**Program #1**

**CS 260 Data Structures**

**Please follow submission instructions in style sheet.**

**GENERAL INFORMATION about PROGRAMS in CS260:**

**Scope:** For each project, we only have approximately 2 weeks to complete the write-ups and code. Therefore, it is critical that you focus on a limited scope, with an emphasis on the class(es) and data structures, rather than building a comprehensive application. Each program this term will have you create an Abstract Data Type (ADT), **which is what you will be primarily graded on**. *This means that we will primarily grade your projects based on the use of classes, member functions, arguments, data structures, pointers, efficiency, and how well the software demonstrates that it has been fully tested. The code needs to compile and run, and include a test program for the member functions provided.* **It is not appropriate to hard code in the test cases - all tests should be interactive with the user.** Your user interface must be clear for us to thoroughly test all features.

**Class construct**: Most programming assignments this term requires only two classes, the data class and the data structure class. You can make the user interface a class as well if you choose to.

**Design is the First Step:** The first step will be to design your software. This should include designing classes that are well structured and that efficiently use the data structure. **The use of recursion** should be included **at least once** in each assignment to better prepare us for future programs. Avoid multiple traversals whenever possible. *Take a look at the style sheet for instruction on the topics that your write-ups need to cover.*

**Creating Test Plans:** Part of the first step will be to develop a test plan. This includes planning out each of the cases that need to be implemented and subsequently tested. It is important to focus on thoroughly testing your class member functions.

**BACKGROUND INFORMATION for Program #1:**

This first program is an exercise in building, traversing, and destroying linear linked lists.

**Programming Assignment:** For the first programming assignment, we are going to create a program that helps keeping track of song lists.

Your primary job will be to create an ADT for Song and an ADT for SongList

Part I: The Song ADT

The **information** about a song should include:

Artist name (e.g., Eminem)

Title (e.g., Lose Yourself)

Length (e.g., 5.23)

Number of likes (e.g., 654,000)

Part II: The SongList ADT

The data members for SongList should be a head pointer to a linear linked list of Song objects and the number of songs in the list. The songs should be organized by popularities with the most popular song as the first node in the list.

This ADT must have **public member** functions to perform the following:

1. Constructor - Construct an object and initialize the data members
2. Destructor - Release all dynamic memory and reset data members to their zero equivalent value
3. Add a new song
4. Edit the number of likes for a song
5. Display all songs in the list
6. Display all songs for an artist (in order of popularity)
7. Remove all songs with fewer than M likes, where M is sent in as an argument

Part III: The driver or the test program

The test program needs to first load the test data set from external file at the beginning of the program.

The menu-based user interface should allow user to use/test ALL the functionalities of the program. Try to make the user interface easier to use.

1. Always prompt user when you need input data.
2. The prompt needs to be meaningful. Example works great. E.g. “Enter the minimum likes (e.g 1000): “
3. When asking user to choose some existing data, index works great. You can display the data with index preceding each one first.

**Things you should know...as part of your program:**

**General Syntax Rules**

1. Do not use statically allocated arrays in your classes or structures. All memory must be dynamically allocated and kept to a minimum for the classes! You can only use statically allocated arrays for temporary storage, such as local variables.
2. Global variables are not allowed in CS260; global constants are encouraged
3. **Do not use the String class! (use cstring, arrays of characters instead);** you may use the cstring library with strlen, strcpy, strcmp etc
4. Avoid using return from within the body of a loop
5. Avoid using while(1) type of syntax
6. Strive towards structured programming techniques
7. Remember that at least one function needs to be written using recursion!
8. For functions that have non-void return types, make sure to return a value through each path through the function.
9. NEVER pass class types by value (always by reference); NEVER return class types by value. If you don’t want the objects modified, put const there.
10. Use the iostream library for all I/O; do not use stdio.h for I/O
11. Use of external data files is required
12. MOST IMPORTANT:

BACKUP your files before creating a TAR archive!!!!

**General rules for building ADTs**

1. The ADTs should be designed to “hide” the data structure as part of the private section of the class. This means your head pointer to a node should be a private data member in your ADT.
2. All data members in a class must be private
3. Public member functions should NEVER have a node pointer as an argument (Remember client programs should not be involved with the physical data structure when working with ADTs)
4. Never prompt and read from the user when inside a class member function; this should be done in the “client” program when developing ADTs
5. Never output error messages from a class member function; this should also be done in the “client” program when developing ADTs
6. Each public member function should have a way to communicate success or failure back to the “client” program. This can be done with a returned value, and argument passed by reference, or exception handling.
7. Use modular design, separating the .h files from the .cpp files. Remember, .h files should contain the class header and any necessary prototypes. The .cpp files should contain function definitions. You must have at least 1 .h file and 2 .cpp files. **Never implement functions in your .h file! And, never "#include" .cpp files!**
8. NEVER archive your writeups. Your writeups need to be uploaded on D2L as separate documents.