Semester Project Part 3: A BST of tickets and user ids

Data Structures and Analysis of Algorithms, akk5

Objectives

- To strengthen student's knowledge of C++ programming
- To give student experience reading and parsing strings of commands
- To give student experience in writing a non-linear data structure
- To give the student experience implementing a BST

Instructions

For this assignment you must write a program that implements and manages two BSTs. Each BST should store a set of user id – ticket number pairs and will represent one "view" on that data set. The first BST should store the set of data by ticket number, we will refer to this BST as ticket; the second BST should store the set of data by user id, we will refer to this BST as user. Unless indicated in the description of the commands below, every operation performed on the ticket BST should be performed on the user BST and vice versa.

Your program should implement a command prompt (text-based interface) capable of handling the following commands:

exit – exits the program

load *<file>* - parses the contents of the file as if they were entered from the command prompt.

display *<BST>* in – displays the contents of the specified ***BST*** in order

display *<BST>* pre – displays the contents of the specified ***BST*** in pre order

display *<BST>* post – displays the contents of the specified ***BST*** in post order

new – deletes both BSTs and replaces them with empty BSTs.

remove *<user id> <ticket>* - Removes the specified ***user id – ticket number*** pair from both BSTs. Should report failure if the pairing doesn't exist.

find *<user id> <ticket>* - Searches the BSTs for the specified ***user id – ticket number*** pair and displays whether they were found or not.

add *<user id> <ticket>* - Inserts the specified ***user id – ticket number*** pair into both BSTs. Should report failure if the pairing exists.

save *<BST>* into *<file>* - saves the post order traversal of the specified ***BST*** into a text file named ***file***.

Note: ***<file>, <user id>,*** and ***<BST>*** are strings but will not contain spaces.

<ticket> is an integer. ***<BST> = {user, ticket}***

Guidance

There are few details for this project you should be careful with. First, BST ticket and BST user use different parts of the user id – ticket number pair as their primary value; this will necessitate implementing a BST for each "view." This can be done by coding two separate BST classes; savvy students will realize that you can accomplish the same result by templating your BST.

Second, as detailed in the lecture, BSTs don't support duplicate data; care must be taken to implement your comparisons between user id – ticket number pairs carefully; one user id can have multiple tickets assigned to it and vice versa. You will need to work out a way to support this; perhaps you can write a comparison function that tests the primary value first and then tests the secondary value if the primary values are equal.

This is an individual assignment. Seeking direct help from students, tutors, and websites such as chegg or stack overflow will be construed as a violation of the honor code.

Grading Breakdown

Point Breakdown	
Structure	12 pts
The program has a header comment with the required information.	3 pts
The overall readability of the program.	3 pts
Program uses separate files for main and class definitions	3 pts
Program includes meaningful comments	3 pts
Syntax	18 pts
Implements the Node class/classes correctly	9 pts
Implements the BST class/classes correctly	9 pts
Behavior	70 pts
Program handles all command inputs properly	
• Exit the program	7 pts
• In order traversal	7 pts
• Pre order traversal	7 pts
• Post order traversal	7 pts
• Load a file	7 pts
• Find a uid, ticket pair in the BSTs	7 pts
• Remove a uid, ticket pair from the BSTs	7 pts
• Insert a uid, ticket pair into the BSTs	7 pts
• Delete both BSTs and create empty ones	7 pts
• Save the post order traversal for the specified BST into a given file	7 pts
Total Possible Points	**100pts**
Penalties	
Program does NOT compile	-100
Late up to 72 hrs	-10 / day
Late more than 72hrs	-100

This is an individual assignment. Seeking direct help from students, tutors, and websites such as chegg or stack overflow will be construed as a violation of the honor code.

Header Comment

At the top of each program, type in the following comment:

/*

Student Name: <student name>

Student NetID: <student NetID>

Compiler Used: <Visual Studio, GCC, etc.>

Program Description:

<Write a short description of the program.>

*/

Example:

/*

Student Name: John Smith

Student NetID: jjjs123

Compiler Used: Eclipse using MinGW

Program Description:

This program prints lots and lots of strings!!

*/

Assignment Information

Due Date:

Files Expected:

This project should require the implementation of one or more classes as well as the function necessary for parsing the various inputs. At a minimum I am expecting BST.cpp, BST.h, main.cpp; you might have more files depending upon your implementation.