CptS **122 – Data Structures**

WASHINGTON STATE UNIVERSITY
World Class. Face to Face.

Programming Assignment 3: Digital Music Manager & Doubly Linked Lists – Part II

Assigned: Wednesday, September 13, 2017
Due: Friday, September 22, 2017 by midnight

In this assignment you will complete the Digital Music Manager that you started in PA 2. You must implement the following features:

- (4) insert
- (5) delete
- (7) sort
- (10) shuffle

➢ *What must "insert" do?*
The "insert" command must prompt the user for the details of a new *record*. The prompt must request the artist name, album title, song title, genre, song length, number of times played, and rating. The new record must be *inserted* at the *front* of the list.

➢ *What must "delete" do?*
The "delete" command must prompt the user for a *song title*, and *remove* the matching record from the list. If the song title does *not* exist, then the list remains unchanged.

➢ *What must "sort" do?*
The "sort" command must prompt the user for 4 different methods to sort the *records* in the list. These include:

1. Sort based on artist (A-Z)
2. Sort based on album title (A-Z)
3. Sort based on rating (1-5)
4. Sort based on times played (largest-smallest)

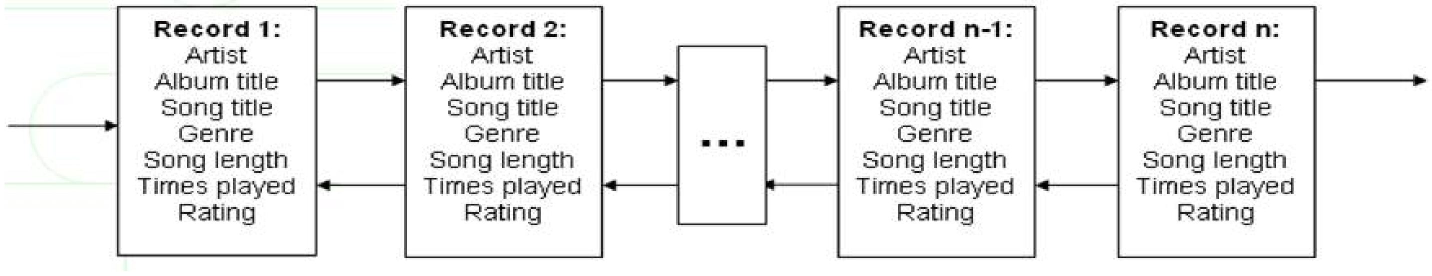
Once a sort method is selected by the user, the sort must be performed on the records in the list. Consider using bubble sort, insertion sort, or selection sort.

➢ *What must "shuffle" do?*
The "shuffle" command must provide a random order in which the songs are played. This command must not modify the links in the list. It must just specify the order in which songs are played, based on the position of the song in the list. For example, let's say we have a list with 5 songs at positions 1 – 5 in the list, shuffle must generate an order 1 – 5 in which the songs are played. An order 2, 5, 3, 1, 4 would require that the second song in the list is played first, the fifth song in the list is played second, the third song in the list is played third, the first song in the list is played fourth, and the fourth song in the list is played fifth. The songs are accessed by traversing the list both forwards and backwards to satisfy the order. Hence, the need for a doubly linked list!

Once again you will find an example `musicPlayList.csv` (here).

IV. Logical Block Diagram

Once again, the logical block diagram for your doubly linked list should look like the following:



As you can see from the illustration a doubly linked list has a pointer to the next node and the previous node in the list. The first node's previous node pointer is always NULL and the last node's next pointer is always NULL. When you insert and delete nodes from a doubly linked list, you must always carefully link the previous and next pointers.

BONUS:

Modify your doubly linked list implementation(s) for your DMM so that last node in the list points to the first node, and the first node points to the last node. Hence, there is no longer a first or last node. This list is now called "circular". Overall, it is called a circular doubly linked list. Any one of the nodes may by the current node!