CSC 1310 LAB 3 – Sort Algorithms

WHAT SHOULD THIS PROGRAM DO?

Your program will sort an array of Song objects by song title in ascending order using **Insertion Sort**, then descending order using **Reverse Bubble Sort**, and then ascending order again using **Quicksort**. For each sort algorithm, the algorithm will be timed and the number of seconds will be printed to the screen. Also, for each sort algorithm, the data will be printed to a text file.

IMPORTANT!!! You are sorting the entire song OBJECT, not just the song title.

Driver.cpp is partially provided for you. This is the file that contains the main function. **You will be adding all the necessary programmer-defined functions to Driver.cpp to accomplish the sorting algorithms. You should not have to write or modify code in any of the other given files!!**

The Song class (**Song.h**) is provided for you. A Song object has a song title, artist, and length of song (integer). **After the sorting algorithm has happened, the song objects should be sorted in increasing order (ascending order) by song title.**

The Timer class (**Timer.h** & **Timer.cpp**) is provided for you. This is the class that is used in the Driver.cpp file to time the algorithms.

I also have provided a folder of sample sort programs (called Sample Sort Programs) that may be helpful to look at for this lab. You don't have to turn these back in with your lab submission.

HOW TO TEST YOUR PROGRAM

Compile your program with `g++ Timer.cpp Driver.cpp`.

You must test your program with **four** different text files, which I have provided: **5songs.txt** (5 songs), **songs.txt** (130 songs), **over10000Songs.txt** (10130 songs), and **over50000Songs.txt** (50130 songs). Note that I created all the songs over the first 130 with a random generator so the data is the correct data types – but doesn't really make sense, but it was just a way that I could generate a really large amount of "song" data without taking years to type it all out.

You will be making a screenshot of each of these four tests successfully running. **These images should be part of your submission**. You can either place the images in one document file or submit them as three different image files. As long as we can see that you were able to successfully run the three tests described below.

FIRST TEST - USING 5SONGS.TXT (CONTAINS DATA ON 5 SONGS)

You should use 5songs.txt as your input. Make sure 5 Song objects were created (this should print out to the screen). During running of the program three text files should be created. Check to make sure that sortFileInsertion.txt contains the songs in ascending order. sortFileReverseBubble.txt contains the songs in descending order, and sortFileQuick.txt contains the songs in ascending order. If this test works and looks similar to the sample output (probably will be 0 seconds for all three sorts) then you are ready to move on to 130 songs.

```
What is the name of the file with songs? (example.txt)
5songs.txt

You have created 5 Song objects.

Insertion sort: 0 seconds
Reverse bubble sort: 0 seconds
Quicksort: 0 seconds
```

FIRST TEST - USING SONGS.TXT (CONTAINS DATA ON 130 SONGS)

You should use songs.txt as your input. Make sure 130 Song objects were created (this should print out to the screen). During running of the program three text files should be created. Check to make sure that sortFileInsertion.txt contains the songs in ascending order. sortFileReverseBubble.txt contains the songs in descending order, and sortFileQuick.txt contains the songs in ascending order. If this test works and looks similar to the sample output (probably will be 0 seconds for all three sorts) then you are ready to move on to over 10,000 songs.

```
What is the name of the file with songs? (example.txt)
songs.txt

You have created 130 Song objects.

Insertion sort: 0 seconds
Reverse bubble sort: 0 seconds
Quicksort: 0 seconds
```

SECOND TEST – USING OVER10000SONGS.TXT (CONTAINS DATA ON 10,130 SONGS)

You should use over10000Songs.txt as your input. Make sure 10,130 Song objects were created (this should print out to the screen). Check to make sure that sortFileInsertion.txt contains the songs in ascending order. sortFileReverseBubble.txt contains the songs in descending order, and sortFileQuick.txt contains the songs in ascending order. If this test works and looks similar to the sample output, then you are ready to move on to over 50,000 songs.

```
What is the name of the file with songs? (example.txt)
over10000Songs.txt
```

```
You have created 10130 Song objects.

Insertion sort: 4 seconds
Reverse bubble sort: 10 seconds
Quicksort: 0 seconds
```

THIRD TEST – USING OVER50000SONGS.TXT (CONTAINS DATA ON 50,130 SONGS)

Note that this test may take a while to run – that doesn't mean your program has crashed. Mine took a little over 14 minutes to run. You should use over50000Songs.txt as your input. Make sure 50,130 Song objects were created (this should print out to the screen). Check to make sure that sortFileInsertion.txt contains the songs in ascending order. sortFileReverseBubble.txt contains the songs in descending order, and sortFileQuick.txt contains the songs in ascending order. If this test works and looks similar to the sample output (probably will be 0 seconds for all three sorts) then you are ready to move on to over 50,000 songs.

```
What is the name of the file with songs? (example.txt)
over50000Songs.txt

You have created 50130 Song objects.

Insertion sort: 95 seconds
Reverse bubble sort: 238 seconds
Quicksort: 10 seconds
```

WHAT TO TURN IN

Zip the following files in a single zip folder and upload to the Lab 3 Submission Folder.

- **Driver.cpp**
- **Song.h**
- **Timer.cpp**
- **Timer.h**
- **5songs.txt**
- **Songs.txt**
- **over10000Songs.txt**
- **over50000songs.txt**
- **screen captures of your program running four different times using the four files.**