Coding 05: Binary Search Tree

In this assignment you will create a Binary Search Tree object that will work with a struct data type, that contains an int called id and a string called data (similar to the stack assignment).

Create a Binary Search Tree class as discussed in class. The class will contain all the data and methods to have a complete working and proper Binary Search Tree object. Your Binary Search Tree Object must conform to the following guidelines:

- The object must be completely self contained and fully functional without any outside “helper” functions.
- The object may not print to the console or contain any other I/O except parameters passed in and return values passed out, EXCEPT for the traverse functions (see below).
- You must use a “linked list” approach.
- Your tree has to be capable of growing to any size.
- The tree itself is an ordered “list” of pointers to structs.

Your program should start with a number 0-20 passed in from the command line that will dictate the number of nodes in the tree (you must do error checking on the user input to ensure a number 0-20). The Tree will “fill up” with with that many pointers to the struct objects that you create to “fill” the tree (0 is an empty tree). The id of each struct is a random number from 100 to 999 *and* unique.

To create your tree use the following process:

- Pass in a number 0-20 from the command line to define the total nodes in the tree.
- Create structs and assign a unique random number 100-999 to the id and anything you like to the string. Pass the pointer to that struct to the tree. -XOR- Pass the data (id and string) to the Tree and let the Tree create the struct and pointer.
- Show your tree works by thoroughly testing it from main( ).

Your Tree must have the following functionality:

- bool isEmpty()
- int getHeight()
- int getNumberOfNodes()
- /*something*/ getHead()
- /*something*/ getEntry(int) /*pass in an id to get*/
- bool add(/*something*/)
- bool remove(int) /*pass in an id to remove*/
- bool clear()
- bool contains(int) /*pass the id you want to test*/
- void preorderTraverse() /*this method may print*/
- void inorderTraverse() /*this method may print*/
- void postorderTraverse() /*this method may print*/

Do not implement the getRootData or setRootData as public methods. If you want to implement them make them as private methods to call as a special case of add(), remove(), and getEntry()

Test your program thoroughly. Main is your test area, do whatever you like in main() to prove your Tree works *fully*. Test all functionality above and account for edge cases like trying to remove from an empty tree or “item not found,” etc. Part of your grade is showing you can create a thorough set of test cases.

All good programming practices apply.