COMP2102 Data Structures and Algorithms
Fall 2022, Project #2
Due Date: 11 December 2022, 23:59

This is a pair (group project) assignment. Students should form groups of two people. Both of them are responsible for the whole project.

Do not share your code with another groups or individuals. The grade will be 1 pts for all parties.

Do not copy-paste online codes.

No doubt, you can discuss with people around, but you cannot get help on code or blocks of code from anyone, including relatives.

You are only allowed to use your lecture or lab section notes. Violating this rule will result in 1 as Project #2 grade.

You must demonstrate your project on declared dates. Otherwise, your project grade will be 2 pts.

One (random) person from the pair (group) will be called for demo. The grade will be depend on his/her answers. It is not possible to select or change the person called for demo. It is not possible to help the person during the demo.

If demo student fails to answer half or more of the questions, your project grade will be 1 pts for both students.

Late submissions will not be accepted!

Binary Search Tree Map based Music Song record system

In this project, you will create a record system that allows the insertion, search, and deletion of musical records. The initial musical records will be read from a text file (D:\\songs.txt) where each line holds one song information as follows.

Song name;Artist;ID;Genre;Year

Implementation Details: Each field has been separated by semicolon(;) from other fields in songs.txt. Firstly each musical record should be copied to an array and then the index of the array containing that record together with the ID field (for the first tree) or song name field (for the second tree) or Artist field (for the third tree) should be inserted separately to three different AVL trees. In other words, Each record will be stored in an array element with all fileds (Song name;Artist;ID;Genre;Year), but the same record will be inserted into one of the three AVL trees with (index, ID), (index, song name), or (index, Artist) fields, where index shows the element of array containing the same record with all fields. The key values in the above mentined trees would be ID, or song name or Artist respectively for the 1st, 2nd and the 3rd trees. The intuition behind this is not to store the same information triple times.

Example.,

İki keklik;Erkan Oğur;1003;Türk Halk;1996

You will prepare your songs.txt file by yourself.

More details:

1. Firstly, build a song class. Then create a song object for each song read from the D:\\songs.txt and then add it to array and then inset it into three different trees according to three fields: song name, ID, and artist, as explained already.
2. Your code should be able to insert, search and delete to/in an AVL tree. Insertion is done only at first, when the text file is being read, but search and delete operations should be done in running time after asking the user what (s)he wants to do, or according to which filed the user wants to search for or delete. Moreover, there should be three types of search operations (methods) which are explained below, but only one delete method is enough. Note that as the output of the search methods, all fields of the found record should be printed appropriately, if the searched method was not found, another appropriate message should be printed.
3. **The first search method** should be able to search songs using the name, artist, or ID fields. For example, the user should be able to search for a song with ID = 1003, or name = "Summer Time". This Search will be done based on only one field, not a combination of fields.
4. **The second search method** allows the user to search and display all songs of a given genre. For example: list all "Türk Halk" songs or "Rock" songs.
5. **The third search method** will let the users search ranges by giving lower and upper bounds of an interval for keys (only the ID field); For example, ID: 1001-1010 displays the list of users with IDs from 1001 to 1010.
6. Note that after finding the index of a searched record in AVL tree, you can access all fields of that record in the array and pring it for the user.
7. In delete opertion, you will delete a record from all trees but no need to delete it from the array.
8. Let the user to delete an existing record based only on its ID field. No need to implement delete operation based on other fields.
9. You can design your interactive menu by yourself.
10. To test your program, read a few records from a file, say D\\Songs.txt and build the tree, then apply the tasks mentioned above on the generated tree.

NOTE: You are allowed to use all code samples or methods used in the lectures and lab sections but copying any part of the code from Internet or getting it from someone is forbidden and will result in score 1 for project 2.

Good luck, Comp 2112 team, Fall 2022