Programming Assignment #4
CS 260 Data Structures

Please follow the style sheet for submission requirements.

Programming – Goal: The goal of this program is to create a binary search tree (BST) and to implement the BST algorithms **recursively.**

Background: The advantage of a binary search tree is the ability to retrieve our data using a logarithmic performance assuming that the tree is relatively balanced and be able to search for a range of information and obtain our data in sorted order. In program #4 we will experience all of these characteristics. **Instead of using a hash table**, we will use close to the same data as Program #3 but this time with a binary search tree. **PLEASE NOTE – we are REPLACING the concept of a hash table with a binary search tree when moving from program #3 to program #4.**

Specifics: In Program #4 we will use a BST to search for a website. **You will be basing this program off of the same data as in program #3 but now using a DIFFERENT data structure!** We are moving away from hash tables and learning about BSTs. **The use of external data files is required**.

Each item in our tree will have the following information (at a minimum) which should be stored as a struct or a class:

1. Topic name (e.g., "Data Structures")
2. **New: keyword** that is part of the website address that can be used for sorting/ searching (e.g., Carrano-Data-Abstraction)
3. Website address
   (e.g., https://www.pearson.com/us/higher-education/product/Carrano-Data-Abstraction-Problem-Solving-with-C-5th-Edition/9780321433329.html)
4. Summary of what you can find at this address (e.g., "The classic, best-selling Data Abstraction and Problem Solving with **C++**: Walls and Mirrors book provides a firm foundation in data structures")
5. Review (e.g., your thoughts about how helpful this site is)
6. Rating (1-5 stars – 1 being not very useful, 5 being very useful)

The **ADT operations that must be performed on this data are:**

1) **Constructor** – initialize all data members
2) **Destructor** – deallocate (release) all dynamic memory and reset the data members to their zero equivalent value; this should call a recursive function that performs postorder traversal (recursively) to deallocate all data and nodes.
3) **Insert** a new website based on the keyword in the website address. For example, in the above it would be: Carrano-Data-Abstraction
4) **Remove** all matches to a topic name
5) **Remove** a particular website based on keyword (only one match)
6) **Retrieve** the information about a particular website (based on its keyword)
    a. Remember, retrieve is NOT a display function and should supply the matching information back to the calling routine through the argument list
7) **Display all** websites (**sorted alphabetically by keyword!**).

Data Structures: Write a C++ program that implements **a binary search tree**. The binary search tree should be a non-linear implementation (using left and right pointers). The underlying data may be stored as a struct or class.

Evaluate the performance of storing and retrieving items from this tree. Monitor the height of the tree and determine how that relates to the number of items. If the number of items is 100 and the height is 90, we know that we do not have a relatively balanced tree!! Use the information from the Carrano reading to assist in determine if we have a reasonable tree, or not. **Your efficiency write up must discuss what you have discovered.**

<u>Things you should know...as part of your program:</u>

1) Do not use statically allocated arrays in your classes or structures. All memory must be dynamically allocated and kept to a minimum!
2) All data members in a class must be private
3) Never perform input operations from your data structure class in CS260
4) Global variables are not allowed in CS260
5) **Do not use the String class! (use arrays of characters instead and the cstring library!)**
6) Use modular design, separating the .h files from the .cpp files. Remember, .h files should contain the class header and any necessary

prototypes. The .cpp files should contain function definitions. You must have at least 1 .h file and 2 .cpp files. **Never "#include" .cpp files!**

7) Use the iostream library for all I/O; do not use stdio.h.

8) Make sure to define a constructor and destructor for your class. Your destructor must deallocate all dynamically allocated memory.