APS 105 — Computer Fundamentals

Lab 9: Linked Lists

Winter 2019

In this lab, you will write a program that maintains information about your personal music library using a Linked List data structure. The program will allow you to add and delete entries from your personal music library, to search your personal music library for songs by song name, and to print out the entire list of your library.

This lab will be due in the week of April 1.

Your Personal Music Library

Your task is to write a complete C program, named `Lab9.c`, to maintain information about your personal music library. The data in your personal music library will be stored in memory with the use of a linked list, with one list node per song. Each node will contain three strings: a song's name, its artist, and its genre (the type of music). The linked list must be kept in sorted alphabetical order, by song name, beginning with A through Z (i.e. increasing alphabetical order). No two songs in your personal music library should have the same name.

Your program should be "menu" driven, with the user being offered a choice of the following five "commands":

- **Command** `I`. Insert a new song into the library. The program should prompt the user for a new song name, its artist's name, and its genre. This information must be placed in a new node that has been created using the `malloc` function (to be clear, you *must* use malloc for this purpose). This node should then be inserted at the appropriate (alphabetical) position in the linked list. Don't forget that the music library must be stored in increasing order, by song name. If a node with the given song name is already in the music library, an error message should be output, and the new node should not be inserted into the linked list.

- **Command** `D`. Delete an entry from your the library. The program should prompt the user for the name of the song to be deleted, and then find and delete the node containing that song name from the library. If no node with the given song name is found, an error message should be output. All memory allocated for a deleted entry must be released back to the system using the `free` function. This includes not only the memory allocated for the node, but also the strings in the node that would have been separately allocated.

- **Command** `S`. Search for a user supplied song name in the library. The program should print the name, artist, and genre of the song, with each piece of information on a separate line. If no node with the given song name is found, an error message should be output.

- **Command** `P`. Print your personal music library, in alphabetical order by song name. Print the song name, artist, and genre of each song, each on a separate line. A blank line should be printed between each song.

- **Command** `Q`. Quit the program. When the program is given the `Q` command, it should delete all of the nodes in the linked list, *including* all the strings contained

in each node. Deletion means both removing from the list, but *also* freeing all dynamically allocated memory using call the `free` function. It should then print the (what should be an empty) linked list.

To assist you in the production of your program, we have provided you with a file, `musiclibrary.c`, that contains part of the complete program. This program is provided on the course website along with this lab. This "skeleton" of the lab 9 program includes all of the C statements required to implement the menu driven parts of the program. It also includes a few helpful functions for reading data and printing messages. You should take this file and edit it to become your version of `Lab9.c`. Note, however, that you may ***not*** change any of the code in the existing implementation of the skeleton program, except where indicated in comments. In particular, you must use the `inputStringFromUser()` function and the prompts provided to obtain inputs from the user, and you must use the given Node structure.

In addition we strongly recommend that you do your work for this lab in the following way:

- Read the entire skeleton program carefully. Take note of the provided functions for reading strings, printing the name, artist and genre of a song, and for printing error messages. Using these functions will make it easier for you to satisfy the exercise and marker programs.
- Add the function for inserting a new node (the `I` command) into the linked list. Your function will need to read the name, artist, and genre of a song. Test your program by trying to insert nodes into the linked list. Try to insert nodes with both new and duplicate song names.
- Add a function for printing the linked list (the `P` command). Test your program by inserting songs into the linked list and then printing them out. Are the entries in the correct order?
- Add a function that searches the linked list for a given *song name* and then either prints the appropriate song or, if a node is not found, prints an error message. This is the `S` command.
- Add the statements that need to be executed when the `Q` command is entered. These statements should delete the linked list by using calls to the `free` function. To check your work, print the linked list after the elements have been deleted.
- Add a function for deleting a song from the personal music library. It will need to search the linked list for a given song name, delete the appropriate node from the linked list, and then use the `free` function to release the memory used to store the node, as well as all the memory that the node uses for storing strings. If the given song name is not found in the music library, print an error message.

We recommend that you test your program after attempting to complete each step. This way, if your program no longer works, you will know which statements are causing the error. Complete each step before moving on to the next one.

Sample Output From Executing The Program

Here is a sample output from an execution of the program that you are to prepare.

```
Personal Music Library.

Commands are I (insert), D (delete), S (search by song name),
P (print), Q (quit).

Command --> P

The music library is empty.

Command --> I
Song name --> The Shade
Artist --> Metric
Genre --> Rock

Command --> I
Song name --> Heads Will Roll
Artist --> Yeah Yeah Yeahs
Genre --> Punk

Command --> I
Song name --> Bad Boys Need Love Too
Artist --> Bahamas (Afie Jurvanen)
Genre --> Folk

Command --> P

My Personal Music Library:

Bad Boys Need Love Too
Bahamas (Afie Jurvanen)
Folk

Heads Will Roll
Yeah Yeah Yeahs
Punk

The Shade
Metric
Rock

Command --> I
Song name --> Heads Will Roll
Artist --> Yeah Yeah Yeahs
Genre --> Punk

A song with the name 'Heads Will Roll' is already in the music library.
No new song entered.

Command --> I
Song name --> Adult Diversion
```

```
Artist --> Alvvays
Genre --> Pop

Command --> P

My Personal Music Library:

Adult Diversion
Alvvays
Pop

Bad Boys Need Love Too
Bahamas (Afie Jurvanen)
Folk

Heads Will Roll
Yeah Yeah Yeahs
Punk

The Shade
Metric
Rock

Command --> S

Enter the name of the song to search for --> Bad Boys Need Love Too

The song name 'Bad Boys Need Love Too' was found in the music library.

Bad Boys Need Love Too
Bahamas (Afie Jurvanen)
Folk

Command --> S

Enter the name of the song to search for --> Young Blood

The song name 'Young Blood' was not found in the music library.

Command --> D

Enter the name of the song to be deleted --> The Shade

Deleting a song with name 'The Shade' from the music library.

Command --> P

My Personal Music Library:

Adult Diversion
```

```
Alvvays
Pop

Bad Boys Need Love Too
Bahamas (Afie Jurvanen)
Folk

Heads Will Roll
Yeah Yeah Yeahs
Punk

Command --> Q

Deleting a song with name 'Adult Diversion' from the music library.

Deleting a song with name 'Bad Boys Need Love Too' from the music library.

Deleting a song with name 'Heads Will Roll' from the music library.

The music library is empty.
```

Grading by TA and Submitting Your Program for Auto-Marking

There are a total of **10 marks** available in this lab, marked in two different ways:

- **By your TA, for 4 marks out of 10.** Once you are ready, show your program to your TA so that it can be marked for style (1 mark), and to ask you a few questions to test your understanding of what is happening (3 marks). Programs with good style have been described in previous labs, so that will not be repeated here. Be ready to explain your approach to Part 2 to your TA.
- **By an auto-marking program for 6 marks out of 10**. You must submit your program, named `Lab9.c`, through the ECF computers for marking. We will use a software program to compile and run your programs, and test them with different inputs.

Similar to the previous labs, long before you submit your program for marking, you should run the **exercise** program that compiles and runs your program and give it sample inputs, and checks that the outputs are correct. You should run the following command:

`/share/copy/aps105s/lab9/exercise`

within the directory that contains your solution programs.

This program will look for the file `Lab9.c` in your directory, compile it, and run it on *some* of the test cases that will be used to mark your program automatically later. If there is anything wrong, the **exercise** program will report this to you, so read its output carefully, and fix the errors that it reports.

Since this lab uses dynamic memory allocation and pointer variables extensively, you may easily run into run-time errors (such as segmentation faults). Please note that run-time errors are treated as failures when the **exercise** and the **marker** programs are comparing the output of your program with the standard output.

Once you have determined that your program is as correct as you can make it, then you must submit your program for auto-marking. This must be done by the end of your lab period as that is the due time. To do so, go into the directory containing your solution files and type the following command:

```
/share/copy/aps105s/lab9/submit
```

This command will re-run the exercise program to check that everything looks fine. If it finds a problem, it will ask you if you are sure that you want to submit. Note that you may submit your work as many times as you want prior to the deadline; only the most recent submission is marked.

Important Note: You must submit your lab by the end of your assigned lab period. Late submissions will not be accepted, and you will receive a grade of zero.

You can also check to see if what you think you have submitted is actually there, for peace of mind, using the following command:

```
/share/copy/aps105s/lab9/viewsubmitted
```

This command will download into the directory you run it in, a copy of the file that has been submitted. If you already have a file of that same name in your directory, that file will be renamed with a number added to the end of the filename.

After the Final Deadline — Obtaining Automark

Briefly after all lab sections have finished, you will be able to run the automarker to determine the automarked fraction of your grade on the code you have submitted. To do so, run the following command:

```
/share/copy/aps105s/lab9/marker
```

This command will compile and run your code, and test it with all of the test cases used to determine the automark grade. You will be able to see those test cases' output and what went right or wrong.