CISC 105 Spring 2007 Project 3

1 Due Friday, May 11 at 11:55 p.m.

Paper copy due Monday in class

Note: Late projects will not be accepted after 3 p.m. Wednesday May 16th.

2 Introduction

I use software that keeps track of the tunes on my music storage device. I exported part of a tune list to a spreadsheet, deleted some columns, and provide the rest here as data in comma-separated value (CSV) format. Testing data may be different, but there will not be more than 200 tunes. You should write some of your own test data (testing will be much easier if you do!).

3 The Data

You'll see that the data include all sorts of punctuation and spaces in fields that we should treat as single entries, such as the album name "Hey! Ho! Let's Go: The Anthology (Disc 1)". This means sscanf with %s won't work (why?), so you will use fgets and strtok.

Your program may read the data from the file only once. Each tune's line of data will be stored in a single struct, and you will use an array of structs to hold all the lines you read.

All tunes read from the file will have a size field greater than zero. Columns composer, album, and year may not have entries for some tunes.

4 The Program

The parts of the project will be as follows:

40 percent:

1. Declare an array of structs (see Data). Read data into the array from a file.Write a function that takes only an open file pointer as a parameter, uses fgets and strtok, and returns a single struct of the type you declared.

   When this is working, separate the whole file reading process into a new function accessible from your menu. Allow the user to use a default data file name, or to enter a new one.

20 percent:

2. Write two functions: one that prints single structs, and one that uses that function to print all structs. Time (stored as seconds) should be printed as MIN:SEC, e.g. 3:24. Size should be printed as megabytes with two decimal places, not bytes as recorded. Size will not exceed 100 Mb. Add printing all tunes to your menu.

3. Write a function to write the data to a new file (name chosen by the user) in the same CSV format your code reads (you will have to demonstrate that it works). Use a function that writes a single struct parameter to a file pointer parameter. Add this to your menu.

   Now a user can write to a new data file and then read from it. Cool!

30 percent:

4. All prototypes will be in an .h file.
5. Let user add a new tune's data. Add this to your menu. Your program must prompt the user for each datum.
6. Sort the array of structs three ways: by tune name, artist, or size. Add a sorting option to your menu, which will take you to a separate sorting menu function to determine sort type. After sorting, the user is returned to the main menu.
7. The input filename will be a command line argument, with a nice error message if it is not entered.

10 percent:

This section may only be started when the others are completed and working properly.

8. Menu will be char driven (i.e. users will enter characters for menu choice, not integers), and will detect faulty input such as "22" or "cat" and ask for new input (see 9.6 in your text for some useful char operations).
9. Each struct will be dynamically allocated as needed. The array will be an array of pointers only.

5 The Menu

No matter how far you have gotten in the project, your project must have a menu to demonstrate functionality. Be sure to allow ample time to complete it properly. All of the functionality listed above must be available from the menu, except reading the file. So for example, your program might sequentially print, add, print, add, write, sort, print, sort, write.

Your menu is to provide a simple means for demonstrating and using your program. This is the interface between you, the programmer, and your customer, so please check carefully for **spelling** and **neatness** (they count).

Use a switch statement for your menu. Each case should consist of minimal setup code and then a function call. The printed menu options must be displayed by a separate function.

6 Tips

Code incrementally. Save lots of backups versions, like proj3.1.c, proj3.2.c etc. Work on small problems in small programs - for example, don't try to get your strtok code working at the same time you are starting to read from a file with fgets; instead, test your code thoroughly with simple input strings that you design. After it works, then try to get functions to work with other functions.

7 Submission

Design and test your program carefully using data that you make up as well as the data provided. Check the calculations! The day the project is due you will be given test data and a project submission sheet. You must show that your program performs correctly for all test data, and you must complete the **submission sheet** and turn it in with your project's **paper copy**. Of course, code for your program must be electronically submitted to **MyCourses** along with your script file. Paper copy is due the following class.