MP3 Player
Project 2 – CS 2308 – Spring 2016

Due: Wednesday, 2/17/16 @ noon
100 points

PROBLEM STATEMENT

Write a C++ program that will allow a user to manage the storage portion of a digital music player. Also write a test driver (a main() function) that tests the functionality of your music player.

The music player will store the following information about each song:

- Song title (e.g., “Hey Jude”: may contain spaces, must be unique)
- Artist name (e.g., “The Beatles”: may contain spaces, not unique)
- Size (the song’s size in MB; a double with value greater than 0)
- Play count (the number of times the song has been played; an integer)
- Lyrics (e.g., “Take a sad song and make it better”: one line of the song’s lyrics, may contain spaces; used in this assignment to simulate ‘playing’ the song)

Your music player should be able to store up to 25 songs and has a maximum storage capacity of 50.0 MB.

The music player supports the following operations:

1. Add a new song to the player (prompt the user for input values)
2. Remove a song from the player (prompt the user for title to remove)
3. Play a song (prompt the user for title to play)
4. Display a song’s metadata (prompt the user for title)
5. Display the full playlist ordered alphabetically by title
6. Display the player’s remaining storage capacity (in MB)

There should be a separate function associated with each of the above operations. (You may have additional helper functions if you want – and in fact, they are strongly encouraged).

For the Add Song operation, the program should prompt the user to enter a song title, an artist name, the song’s size in MB (with input validation), and a line of lyrics. A complete solution will ensure that the music player is not full (25 songs, or no storage capacity remaining for the song’s size). If the song cannot be added, the program should output a message stating why.
For the **Remove Song** operation, the program should prompt the user to enter a song title to remove. The program should indicate whether the operation was successful or not.

For the **Play Song** operation, the program should prompt the user to enter a song title to play. The program should increment the play count of the song, then output the song's name and artist followed by its lyrics. If the song is not found, the program should indicate an error. (You don't need to validate that the play count doesn’t overflow the max value of an integer).

For the **Display Metadata** operation, the program should prompt the user to enter a song title to display. It should then output the title, the artist name, the file size, and the play count of the song. If the song is not found, the program should display an appropriate error message.

For the **Display Playlist** operation, the program should display the title, artist, size, and play count for each song, with each song printed on its own line. If the music player is empty, the program should display a message stating so.

For the **Display Remaining Capacity** operation, the program should display the number of MB of remaining storage space available on the music player.

All output is to standard (console) out.

Your `main()` function should declare the variables to store the songs and track the song count and capacity. It should then call a sequence of the functions described above in order to exactly match the expected output provided in `pr2_expected_out.txt` (on the Assignments tab of the course website).

You can call the functions anything you’d like and set up the parameters and return values however you need. As an example, here’s a small segment of my `main()` function, with the arguments removed:

```plaintext
addSong(/* args */);
addSong(/* args */);
addSong(/* args */);
addSong(/* args */);
addSong(/* args */);
addSong(/* args */);
displayEmptySpace(/* args */);
addSong(/* args */);
displayPlaylist(/* args */);
```
ASSIGNMENT SPECIFICS

This assignment is to be completed individually and all submitted code must be your own individual work. Remember that I will grade your program on athena and your grade will reflect only how it compiles/runs on that machine.

Additional requirements:

• **Do not use global variables!** Global named constants are encouraged.

• Use an array of Song structures to represent the music player contents. The structure definition should be global, but the array of structures may **NOT** be global.

• When the music player contains fewer than 25 songs, all empty slots should be at the back of the array. Keep a count of how many songs are stored in the music player. The songs should always be stored in elements 0 through count-1 of the array.

• You MUST use **binary search** for all searches! Your code **may not have a linear search anywhere!**

• A good solution may end up needing **sort** code as well. You may use any of the sorts covered in class.

• Your program must be modular, with significant work done by functions. Each function should perform a single, well-defined task. **You must have a function for each of the 6 music player operations described above.** Additionally, you should use functions to avoid code duplication (e.g., don’t write the same binary search code twice – call a function!). Pay attention to which function parameters need to be references.

Because the song title and artist names may include spaces, you will need to use getline() for some input. Remember that using ‘cin >>’ in combination with getline() can cause problems because ‘cin >>’ leaves whitespace (including newlines) in the buffer, messing up the next getline() call. You can throw away this whitespace with:

```
cin >> ws;
```

*This is a long program! Get started early.*
Please follow the course style guidelines, which you can find here:
http://cs.txstate.edu/~mo1162/cs2308/docs/StyleGuide.pdf

Pay particular attention to the comments required at the top of the file and for all functions, the requirement that variables have descriptive names, and the requirement to correctly indent your code (using spaces, not tabs). **No magic constants!** Be consistent about all style decisions. **You may lose points for style issues;** however, **no more than 15% of the assignment grade will be deducted for style.**

---

**SUBMISSION INSTRUCTIONS**

At the top of the file, include a block of comments listing the name of the file, your name, the course and section number, the assignment number, and a brief description of the code’s purpose.

You must name your C++ source file as follows:

```
mp3player_AXXXXXXXX.cpp
```

where AXXXXXXXX is your TxState ID number (**not your NetID**).

**Upload your C++ source file** to TRACS. TRACS will not allow submission after the deadline. You are welcome to submit multiple times, and we will only grade your most recent submission before the deadline.

**No hard-copy submission is required for this assignment.**