**COMP B19 - Assignment   
20 points**

**Due December 1, 2016**

*You will be building a linked list. Make sure to keep track of both the head and tail nodes.*

(1) Create three files to submit.

* Playlist.h - Class declaration (use file provided)
* Playlist.cpp - Class definition
* main.cpp - main() function

Build the PlaylistNode class per the following specifications.

* Default constructor
* Parameterized constructor
* Public member functions
  + InsertAfter()
  + SetNext() - Mutator
  + GetID() - Accessor
  + GetSongName() - Accessor
  + GetArtistName() - Accessor
  + GetSongLength() - Accessor
  + GetNext() - Accessor
  + PrintPlaylistNode()
  + OutputToFile()
* Private data members
  + string uniqueID - Initialized to "none" in default constructor
  + string songName - Initialized to "none" in default constructor
  + string artistName - Initialized to "none" in default constructor
  + int songLength - Initialized to 0 in default constructor
  + PlaylistNode\* nextNodePtr - Initialized to 0 in default constructor

Note: Some functions can initially be function stubs (empty functions), to be completed in later steps.

Ex. of PrintPlaylistNode output:

Unique ID: S123

Song Name: Peg

Artist Name: Steely Dan

Song Length (in seconds): 237

Hint: Hardcode ten to twelve records that are automatically input each time the program starts (to save having to type the entries each time you run the program). You can include these in your code or import these from a textfile. If you opt to import from a textfile you may add an addtional function declaration to your .cpp and .h files.

(2) In main(), prompt the user for the title of the playlist.  
  
Ex:

Enter playlist's title: JAMZ

(3) Implement the PrintMenu() function. PrintMenu() takes the playlist title as a parameter and outputs a menu of options to manipulate the playlist. Each option is represented by a single character. Build and output the menu within the function.

If an invalid character is entered, continue to prompt for a valid choice.

Hint: Implement Quit before implementing other options. Call PrintMenu() in the main() function. Continue to execute the menu until the user enters q to Quit.

Ex:

JAMZ PLAYLIST MENU

a - Add song

d - Remove song

c - Change position of song

s - Output songs by specific artist

t - Output total time of playlist (in seconds)

o - Output full playlist

p - Output full playlist to storage

q - Quit

Choose an option:

(4) Implement "Output full playlist" menu option. If the list is empty, output: Playlist is empty

Ex:

JAMZ - OUTPUT FULL PLAYLIST

1.

Unique ID: SD123

Song Name: Peg

Artist Name: Steely Dan

Song Length (in seconds): 237

2.

Unique ID: JJ234

Song Name: All For You

Artist Name: Janet Jackson

Song Length (in seconds): 391

3.

Unique ID: J345

Song Name: Canned Heat

Artist Name: Jamiroquai

Song Length (in seconds): 330

4.

Unique ID: JJ456

Song Name: Black Eagle

Artist Name: Janet Jackson

Song Length (in seconds): 197

5.

Unique ID: SD567

Song Name: I Got The News

Artist Name: Steely Dan

Song Length (in seconds): 306

(5) Implement the "Add song" menu item. New additions are added to the end of the list.  
  
Ex:

ADD SONG

Enter song's unique ID: SD123

Enter song's name: Peg

Enter artist's name: Steely Dan

Enter song's length (in seconds): 237

(6) Implement the "Remove song" function. Prompt the user for the unique ID of the song to be removed.

Ex:

REMOVE SONG

Enter song's unique ID: JJ234

"All For You" removed

(7) Implement the "Change position of song" menu option. Prompt the user for the current position of the song and the desired new position. Valid new positions are 1 - *n* (the number of nodes). If the user enters a new position that is less than 1, move the node to the position 1 (the head). If the user enters a new position greater than *n*, move the node to position *n* (the tail). 6 cases will be tested:

* Moving the head node
* Moving the tail node
* Moving a node to the head
* Moving a node to the tail
* Moving a node up the list
* Moving a node down the list

Ex:

CHANGE POSITION OF SONG

Enter song's current position: 3

Enter new position for song: 2

"Canned Heat" moved to position 2

(8) Implement the "Output songs by specific artist" menu option. Prompt the user for the artist's name, and output the node's information, starting with the node's current position.   
  
Ex:

OUTPUT SONGS BY SPECIFIC ARTIST

Enter artist's name: Janet Jackson

2.

Unique ID: JJ234

Song Name: All For You

Artist Name: Janet Jackson

Song Length (in seconds): 391

4.

Unique ID: JJ456

Song Name: Black Eagle

Artist Name: Janet Jackson

Song Length (in seconds): 197

(9) Implement the "Output total time of playlist" menu option. Output the sum of the time of the playlist's songs (in seconds).   
  
Ex:

OUTPUT TOTAL TIME OF PLAYLIST (IN SECONDS)

Total time: 1461 seconds

(10) Output the playlist to an text file.

**Submissions Requirements**

You are encouraged but not required to work with a partner on this assignment.

Include the following as a comment at the top of your code:

/\*   
Author: Lastname, Firstname  
Partner: Lastname, Firstname

Assignment: Programming Assignment Four

Course: COMP B12  
Instructor: Richard Miles  
Date submitted: 2016/12/01  
\*/

**Playlist.h**

#ifndef PLAYLIST\_H

#define PLAYLIST\_H

#include <string>

using namespace std;

class PlaylistNode {

public:

PlaylistNode();

PlaylistNode(string initID, string initSongName, string initArtistName,   
 int initSongLength, PlaylistNode\* nextLoc = 0);

void InsertAfter(PlaylistNode\* nodePtr);

void SetNext(PlaylistNode\* nodePtr);

string GetID() const;

string GetSongName() const;

string GetArtistName() const;

int GetSongLength() const;

PlaylistNode\* GetNext();

void PrintPlaylistNode();  
 void OutputToFile();

private:

string uniqueID;

string songName;

string artistName;

int songLength;

PlaylistNode\* nextNodePtr;

};

#endif