Project 2: Rad.io Music

CPSC 131 Spring 2019

### Introduction

Welcome to Rad.io, you've been hired to work on our music streaming app, think of it as Spotify only more rad! You're in charge of handling our customer’s song list. When a user selects a playlist it will load into the list a number of songs. Users can skip to the next song, move to the previous, they can select a song to play next or select a song to add to the end of their list.

### Objective

You are given two header files, Song.hpp and RadList.hpp. Song holds a song’s name, artist, album, duration (in seconds) and whether or not it contains explicit lyrics. RadList keeps track of all Song objects in the linked list List. RadList also has an iterator that points to the Song object currently playing.

Complete the implementation of these classes, and make sure all tests pass. Your instructor may require you to keep your header and implementation code seperate in both .hpp and .cpp files. Your code is tested in the provided main.cpp.

You will need to implement the following functions:

* **Song Constructor** - This should initialize all Song member variables.
* **Song Getters** - name, artist, album, minutes, seconds and explicit lyrics should all be returned with these member functions. Keep in mind the duration in seconds is stored. The minutes() and seconds() getters should return the minutes and leftover seconds of a song. For example a song with a duration of 291 seconds should return 4 minutes and 51 seconds.
* **next()** - This should move the nowPlaying iterator to the next song in the list.
* **prev()** - This should move the nowPlaying iterator to the previous song in the list.
* **nowPlaying()** - This should return the current Song the nowPlaying iterator is currently pointing to.
* **addToList()** - This should add the passed in song to the end of the list.
* **playNext()** - This should add the passed in song as the next song in the list right after the current song playing.

Initially the given code will not compile. As you complete the code, the tests should start to pass in main.cpp.

### Source Code Files

You are given “skeleton” code files with declarations that may be incomplete and without any implementation. Implement the code and ensure that all the tests in main.cpp pass successfully.

* Song.hpp: This is to be completed
* RadList.hpp: This is to be completed
* main.cpp: The main function tests the output of your functions. You may wish to add additional tests. During grading your main.cpp file will be replaced with the one you were provided with.
* README.md: You must edit this file to include your name and CSUF email. This information will be used so that we can enter your grades into Titanium.

### Hints

The goal of this assignment is to make a class that wraps around a linked list class. You do not need to implement the linked list class, just use the standard library implementation. You can refer to <http://www.cplusplus.com/reference/list/list/?kw=list> for a list of methods of the std::list class.

The use of iterators is key. It may help to draw out the list with each song and trace through the list with every operation, keeping track of where the nowPlaying iterator should point.

Make sure your code compiles, and then try and solve the logic. Focus on solving one test at a time. It’s recommended that you start with Song.hpp first and then RadList.hpp next. The order of each function is the order in which it’s recommended that you solve.

You do not need to worry about error checking although it would be a good idea!

### Obtaining and submitting code

We will be using GitHub Classroom to distribute the skeleton code and collect your submissions. This requires you to have an account on github.com. If you are new to GitHub, do the following to get started:

1. Create an account at github.com. You may want to use this account to show a portfolio of your work to prospective employers in the future, so choose something professional.
2. Read Understanding the GitHub Flow and Hello World at GitHub Guides.
3. Read the instructions below for instructions on how to test.

Once you understand the basic operation of git, click the assignment link to fork your own copy of the skeleton code to your PC.

Do not fork your repository to your personal github account (instructors have admin access to private repositories under https://github.com/CSUF-CPSC-131-Spring2019/). Your code should have a URL like https://github.com/CSUF-CPSC-131-Spring2019/project1-brians, NOT https://github.com/brian/project1-brians.

<https://classroom.github.com/a/4azpGcAw>

Then edit your code locally as you develop it. As you make progress, commit and push your changes to your repository regularly. This ensures that your work is backed up, and that you will receive credit for making a submission. Don’t wait until the deadline to learn how to push code!

### Testing

Unless otherwise directed, use the following command to compile your program:  
**clang++ -g -Wall -std=c++14 main.cpp -o test**

If you included separate implementation files your command will need to include the separate implementation files as well (but never the header files):

**clang++ -g -Wall -std=c++14 main.cpp Song.cpp RadList.cpp -o test**

To attempt to run the compiled test program, use the following command:  
**./test**

### Grading rubric

Your grade will be comprised of two parts, *Form* and *Function*.

*Function* refers to whether your code works properly as tested by the main function (80%).

*Form* refers to the design, organization, and presentation of your code. An instructor will read your code and evaluate these aspects of your submission (20%).

### Deadline

The project deadline is Monday, March 4th before midnight.

You will be graded based on what you have pushed to the main branch of your GitHub repository as of the deadline. Commits made after the deadline will not be considered. Late submissions will not be accepted.

**Your code must compile/build for it to be tested and graded. If you only complete part of the project, make sure that it compiles before submitting.**