ASSIGNMENT #5

Purpose

The purpose of this lab is to continuing practicing creating dynamic structures. You will be creating a templated Binary Search Tree.

Instructions

In this lab you will create classes to implement a templated Binary Search Tree. Due to the way templates are compiled you should write all of your function definitions in the **`.h`** files of the corresponding classes.

Class 1 - BSTreeT

Your Binary Search Tree to hold templated nodes. Note that this class must be templated.

Private Data Members: Root pointer to the first node.
Size to hold how many nodes are in your tree.

Constructor: Initializes root to **`NULL`** and your node count to **`0`**.

Destructor: Calls the **`Clear`** Function

Member Function 1: Named **`GetSize`**. Returns the number of nodes in the tree. **`const`** function.

Member Function 2: Named **`Clear`**. Deletes all nodes in the tree. (Frees the Memory)

Member Function 3: Named **`Insert`**. Has a parameter which is the value to add. If the value is already in the tree just increment that node's count. Returns the count of the value in the node. i.e. returns **`1`** the first time the value is added to the tree. Returns **`2`** the second time the value is added, **`3`** the third time, etc.

Member Function 4: Named **`Exists`**. Searches your tree for a particular value. If found returns **`true`**, else returns **`false`**.

Member Function 5: Named **`Remove`**. Searches your tree for a particular value and decrements the corresponding count. If the current count is **`1`** remove the node instead of decrementing the count. Returns the current count of the value in the tree, **`0`** if it was just removed, and **`-1`** if it doesn't exist in the tree.

Member Function 6: Named **`Get`**. Searches for a particular value. Returns a pointer to the node if found otherwise returns **`NULL`**.

Member Function 7: Named **`ToStringForwards`**. Outputs the contents of the nodes in the tree (in order) as a comma separated list ("hello", "goodbye", etc). If the list is empty return the empty string.

Member Function 8: Named **`ToStringBackwards`**. Outputs the contents of the nodes in the tree (in backwards order) as a comma separated list ("goodbye", "hello", etc). If the list is empty return the empty string.

Private Member Functions: Helper functions for **`Clear`**, **`Insert`**, **`Exists`**, **`Remove`**, **`Get`**, **`ToStringForwards`** and **`ToStringBackwards`**.

Class 2 – BSTNodeT

Models information about a single node in the **`BSTreeT`**

Private Data Members:
A templated data type for the contents of the node
An unsigned int to hold a count of how many times the contents has been added
A pointer to the left child
A pointer to the right child

Constructor #1:
No parameters
Sets the templated contents to its default
Sets the count to 0
Sets the left child to **`NULL`**
Sets the right child to **`NULL`**

Constructor #2:
One templated parameter for contents
Sets the templated contents to the parameter
Sets the count to 0
Sets the left child to **`NULL`**
Sets the right child to **`NULL`**

Destructor: Sets left and right child to **`NULL`**

Mutator 1: Named **SetContents**. Has one templated parameter. Sets the contents to the given parameter's value.

Mutator 2: Named **SetLeft**. Has one parameter which is a pointer to a **BSTNodeT**. Sets the internal left pointer to the given parameter's value.

Mutator 3: Named **SetRight**. Has one parameter which is a pointer to a **BSTNodeT**. Sets the internal right pointer to the given parameter's value.

Mutator 4: Named **IncrementCount**. Increments the count by 1.

Mutator 5: Named **DecrementCount**. Decrements the count by 1.

Accessor 1: Named **GetContents**. Returns the value of the templated contents. **const** function.

Accessor 2: Named **GetLeft**. Returns a pointer by reference to the left node.

Accessor 3: Named **GetRight**. Returns a pointer by reference to the right node.

Accessor 4: Named **GetCount**. Returns the count.

Objectives

- Implement and perform a complete set of operations on a templated Binary Search Tree
- Improve skills of memory management and pointer syntax
- Practice building templated classes

Requirements

Your code must follow the styling and documenting guidelines presented in class. Please note that I do not give points for style and documentation. You can only lose points. Please make sure your source code is documented correctly and is neatly and consistently formatted using guidelines provided in class.

Your program must provide the features described above:

[50 pts] - Fully complete **`BSTNodeT`**

[100 pts] - Fully complete **`BSTreeT`**

IF YOUR PROGRAM DOES NOT COMPILE YOU WILL RECEIVE A ZERO!!!

Deliverables

Commit/Push your files to GitHub. You should submit the following two files and the unit test:

- **`bst_nodet.h`**
- **`bs_treet.h`**