions)
- Find the min, max and average of the tree.
- Print the three results to their respective output file. (Header for the output should include the words “Before Deletions”)
- Use the DeleteNode function to remove the following numbers from the tree: min, max, 5,15, 28, 48, 37, 31 (NOTE: One of these numbers is not in the list, that is intended)
- Re- Print the list to your three output files. (Print the header for the output from the main function)
- Find the min, max and the average of the tree for a second time.
- Print the three results to their respective output file. (Header for the output should include the words “After Deletions”)
- Free all the nodes from the tree.
- After you free the nodes, do an if-statement to check that the root is NULL, and print a message to the screen stating that the “List is Free!”.
- Close your files

RUNNING YOUR PROGRAM

Your program must run in gcc on the unix operating system with 0 errors and 0 warnings.

To run your program:

```
gcc *.c -Wall
```

This will create an output file named a.out

Valgrind must run with 0 errors and 0 warnings.

To run Valgrind:

```
valgrind --leak-check=full ./a.out
```

FILES TO SUBMIT TO BRIGHTSPACE

- <Last Name><First Initial>_Lab8.c
- <Last Name><First Initial>_Lab8_Functions.c
- <Last Name><First Initial>_Lab8.h
- PreOrderOutput.txt
- InOrderOutput.txt
- PostOrderOutput.txt

ADDITIONAL PARTIAL CREDIT GRADING RUBRIC:

NOTE: The full rubric is on Brightspace under Misc

Infraction	Points Lost
Not checking for divide by 0	-10 points

Note to Grader: Any partial credit outside of the rubric must be approved by the teacher.