Homework for Dynamic Variables and Memory Management

# Introduction

You must complete homework assignments in order to be assessed, but homework does not directly affect your final course grade in any way. Homework provides an opportunity for you to practice the application of course concepts, and is an essential part of the learning process.

You may re-submit homework as many times as desired. You can ask for help as often as you need while you create your programs.

You must receive a grade of "Meets" on the homework before you can be assessed on this topic. This homework will be graded using the rubric for Dynamic Variables and Memory Management.

Homework for this module is entirely on Linux. There is no zyBooks component.

## Linux Memory Exercises

### Goals for This Project:

## More about class and OOP

## Using pointers and dynamic memory

## Focusing on dynamic arrays.

## Using memory leak detection tool

### Problem You Need to Solve for This Project (Same as HW 5):

## You are asked to write an app to keep track of a relatively small music library. The app should load song information from a data file once the app is started. It should allow the user to view, add, remove, and search for songs. The app should save the data back to the same data file when the program exits.

### What Your Program Should Do (Same as Project 5):

### Write an interactive text based menu interface (using a loop) that will allow the user to:

### Enter information for a new song

### Display information for all the songs in the database with index for each song

### Remove a song by index

### Search for songs by a certain artist

### Search for songs by a certain album

### Quit

### For each song, you need to keep track of:

### Title Artist Duration album

### Allow the program to keep looping until the user wants to quit. When the program starts, it should load the tasks from an external file ("songs.txt") into memory. When a user enters information about the new song, the program needs to read them in, save them in memory and eventually write them to the external data file ("songs.txt").

### The file format could look like:

### title;artist;duration;album title;artist;duration;album

### The ';' is used as a delimiter or field separator. Each record ends with a new line character.

### Some Implementation Requirements:

1. Use structs or class named Songto model song.
2. You really should think about changing the struct to class if you have not done so in Lab 2.
3. Use class named SongListto model the collection of songs.
4. Use dynamic array of Songto implement SongList.
5. Use dynamic character array to model the strings inSong, such as title and artist. The character array should be the exact size as needed, e.g "CS162" should use a character array of size 6 including '\0'.
6. Use destructor to deallocate the dynamic memory for the object.
7. Make sure your program is "memory-leak-free" by using valgrind valgrind --tool=memcheck --leak-check=fullexecutable-file.
8. When using class, please make sure you encapsulate the data which means make all the instance data members private and provide accessor methods and mutator methods to access and manipulate the data.
9. For submission, your data file should contain a sufficient set of test data. It should have test cases for the same artist with multiple songs and the same album with multiple songs in it.
10. You are required to have a makefile inside the project directory. The tar file should have the makefile in it. When I grade your project, after extracting from the tar file, all I will do is to type “make”to build your app and “make clean”to remove the executable and object code.

### How to Submit your files.

* 1. Create directory for the class: mkdir cs162
  2. Go inside the directory: cd cs162
  3. Create directories for the project: mkdir homework6
  4. Go inside the project directory: cd homework6
  5. Create source file for the program using vi, the editor: vi homework6.cpp
  6. An alternative editors nano: nano homework6.cpp
  7. Compile the source code(later on you will use makefile): g++ -o hw6 homework6.cpp
  8. Run the executable file: ./hw6
  9. To create a tar file of the project directory:
     1. remove the executable files in the project directory: rm hw6
     2. go to the parent directory that contains the project directory:

cd ..

* + 1. tar -cvf homework6.tar homework6
  1. The archive file should be transferred to your desktop, and then uploaded to Desire2Learn. You should also upload the design document for this program to the Desire2Learn dropbox.