**MelodyLinks**



**A Music Box Implementation with Doubly Linked Lists**

**Due Date: November 2nd 11:59pm**

The primary objective of this assignment is to delve deep into the understanding and hands-on implementation of data structures, with a focus on the doubly linked list. By completing this assignment, you will also gain practical experience with sorting algorithms, learning how they can be applied effectively in real-world scenarios. Successfully executing this project not only solidifies your foundational knowledge in computer science but also serves as a valuable addition to your professional *resume*, showcasing your ability to transform theoretical concepts into functional solutions.

**Core Features:**

1. Implement a template-based doubly linked list. The linked list should support the following operations:
   * **push\_back**: adds an item to the end of the list
   * **remove**: removes the first occurrence of the given item from the list
   * **getSize**: returns the current size of the list
   * **at**: returns the item at the given index in the list
   * **contains**: checks if the list contains a given item
   * **replace**: replaces the item in the given index with the new item

1. A music box will maintain a playlist which is implemented using a doubly linked list of Songs. The music box should be able to perform the following operations:
   * **addSong**: Add a song to the end of the playlist
   * **removeSong**: Remove a specified song from the playlist
   * **searchSong**: searches for a specified song in the playlist
   * **playNext**: advances the current position to the next song in the playlist. If the current song is the last in the playlist, it wraps around to the first song.
   * **playPrevious**: advances the current position to the previous song in the playlist. If the current song is the first in the playlist, it wraps around to the last song.
   * **currentSong**: returns the currently playing song
   * **displayPlaylist**: displays all the songs in the playlist
   * **sort**: sorts the playlist by the titles of the songs using the quick sort algorithm, choosing the initial song as the pivot point.

Remember, the key is to understand the underlying principles. Implementation becomes a lot easier when you have a clear conceptual understanding. Good luck and have fun building your MelodyLinks music box!

**Sample Run:**

Welcome to the MelodyLinks!

Menu:

1. Add song to the playlist

2. Remove song from the playlist

3. Play the next song

4. Play the previous song

5. Display current song

6. Display the entire playlist with durations

7. Search for a song

8. Exit

Choose an option: **1**

Enter song title: **Stereo Hearts**

Enter song duration (in seconds): **240**

"Shape of You" added to the playlist.

Choose an option: **1**

Enter song title: **Despacito**

Enter song duration (in seconds): **220**

"Despacito" added to the playlist.

Choose an option: **1**

Enter song title: **Believer**

Enter song duration (in seconds): **205**

"Believer" added to the playlist.

Choose an option: **5**

Now playing: "Stereo Hearts" Duration: 240 seconds

Choose an option: **3**

Now playing: "Despacito" Duration: 220 seconds

Choose an option: **4**

Now playing: "Stereo Hearts" Duration: 240 seconds

Choose an option: **6**

Playlist:

Stereo Hearts - 240 seconds

Despacito - 220 seconds

Believer - 205 seconds

Choose an option: 7

Enter song title to search for: **Despacito**

Song "Despacito" found in the playlist!

Choose an option: **2**

Enter song title to remove: **Despacito**

"Despacito" removed from the playlist.

Choose an option: **6**

Playlist:

Stereo Hearts - 240 seconds

Believer - 205 seconds

Choose an option: **8**

Thank you for using the MelodyLinks!

**Bonus Challenges (OPTIONAL)**

**[10 bonus points]** Add a shuffle feature! Enable the MelodyLinks to randomly reorder songs in the playlist.

**[10 bonus points]** Bring your MelodyLinks box to life! Incorporate a few audio tracks (e.g., Mp3) and showcase your application by playing real songs. Capture this in action and share a video recording. Remember to attach the video link in your submission.

**Submission**

You are asked to submit your work as a single zip file via CANVAS. Zip file will include all source codes (.h and .cpp files) and also include Readme.txt where you explain what you’ve learned in this assignment.

Please use the following file format while naming the zip file:

LastNameFirstnameX\_Y.zip where LastNameFirstname is your last name with the first letter in capital, followed by your first name with the first letter in capital; the X is the course code; the Y is the assignment #. (ex: SerceFatmaCS300\_3.zip)

**Important Notes**

Before you submit your work, please make sure the entire folder works. Here is a sample command that you can perform a sanity check:

*Windows Users:*

yourfolder> g++ -o myProgram.exe -std=c++14 \*.cpp  
  
yourfolder>myProgram.exe

*Mac Users:*

yourfolder$ g++ -o myProgram -std=c++14 \*.cpp  
  
yourfolder$ ./myProgram

**Evaluation Criteria**

1. *Correctness of Implementation*: All functionalities should work without any errors or bugs. Each functionality is 5 points **[14 x 5 = 70 points]**
2. *Code Organization and Structure*: The code should be well-organized, modular, and structured. Proper indentation and naming conventions should be followed. **[10 points]**
3. *Comments and Documentation*: The code should be well-commented, explaining complex logic, tricky parts, and the purpose of functions/methods. **[10 points]**
4. *Edge Case Handling*: The code should handle edge cases gracefully without crashing. For example, trying to play the next song when no songs are left or searching for a song that isn't there. **[10 points]**