CS 474: Object Oriented Programming Languages and Environments

Fall 2022

First C++ Project

Due time: 11:59 pm on Sunday 11/13/2022

Implement a simple *string set* calculator in C++11 or subsequent version. The calculator implements sets of strings as binary search trees (BSTs) while exposing a command line interface. The calculator maintain two sets of strings at any point in time named S1 and S2. The calculator supports common set operations by responding to commands listed below. *You must comply with the implementation requirements listed below for full credit.*

Recall that BSTs are binary trees subject to three conditions. First, a string value is associated with each tree node. Second, each node can have at most two children, a left child and a right child. Third, given a node x, the values of all nodes in the left subtree of x are less than the value of x, and the values of all nodes in the right subtree of x are greater than the value of x. Use lexicographical ordering to compare strings. (You are encouraged to use the ASCII code of each string character to determine the value of that character.) No duplicate values will be allowed in your sets.

Implementation requirements. You are required to comply with the following guidelines when implementing your string set calculator.

1. Define your own *String* class for strings contained in the BSTs. The class should have only two data members *size_* and *chars_* of type *int* and *char** respectively. The class should implement, among other member functions, a copy constructor—performing a deep copy—and a destructor to avoid memory leaks. Other members functions should be defined as needed, possibly including an assignment operator consistent with the copy constructor and the destructor.

2. Sets are implemented by a class named *BST*. You are required to implement the copy constructor and the destructor for *BST*. The copy constructor must perform a deep copy; the destructor must delete all dynamically-allocated objects used exclusively by the receiver. Other methods could be added as needed to implement the functionality below.

3. You are not required to balance instances of class *BST*

Your command line interface will execute a loop prompting the user for a command, executing the command, and then printing the content of S1 and S2. Here is a list of commands. Make sure not to cause any memory leaks or dangling pointers in the implementation of these commands.

`e` – Erase set. This command allows interactive users to delete the current S1 set. The previous value stored in S1 is lost forever.

`s` – Switch sets. The sets associated with S1 and S2 are swapped, meaning that S1 will receive the previous S2 set and vice versa.

`b` – Subset operation. This command compares S1 and S2 and prints an appropriate message on the console indicating whether S2 is a proper subset of S2. Two strings are deemed to be equal if they have the same length and content; however, the comparison is case sensitive. S1 and S2 are not modified

`c` – Copy set. Set S1 is deep copied into S2. The previous content of S2 is deleted. The content of S1 is not affected. The two sets must not share any data structures, that is, they can be modified independently of each other.

`l` – List set contents. The string values stored in the two sets are printed on the standard output stream in alphabetical order. The two sets are not modified.

`a` – Add element. This function allows a user to add a new string to S1. The value is obtained through an appropriate prompt with an interactive user. No action is taken if the string in question is already in the set. The insertion should preserve the BST properties of S1.

`u` – Union. This element takes the set union of S1 and S2 and stores the resulting value in S1. The previous content of S1 is lost. S2 is not modified by this operation.

`i` – Intersection. This command takes the set intersection of S1and S2 and stores the resulting value in S1. The previous content of S1 is overriden. S2 is not modified by this operation.

`q` – Prints sets S1 and S2 and quits the set manager.

You must work alone on this project. Save all your code in a collection of header and code files and submit a zip archive with a (short) readme file containing instructions on how to use your Set Calculator. Submit the archive by clicking on the link provided with this assignment. Your code should compile under the GNU C++ compiler. You can use C-style strings, the library class `string`, or your own defined class to implement strings. Make sure to split the implementation of all your classes between a header and a code file. No late submissions will be accepted.