Algorithm for the music playlist program
Siqi Wen 2021 CS162

1. Create a structure(struct music) with 5 members:

i.Artist(char artist[ARTIST], ARTIST is a constant number defined globally);
ii. Title of the Song(char title[TITLE], TITLE is another constant number defined globally);
iii. The type of music (char type[TYPE], TYPE is another constant number defined globally);
iv. When you like to listen to it (char when[WHEN], WHEN is another constant number defined globally);

v.The link to access (char link[LINK], LINK is another constant number defined globally) .

2. Create a function to read in one song: void read_a_song(music & a_song)

This function will take one argument — the structure will be passed by reference. In this function, prompt the user to enter the artist, title, type, when to listen and the link, use cin.get(a_song.xxx, XXX, '\n') to read in the information for the first four members in the structure; use cin for the last one(since links cannot contain spaces. Then use cin.ignore(100, '\n') afterwards.

3. Create a function to display the contents of a song: void display_a_song(music & a_song)

This function will take one argument — the structure will be passed by reference. In this function, use "cout << a_song.xxx …" to display the contents of a song.

4. Create a function to read in multiple songs: void read_songs(music library[ ], int & num)

This function takes two arguments(the structure array and the number of songs) and returns nothing (Music library[SONGS] will be created in main, SONGS is a constant number globally defined; int num will be defined in main and initialized to 0). In this function, create a character variable: char response = 'n' (initialize it to character 'n'); prompt the user by asking if they want to read in a new song, read in the answer the user types(cin >> response;) followed by "cin.ignore(100, '\n');". Then, there will be a while loop (while(toupper(response) == 'Y' && num < SONGS), in this while loop, call the read_a_song function(the argument for this function would be: library[num]); then prompt the user again by asking "Another song?", read in the answer, followed by cin.ignore(100, '\n'), then increment "num" by 1: ++num.

5. Create a function to display the songs: void display_songs(music library[ ], int num)

This function displays the contents of all songs, it takes two arguments: the structure array and the number of songs(which doesn't need to be passed by reference this time). In this function, create a for loop: for(int i=0; i<num; ++i), in this for loop, call the display_a_song function and the argument passed by will be "library[i]".

6. Create a function to return the number of songs read in: int load(music library[ ])

This function reads information from the file, it takes the structure array as an argument. In this function, create an ifstream variable called "file_in" and another variable "int i = 0", then open the file: file_in.open("song.txt"). Use "if(file_in)" to see if we connected to the file successfully. If we did, then "file_in.get(library[0].artist,

ARTIST, '|'); file_in.ignore(100, '|');" will be executed. Then create a while loop : while (!file_in.eof() && i < SONGS), in this while loop, use "file_in.get(library[i].title, TITLE, '|'); file_in.ignore(100, '|'); file_in.get(library[i].type, TYPE, '|'); file_in.ignore(100, '|'); file_in.get(library[i].when, WHEN, '|'); file_in.ignore(100, '|'); file_in >> library[i].link; file_in.ignore(100, '\n');" to get the rest information for the first song. Then increment i by 1: ++i; after this, use "file_in.get(library[i].artist, ARTIST, '|'); file_in.ignore(100, '|');" to get the beginning information for the second song. This while loop will continue until any condition fails. Use "file_in.close();" to close the file. At the end of this function, return i.

7. Create a function to save the songs read in to an external data file: void save(music library[ ], int num_to_save)

This function takes two arguments: the structure array and an integer representing the number of songs to save. In the function, create a variable: ofstream file_out; then open the file by using: file_out.open("song.txt", ios::app). Then use: if(file_out), to see if we connected to the file successfully, create a for loop: for(int i = 0; i < num_to_save; ++i), in this for loop, use "file_out" to write the content of the songs to the file: file_out << library[i].artist << '|' << library…….<< library[i].link << endl; when the for loop is done, use "file_out.close()" to close the file.

8. Create a function to display all songs that match a particular choice: void display_specific_songs(music library[ ], int num)

This function takes two arguments: the structure array and an integer for the number of songs. In this function, prompt the user to input an answer for "when" they want to listen to the songs (so that we can find the songs that match with this "when" situation); create a for loop to iterate over the array, for(int j = 0; j < num; ++j), in this for loop, use "strcmp" to compare each "library[j].when"

with the user’s input, output all the songs that have the same “when” information together as a playlist.