CSCI 2170 Open Lab 7

This program aims to write a C++ program that uses the ***Sorted List class (Linked list implementation)*** to maintain the top hit songs. The songs should be kept in ascending order based on the artists' names.

The header file and implementation files of the sorted list class are named **slist.h** and **slist.cpp**. The main program is named **songs.cpp**.

First, your program reads from a data file named "topsongs.dat" and build a list of songs. Each song in this data file is described by four information: (1) the rank of the song at the end of the year, (2) the name of the artist, (3) the title of the song, and (4) the year the song debut. An example song record is shown below:

1	← number 1 ranked song
Katy Perry	← artist name
Dark Horse	← song title
2014	← billboard year

Use type.h and type.cpp to define a struct for the song type.

Copy the data file **topsongs.dat** into your project folder.

To allow the user to interact with the program, display a menu as the following. The program should allow the user to continually interact with the program by selecting menu choices '1' – '4' repeatedly. The program only terminates when the user selects the menu choice '5'.

Billboard Top Song Management

Please select from the following menu choices:

1. Look up an artist
2. Add a song
3. Delete a song
4. Print
5. Exit

- If the user selects choice '1', (s)he should be prompted to enter the name of the artist. The program then displays ALL the songs by this artist that have appeared in the billboard charts in the database. If the artist does not have any top songs in the database, an error message should be displayed;

  Display information about the songs in tabular format using **setw(number).** For example if the name of the artist entered is: **Beyonce**, an example output of the program is as the following:

Here are the songs by Beyonce

Title	**Rank**	**Year**
Drunk in Love	**22**	**2015**
Partition	**72**	**2015**
Best Thing I Never Had	**57**	**2012**
Countdown	**100**	**2012**

- If the user selects choice '2', (s)he should be prompted to enter the information about: (1) the title; (2) the artist name, (3) the rank, and (4) the year of a **new song**. This new song record should then be added to list in the correct sorted location.
- If the user selects choice '3', (s)he should be prompted to enter the title of the song to be deleted. The program then deletes the song from the database. If no such song is found, display an error message.
- If the user selects choice '4', (s)he should be prompted to enter the year value. The program displays all the top songs in that year. The output should include: (1) artist name, (2) title of the song, and (3) the ranking of the song.
- If the user selects '5', the program terminates.

Programming requirements

The **Sorted List class** with linked list implementation should be used for this program. **Do not add any additional method to the *Sorted List class (Linked list implementation)* discussed in class.**

1. Define a struct type to store information of a song record. Use typedef to nake "ListItemType" an alias of this type.
2. In the client program, write separate user defined function for each of the following:

    a. Display a menu
    b. Read the information from a data file and build a list
    c. Prompt the user for the name of an artist and displays the songs by this artist
    d. Prompt the user for the title of a song and deletes the song
    e. Prompt the user for the year value and displays all the top songs in that year

Documentation requirements for programs involving Abstract Data Type

For a client to use one ADT type, (s)he only needs to look at the specification file to find out what methods are available, and what each methods do. There is no need to know the details of how each method is implemented. Therefore for the client of the ADT, it is essential that the specification file is a well documented about how to use this ADT.

All the description for the methods should be kept in the specification program (aka in the header file). This means for each method in the class, a description of what this method does need to be placed before the declaration statement of that method, inside the class definition. Please refer to the example code discussed in class for example on how to write the description.

For the implementation file/program, you are not required to place description in front of the definitions of individual methods. For each logically related statement, for example a loop, a decision statement, or a group of statements for achieving a coherent task, a description needs to be given to explain what that group of statement does.

For the client program, the documentation requirements we had before applies. https://www.cs.mtsu.edu/~cen/2170/private/ola/programrequirements.pdf.

Compile and build the project

You are required to create a ***makefile*** for this program.

Submit the following files for this Open Lab:

type.h type.cpp slist.h slist.cpp songs.cpp makefile