BST ADT, Webgraphviz

Summary: In this lab you will develop a program that interactively builds a BST from user input, then outputs the visual representation of the the resulting tree using the drawing software Graphviz (short for Graph Visualization Software). ALL FILES NEEDED ARE ABOVE. There is a framework with sample files (lab4_files.tgz), a document on Graphviz (dotguide.pdf), and a sample running of the **working** program you will find the in the framework (initial_program_output.txt).
Compile the program as follows: $g++ -W -Wall -Werror *.cc
Run the program as follows: $a.out 1

Graphviz Dot files - Graphs (A BST is a directed graph more generally or "digraph") represented in .dot file and can be displayed in the WebGraphviz viewer online at http://webgraphviz.com/ .

Below are several sample dot files. Visually inspect each, by looking at a particular text file, and entering its contents into the WebGraphiz vieweronline. This is done by copy-paste into the textbox on the website, then click on the "Generate Graph!" button. If successful, you will see a graphical image-based representation of the inputted file. You should try this now with the four sample files provided to you in the lab4_files.tgz. Also, be sure to run the 5 samples on the Webgraphiz webpage, ("Sample 1," "Sample 2," ...).

These files can be found in the lab4_files.tgz

- simple_bst.dot - This file contains the representation for a BST with node labels that indicate each node's key. This is the most basic format for producing a BST using Graphviz. The identifier for each node is that node's label, for example "key25" represents that the node has a key value of 25. This file is statically created (not at run-time) and is for illustrative purposes only.
- height_color.dot - This file contains the representation for a BST with node labels that indicate each node's key and the node's height in the tree. This file is statically created (not at run-time) and is for illustrative purposes only.
- preorder_color.dot - This file contains the representation for a BST with node labels that indicate each node's key and the node's order in which it is visited in an preorder traversal of the tree. This file is statically created (not at run-time) and is for illustrative purposes only.
- inorder_color.dot - This file contains the representation for a BST with node labels that indicate each node's key and the node's order in which it is visited in an inorder traversal of the tree. This file is statically created (not at run-time) and is for illustrative purposes only.

There is additional information on the *.dot format in the dotguide.pdf.

Program

The BST class is composed of BinaryNodes. Insert, remove, and a useful inorder display function (which displays a BST instance to the console) have already been implemented for you. You should compile and run the program now, inspect the code, and get a feel for how it functions. A good plan of attack would be to insert several ints into a bst, next display that bst, then remove some ints and display the resulting bst.

The program is run from the command line in the following manner, where a.out is the exectable.

```
$ a.out mode
```

mode is a command line argument from the user on the specific format your program should output to the dotty viewer. For example if mode == height the program outputs a height decorated bst, if mode == preorder the program outputs a preorder decorated bst, etc. The command line argument mode is already read into a string variable inside source code provided.

The program accepts the following commands when run in the terminal.

- insert x - Inserts the integer x into the BST. You need to add code to the insert function that outputs an error message if x is already in the tree.
- remove x - Removes the integer x from the BST. You need to add code to the remove function that outputs an error message if x is NOT in the tree.
- display - Displays the current BST to the console in an inorder manner. Use this to help debug your .dot producing functions (described below).
- dotty - Traverses the BST currently in memory and writes to output.dot. This .dot file's format is determined by the user, via the command line argument mode (see below).
- end - Terminates the program.

You can read about what it means for the Node struct's functions to be static here: https://msdn.microsoft.com/en-us/library/y5f6w579.aspx

Demo: Demo your working code for your TA.
Submission: Submit your working code as lab4_part1.tgz.

Rubric-
20 pts Attendance (On-time)
80 pts Perfect
-15 pts Some errors/bugs
-15 pts Incomplete/significant errors
(End Lab 4 - Part 1)
(Start Lab 4 - Part 2)

Summary: In this lab you will extend your BST program from lab 5 to have the following functionality.

- mode == height - Write a method that writes a .dot file that contains the representation for a BST with each node labeled with the node's key and the node's height in the tree. The .dot file your program produces should look similar to height_color.dot.
- mode == preorder - Write a method that writes a .dot file that contains the representation for a BST with each node labeled with the node's key and the node's order in which it is visited in a preorder traversal of the tree. Assume the first node visited order is 1. The .dot file your program produces should look similar to preorder_color.dot.
- mode == inorder - Write a method that writes a .dot file that contains the representation for a BST with each node labeled with the node's key and the node's order in which it is visited in an inorder traversal of the tree. Assume the first node visited order is 1. The .dot file your program produces should look similar to inorder_color.dot.
- mode == postorder - Write a method that writes a .dot file that contains the representation for a BST with each node labeled with the node's key and the node's order in which it is visited in a postorder traversal of the tree. Assume the first node visited order is 1. The .dot file should look similar your preorder and postorder dotty files, but represent the postorder traversal of a BST.
- Add a command to your program that allows the user to pick whether a preorder, inorder, postorder, or height dotty file is generated and displayed during the running of the program.
- Add a destructor to the BST class.

Demo: Demo your working code for your TA.
Submission: Submit your work as lab5.tgz via iLearn.

Rubric-
20 pts Attendance (On-time)
80 pts Perfect
-15 pts Some errors/bugs
-15 pts Incomplete/significant errors