CpSc 1010 Spring 2018
Programming Assignment #2
Playlist

Due 11:59 pm Wednesday, March 14th

1. Overview

The purpose of this assignment is to give you practice with arrays and with writing functions, in addition to the other topics from the first assignment, such as declaring and using variables, writing mathematical expressions, using `printf()` and `scanf()` functions, as well as loops and conditional statements.

2. More Assignment Specifics & Example Output

PHASE I

For this assignment, you will convert assignment 1 so that the song data (genre and length of song) is put into a 2-D integer array. The first dimension corresponds to the number of rows, i.e. number of songs. The second dimension corresponds to the number of columns, i.e. genre in first column, length of song in second column.

The first "phase" for this assignment should consist of just this part, where the data is read into a 2-D array. The data being read in will change slightly from the first assignment, though. The user should be asked for the number of songs in the playlist; once that is known, the array can be declared. That number will also be used as a sentinel value for whatever loops are in your program. The user will no longer be prompted for whether there is another song. See below (user input in bold):

```
How many songs are in your playlist?

6

genre:
           1 for rock
           2 for blues
           3 for indie
           4 for rap
           5 for dance/electronic
           6 for reggae
           7 for country
           8 for r&b
           9 for pop
 - - > 3
   length of song: 194


genre:
           1 for rock
           2 for blues
           3 for indie
           4 for rap
           5 for dance/electronic
           6 for reggae
           7 for country
           8 for r&b
           9 for pop
 - - > 5
   length of song: 198


genre:
           1 for rock
           2 for blues
           3 for indie
           4 for rap
           5 for dance/electronic
           6 for reggae
           7 for country
           8 for r&b
           9 for pop
 - - > 4
   length of song: 219


genre:
           1 for rock
           2 for blues
           3 for indie
           4 for rap
           5 for dance/electronic
           6 for reggae
           7 for country
           8 for r&b
           9 for pop
 - - > 3
   length of song: 231


genre:
           1 for rock
           2 for blues
           3 for indie
           4 for rap
           5 for dance/electronic
           6 for reggae
           7 for country
           8 for r&b
           9 for pop
 - - > 3
   length of song: 156


genre:
           1 for rock
           2 for blues
           3 for indie
           4 for rap
           5 for dance/electronic
           6 for reggae
           7 for country
           8 for r&b
           9 for pop
 - - > 4
   length of song: 297

Your playlist of 6 songs has:
   3 indie songs
      shortest was at 2:36
      longest was at 3:51
      average length was 3:13

   2 rap songs
      shortest was at 3:39
      longest was at 4:57
      average length was 4:18

   1 dance/electronic songs
      shortest was at 3:18
      longest was at 3:18
      average length was 3:18
```

Once you're sure this works, you can create an input file for the short list of songs and redirect the input to make sure it works. Using the 6 songs above, and commenting out all the user prompts, the output would look like this:

```
Your playlist of 6 songs has:
   3 indie songs
      shortest was at 2:36
      longest was at 3:51
      average length was 3:13

   2 rap songs
      shortest was at 3:39
      longest was at 4:57
      average length was 4:18

   1 dance/electronic songs
      shortest was at 3:18
      longest was at 3:18
      average length was 3:18
```

PHASE II

The next phase for the assignment will consist of splitting your code up into functions. Prototypes for the functions that are required are:

```
void fillArrays(int howMany, int songs[][2]);
void getInfo(int howMany, int songs[][2]);
void getSongType(int genreNum, char songType[45]);
```

You may have more functions than these, but these three are required.

Remember that the purpose of having functions is to reduce redundant code, and also reduce the length of the functions so that a function generally performs one task. With the use of arrays, the number of variables is also reduced. You'll start to see a more concise, compartmentalized program with this assignment.

For example, my `main()` function consists of only 5 lines (not including the return statement) and only three local variables.

The `fillArrays()` function will be called from within the `main()` function. The first argument is the number of songs entered by the user and will be used as a sentinel value for the loop inside this function which will get the genre and the length of song values, putting them into the array, which is the second argument. I have two local variables declared inside this function.

The `getInfo()` function will be called from within the `main()` function. The first argument, again, is the number of songs entered by the user, and the array is the second argument. Inside this function, you will traverse the array, once for each genre, each time getting the information needed from the array to calculate and display the summary information. The print statements that show the summary information will be printed from within this function, after the information for each genre is determined. I have a total of 13 variables in this function.

The `getSongType()` function will be called from within the `getInfo()` function (to reduce the number of lines in the `getInfo()` function). The first argument is the genre number, and the second argument is a character array, or string, to hold the string for the name of the genre. So, if the genre number is 1, then the array will hold the string "rock". You have the choice of using a switch statement or an if-else-if statement in this function. The purpose of this function is to give you practice with calling a function from within another function other than the `main()` function. No other local variables are needed in this function.

3. What to Hand In

Your file MUST BE NAMED playlist2.c and submitted along with your data file to the handin.cs.clemson.edu page by midnight Wednesday, March 14th.

The source code will be evaluated not only on its correctness, but also on its adherence to the coding style standards "Programming Assignment Requirements" provided on Canvas. You should also take a look at "Formatting Code Examples" also provided on Canvas. Don't forget to use meaningful variable names, and don't forget about indentation, comments, and lines not exceeding 80 characters (including spaces).

4. Notes on Collaboration

You are required to work individually on this assignment. **Please do not consult *anyone* other than your instructor or the lab assistants on *any* aspect of this assignment**. Other tutors that are available may be able to help with more general C concepts that you may be struggling with, but not with questions specific to the assignment.

Review the section on "Academic Integrity" on the Course Syllabus for the policy on what happens when copying or sharing code.

For any of the programming assignments in lecture and/or lab, it is considered cheating to do any of the following:

- discuss in detail the C code in your program with another student
- show or share the C code in your program with another student
- use C code obtained from another student, or any other unauthorized source, either modified or unmodified (each student is responsible for protecting his or her files from access by others)
- use re-engineering tools
- submit work of others, from the Internet or any other source, even if attributing the work to others

5. Grading Rubric

If your program does not compile on our Unix machines or your assignment was not submitted on time, then you may receive a grade of zero for this assignment. Otherwise, points for this programming assignment will be earned based on the following criteria:

55%	functionality (program works as shown in the example output)
10%	formatting of code (such as header comment + other comments, indentation, meaningful variable names, lines not longer than 80 characters, etc.)
15%	use of a 2-D array for the song data
15%	at least the three required functions
5%	no warnings when compile