APS 105 — Computer Fundamentals

Lab 9: Sorting

Winter 2018

In this lab, you will write a program that maintains your personal music library in an array, and sorts it by artist. This lab will be due on April 6 or April 9, depending on your lab section. The due dates are the same as Lab 8.

Your task in this lab is to use a specific sorting algorithm, called *cocktail sort*, to sort your personal music library in ascending alphabetical order by artist. Your complete C program in this lab should be named `Lab9.c`. You may reuse as much of the code from Lab 8 as you wish. Moreover, as with Lab 8, we will provide you with skeleton code for Lab 9. (It is possible to complete Lab 9 by adding only 10 lines of code to the skeleton code.)

Each song in your program will be represented by the following structure:

```
typedef struct song {
    char songName[MAX_LENGTH];
    char artist[MAX_LENGTH];
    char genre[MAX_LENGTH];
} Song;
```

Rather than using a linked list, your personal music library in this lab will be stored in an array. You can assume that there will be no more than 100 songs in the library:

```
Song Library[MAX_LIBRARY_SIZE];
```

This declaration creates 100 elements of an array, each of which has a `songName`, `artist`, and `genre` in it. As a result, you will not need to dynamically allocate any data. A pointer to the string for the song name for element `i` of that array would be referred to as:

`Library[i].songName`,

and similarly the artist name by:

`Library[i].artist`.

Also, to access the third character of the song name for element `i` of the array, you would refer to it as

`Library[i].songName[2]`.

Similar to Lab 8, your program will support a command-driven interface, with the following four commands:

- **Command** `I`: Inserts a new song to the end of the music library. You may use the same `inputStringFromUser()` function in Lab 8 to prompt the user for a song name, an artist, and a genre. The insertion should be performed into the element after the end of the current set of songs in the array. For example, the first insertion should go in the first element of the array, the next at the second, and so on.

- **Command** `P`: Prints the entire music library. If the library is empty, your program prints `"The music library is empty."` by calling the `printMusicLibraryEmpty()` function, provided to you in Lab 8.
- **Command** `S`: Sorts the music library in ascending alphabetical order by artist. If two or more songs have the same artist, it sorts in ascending alphabetical order by their song names.

  To implement this command, your program should call the `cocktailSort()` function with the following prototype:

  ```
  void cocktailSort(Song library[], int size);
  ```

  The function implements a slight variation of the bubble sort algorithm: instead of repeatedly passing through the list from bottom to top as in each iteration of bubble sort, Cocktail sort passes alternately from bottom to top and then from top to bottom. Similar to bubble sort, the sorting algorithm stops as soon as it is detected that the array is sorted.

- **Command** `Q`: Quits the program without printing anything.

An error message `"Invalid command."` will be printed for all other commands. The program will then continue to prompt for the next command.

As always, we strongly recommend that you implement this program in small increments, testing the program after each step.

Sample Output From Executing The Program

Here is a sample output from an execution of the program that you are to prepare. There is one space after each colon (`:`).

```
Personal Music Library.

Commands are I (insert), S (sort by artist), P (print), Q (quit).

Command --> P

The music library is empty.

Command --> A

Invalid command.

Command --> I
Song name --> Shape of You
Artist --> Ed Sheeran
Genre --> Boring

Command --> I
```

```
Song name --> The Shade
Artist --> Metric
Genre --> Rock

Command --> I
Song name --> Heads Will Roll
Artist --> Yeah Yeah Yeahs
Genre --> Punk

Command --> I
Song name --> The One
Artist --> The Chainsmokers
Genre --> Pop

Command --> P

My Personal Music Library:

Shape of You
Ed Sheeran
Boring

The Shade
Metric
Rock

Heads Will Roll
Yeah Yeah Yeahs
Punk

The One
The Chainsmokers
Pop

Command --> S

My Personal Music Library:

Shape of You
Ed Sheeran
Boring

The Shade
Metric
Rock

The One
The Chainsmokers
Pop
```

```
Heads Will Roll
Yeah Yeah Yeahs
Punk

Command --> I
Song name --> Portland
Artist --> Drake
Genre --> Hip hop

Command --> P

My Personal Music Library:

Shape of You
Ed Sheeran
Boring

The Shade
Metric
Rock

The One
The Chainsmokers
Pop

Heads Will Roll
Yeah Yeah Yeahs
Punk

Portland
Drake
Hip hop

Command --> S

My Personal Music Library:

Portland
Drake
Hip hop

Shape of You
Ed Sheeran
Boring

The Shade
Metric
Rock

The One
```

```
The Chainsmokers
Pop

Heads Will Roll
Yeah Yeah Yeahs
Punk

Command --> Q
```

Grading by TA and Submitting Your Program for Auto-Marking

There are a total of **10 marks** available in this lab, marked in two different ways:

1. **By your TA, for 4 marks out of 10.** Once you are ready, show your program to your TA so that it can be marked for style, and to ask you a few questions to test your understanding of what is happening. Programs with good style have been described in previous labs, so that will not be repeated here.

2. **By an auto-marking program for 6 marks out of 10**. You must submit your program file, named `Lab9.c`, through the ECF computers for marking. We will be using a software program to compile and run your program, and test them with different inputs.

Similar to the previous labs, long before you submit your program for marking, you should run the **exercise** program that compiles and runs your program and give it sample inputs, and checks that the outputs are correct. You should run the following command:

```
/share/copy/aps105s/lab9/exercise
```

within the directory that contains your solution programs.

This program will look for the file `Lab9.c` in your directory, compile it, and run it on *some* of the test cases that will be used to mark your program automatically later. If there is anything wrong, the **exercise** program will report this to you, so read its output carefully, and fix the errors that it reports.

Once you have determined that your program is as correct as you can make it, then you must submit your program for auto-marking. This must be done by the end of your lab period as that is the due time. To do so, go into the directory containing your solution files and type the following command:

```
/share/copy/aps105s/lab9/submit
```

This command will re-run the exercise program to check that everything looks fine. If it finds a problem, it will ask you if you are sure that you want to submit. Note that you may submit your work as many times as you want prior to the deadline; only the most recent submission is marked.

The **exercise** program (and the **marker** program that you will run after the final deadline) will be looking for the exact letters as described in the output in this handout, including the capitalization. When you test your program using the exercise program, you will see that it is expecting the output to be exactly this, so you will have to use it to see if you have this output correct.

Important Note: You must submit your lab by the end of your assigned lab period. Late submissions will not be accepted, and you will receive a grade of zero.

You can also check to see if what you think you have submitted is actually there, for peace of mind, using the following command:

```
/share/copy/aps105s/lab9/viewsubmitted
```

This command will download into the directory you run it in, a copy of the file that has been submitted. If you already have a file of that same name in your directory, that file will be renamed with a number added to the end of the filename.

After the Final Deadline — Obtaining Automark

Briefly after all lab sections have finished, you will be able to run the automarker to determine the automarked fraction of your grade on the code you have submitted. To do so, run the following command:

```
/share/copy/aps105s/lab9/marker
```

This command will compile and run your code, and test it with all of the test cases used to determine the automark grade. You will be able to see those test cases' output and what went right or wrong.