I. Introduction of the Program.

This is a Music Record Library. This program basically will help you in recording songs and some of the information about the song, namely, the title, the artist, composer, album, genre, your own rating and also your remarks about the song). It is easy to use and also very convenient in many ways. It has mainly 4 functions. First is that the program can add songs information. Just enter the title and fill up the other fields. (which is optional - skipping through that field means that it is either empty or in default). Second, the program can also update information of the songs that were updated in the library. Just input the title (be careful for it is case-sensitive) and fill up the other fields. Third, the program can also list songs according to the input query you chose, and of course, in a sorted manner. Lastly, there's the exit function which is needed for freeing memory-allocations and updating the library to the record file. Don't worry for the program validates input and is flexible to changes.

II. Functions used and its uses

1.) char *remove_spaces(char string[]) = This function removes unnecessary spaces made by the user such as leading and trailing spaces.

2.) int check_string(char title[]) = This function just double checks if there are any spaces or invalid characters which are not present in most of the information for songs.

3.) int check_title(Song *head,char title[]) = Thie function checks and counts how many nodes that have "title" equal to the node's title attribute.

4.) int update_song(char title[], Song *head) = This is the function that will update program and the library if any changes in the current data is wanted. Returns 1 if the title of the song found and 0 if not.

5.) int check_action(char string[]) = This function will validate the action done in the main program(Choosing the library function to use).

6.) Song *add_song(Song* tail, int n) = This function is basically the add song function which is literally used for adding songs in the library. The head of the null is needed and also for the numbers of nodes in the list. Returns the pointer to the added song node. The process are almost identical to each other. Will ask for input,make the last eleent equal to null then validates it, continues if correct and loops if it is wrong.

7.)Song *unsorted_list(Song* tail, Song* current) = This function is like the add song function but only contains nodes selected nodes. The fields are filled by copying the field from the current's field. Returns the pointer of the node added.

8.) void exit_library(FILE *fp, Song* top) = This function is the exit library function. Tt's purpose is to update the library and freeing the memory allocation happened during the program.

9.) int check_substring (char string1[], char string2[]) = This function checks the two strings if they are equal to each other regardless of the case. Return 1 if yes and 0 if no.

10.) int check_substring2(char string[], char string2[]) = This function checks the two strings if they are equal considering the case. Returns 1 if no and 0 if yes.

11.) int check_equal(char string1[], char string2[]) = This function checks two strings if they are equal while also without considering the case of the characters of the string..

Returnd 1 if they are equal and 0 if no.

12.) Song* sort_list(Song *tail, int n) = This function sorts the list according to the title field of the nodes in alphabetical order.If ever the two nodes have the same song title, they will be sorted acoording to the artist field.
13.) Song* sort_list2(Song *tail, int n) = This function is exclusively for the sorting of nodes based on rating in the list song function. It will sort the list in decreasing order, 1 as the lowest and 5 as the highest.
14.) Song* list_song(Song* top) = This function is the list song function. It will have the head node of the list as its argument. It will basically ask for a choice to what songs will be listed according to the field. It will output the nodes that satisfied the condition in sorted manner. It returns head node but returns listhead instead if "All" was chosen for it is the head of the sorted list.

III. User-defined data types

typedef struct song_info{ - This is the structure for the song which will hold the necessary information and data.
int song_ID; - Randomized number which is given by the program itself
char title[25]; - This attribute holds the title of the song (Cannot be empty)
char artist[25]; - This attribute holds the artist of the song (Can be empty and "No Artist" is allowed).
char composer[25]; - This attribute holds the composer of the song (Can be empty and "No Composer" is allowed)
char album[25]; - This attribute holds the album of the song (It's default value if "Single")
char genre[25]; - This attribute holds the genre of the song (Only restricted to Art Music, Popular Art and Traditional Music)
int rating; - This attribute holds the rating of the song (Values can only range from 0 to 5 as 0 being the lowest and 5 as the highest)
char remarks[50]; - This attribute holds the title of the song
struct song_info *next; - Points to next node (traverses towards NULL)
struct song_info *previous; - Points to previous node (traverses away NULL)
} Song; - The name of my user-define data type for struct song_info.

IV. Daily Log

Day 1 (November 20, 2015)

-I started filling up the main function. I started building and planning the interface of my program to reduce and prevent confusion in reading and understanding the program in execution.
-Problems encounter:
>No problems encountered except for the design of the programs.
>I attempted using the gotoxy function under the windows.h library but the problem is there is a possibility that the one checking the program will not use windows but instead, Linux or Mac which will conflict to the compilation and execution of the program.
*SOLUTION : Less text and more use of the remaining white space without being OS and Compiler dependent.

----***----

Day 2 (November 23, 2015)

-I started doing test runs and experiments in file manipulation - reading, writing and appending.
-I tried doing appending as my way of updating the library but have encountered problems - having NULL between the data of the original file and the file appended. I failed in finding the solution for this problem, how to update the library and manipulate the files.

----***----

Day 3 (November 27, 2015)
-After consulting with an upperclassmen, I have grasped a part of knowledge in manipulations and why things are not working out. Also, I found the solution in updating the library.

*SOLUTION - First read the file then run the program. Before terminating the program, overwrite the file with the updates in the program.

-I experimented and made many test runs until it became successful. (Mostly the problems is what and how to use the functions for files)

----***----

DAY 4 (November 28, 2015)
-I started doing the Add Song library function.
-I made test runs in another source file that is why I didn't encounter much problem in this exact source file.
-Problems encountered:

>I had a problem in outputting random numbers for the Song ID of a song.
It either gets the same series of numbers or repeats the same number in the local function.

*SOLUTION : I put the "randomization" process outside the local function but inside the main function. After that, It outputted random numbers without problems.

> The problems is how to put data in the structures correctly since I used pointers. I want to avoid wasting memory.

*SOLUTION : Because of the attributes of the structure, it is pretty inevitable to lessen the space in writing the code but I used repeated patterns for readability purposes. By allocating and resizing the memory I was able to minimize the usage of the memory.

----***----

Day 5 (November 30, 2015)
-I started doing the Update Song library function
-This part was easy because this function has the same logic of the add song function
- The only problem encounter was the allocation memory. At first, the program outputted garbage characters because I wasn't able to
notice that mistake.

*SOLUTION = allocation, resizing and also freeing if necessary memories were done throughout the whole code.

----***----

DAY 6 (November 28, 2015)
-I started doing the List song function
-It was hard at first doing this because I do not know how to swap the values/nodes in the list. It took me the whole day to come up with the solution. I first thought I can just swap the two nodes like how to swap values in number but it wasn't successful because the pointers was only overwritten with the same pointer

since it is a pointer. The values and the nodes weren't swapped. They were overwritten and became the same at the end of the sorting process (Bubble sort). I then tried to use brute force and swap the values themselves to the other nodes but it was also unsuccessful because there were many moments that the program freezes up and shows run time errors. I then tried swapping the address of the two pointers of the nodes to see if they will be swapped. But for some reason, it didn't work and I didn't know why.

*SOLUTION : For the first, second and third node, to swap the second and third node, I must point the first->next pointer to the third node. Then connect the second->next to NULL and lastly to connect the third->next to the second.

-That solution only came up in the morning actually what I was doing while coming up with the solution is fixing the documentation of this Machine problem

-But I started also filling up the list song function with input validations. The only thing that is missing is the sorting algorithm itself.

----***----

Day 7 (December 2, 2015)

- I got the logic for the sorting of the nodes but I still can't implement it through code. I came up with 4 different algorithms and all of them failed.
- I stopped working for the list song function. I started testing the current program that I made.
- Problem encountered:

>I get random run time errors in any part of the program and also get random characters for my arrays.

*SOLUTION : It took me the whole day just to fix this problem. It is because on how I use the pointers. I tried running my code through Linux/Mac and it showed the “Segmentation Fault” message. I then changed almost all of the pointers into arrays and almost change the whole program just to fix the problem. It was successful in the end.

----***----

Day 8 (December 4, 2015)

-I continued doing the list song function but the only problem is the sorting algorithm and how will the process work around and outside the algorithm. It took me again the whole day but still didn’t get the answer.

----***----

Day 9 (December 5,2015)

-I started asking the upperclassmen on how to swap nodes in a list. I generally get the same answer but I still can’t implement it in C. Until one of them introduced a new algorithm on how to swap nodes. It was hard to grasp in the start but I was able to implement the logic through code and I got it right.

-There weren’t any significant problems encountered in the since I am already 90% finished without the sorting algorithm.