Lab 03 Specification – Sorting your Music PlayList
Due (via your git repo) no later than 8 a.m., Thursday, 14th March 2019.
50 points
[A Team based two-week lab assignment]

Lab Goals

- Getting familiar with Sorting algorithms such as Insertion, Bubble, Selection, and Shell sort and most importantly apply these algorithms to a real life problem. As we progress into the next two weeks timeline, we will examine other Sorting algorithms such as Quick sort, Merge sort, and Heap sort. It is recommended to implement the new set of algorithms mentioned above into your lab assignment as we discuss the topics in the upcoming classes.
- Apply data structure [such as Arrays] knowledge within the context of Sorting algorithms to implement an efficient solution to a real life problem.
- To explore as a team to solve a real life problem, from the design phase to address open ended issues, to integrating algorithmic solutions, to implementing an end product with appropriate testing and validations. This exploration is a huge learning objective for the lab and a preparatory step to gear towards the final course project.

Suggestions for Success

- Take a look at the suggestions for successfully completing the lab assignment, which is available at: `https://www.cs.allegheny.edu/sites/amohan/resources/suggestions.pdf`
- Often times, the lab specification is purposefully developed to include open ended tasks and not exposing too low level details. The intention is to trigger you towards further refining your thinking skills. Thinking is more important to solve any algorithmic problem provided. Thinking through, may lead you to ask further clarifying questions, which is what my expectation is. So, there will be one or more open ended parts added to the lab. Attending labs regularly and steering discussions with the Professor, TA's and your classmates is going to be the success mantra for doing a great job in the labs. Waiting till the last minute, will restrict a student from both learning the content discussed and to have a negative effect to the effectiveness of the lab work produced.

Learning Assignment

If you have not done so already, please read all of the relevant "GitHub Guides", available at https://guides.github.com/, that explain how to use many of the features that GitHub provides. In particular, please make sure that you have read guides such as "Mastering Markdown" and "Documenting Your Projects on GitHub"; each of them will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read

- Sedgwick chapter 02 [as recommended in the reading assignment section of the class slides.]

Assignment Details

We will follow the team assignment document that is accessible through our course webpage. The lab work should be done as a team through proper collaboration within the team and each member should taking the responsibility to contribute to the project in a fair manner.

Although the lab is not due for two weeks, I expect the students to be in the lab on March 7th to continue working with your group on this assignment and ask proper clarifying questions with the TA's and the Professor.

Interacting with the team member's regularly, formulating a solid plan for implementation, talking through design issues, distributing the workload in the most efficient way by considering individual strength and weakness is all going to be the key to do well in this lab assignment. If not properly planned, then there is a risk that the team will not be able to deliver the lab solution on time. Additionally, it is highly recommended to take proper assistance from Professor Mohan. Professor Mohan can partner with you and provide design level suggestions as an architect in your team. Do not hesitate to get assistance from the Professor. As the african proverb states a single stick may smoke but not burn. It is important to accept the fact that a good software project is developed only by utilizing all the available resources in an appropriate manner.

The project code can be implemented in any programming language. Each team can make their own decision to choose an appropriate programming language based on the team's strength, weakness, and personal choice. But, since we had used primarily Java as the programming language to implement our in-class discussion points and ideas, there are Java based code files provided for the Sorting algorithm implementations under the **src** sub-directory. In order to ease up the process of compiling, and executing the Java code files, there is two bash files provided along within the repository.

In order to compile your programs, make sure to place all your Java files under the src sub-directory, Then, open the terminal and navigate to the lab repository and type in:
"./compile.sh"
This should compile all your files and automatically place their class files in the classes sub-directory located in the lab repository.

In order to execute your programs, make sure to open the terminal and navigate to the lab repository and type in:
"./run.sh MusicPlayList"
Here MusicPlayList is the class name. The bash script can be executed in the similar way for a different class name as well.
If the team had chosen a programming language other than Java for implementation, then it is required to provide detailed information on how to compile and execute the program through the README file. Although not required, it is recommended that the team would make necessary modifications to the bash script files based on the programming language, so as to ease the process of compiling and executing the code.

Playlist Population

Requirement: Populate your playlist as a simple text file. It should contain at least 30 entries and should contain the following information for each song (you will sort your playlist based on these):

- Song name
- Artist(singer) name
- Genre (rock, pop, alternative, hip hop, classical, country, etc.)
- Year
- Ranking (number of views, listens, or downloads can be used as a measure of ranking; the ranking is allowed to be made-up if you can not find the current ranking data)

It is not required to type all the information by yourself, feel free to copy and paste information about the songs you want from existing music databases or you can use a part of existing dataset (you should be able to find these by searching for 'music playlist dataset'). Although not a requirement, it is recommended to have more than 30 songs in your text file. If you are interested to take this to the next step and have a large repo of songs, then you may consider partnering with Professor Mohan and setup a web scraper to scrape the data from random online data sources and automate this process completely. This could be a very good seed project to do an interesting course project for this semester.

Playlist Processor

Requirement:

Write a program that will process the playlist and save the entries of the songs in appropriate data structures. For example, you may create a simple array that will hold the ranking, and an array that will hold the song name, etc. For genre, you may use a more complicated data structure. As part of the learning objective for this lab. it is important to make sure that the team gets an opportunity to apply their Data Structure knowledge in the context of Sorting algorithm.

Playlist Sorter

Requirement:

Extend your program to include sorting of your playlist according to the song entries. You may implement a series of sorting algorithms into the Sorting code repository. The sorting algorithms and their implementation details can be considered as a black box to the end user. As mentioned in the document earlier, the repository may include Insertion, Bubble, Selection, and Shell sort. As we progress into the next two weeks timeline, we will examine other Sorting algorithms such as Quick, Merge, and Heap sort. It is recommended to consider implementing the new set of algorithms mentioned above into the Sorting code repository as we discuss the topics in the upcoming classes. A hard requirement is to include at least three of the Sorting algorithms from the list provided above. As a team, you should discuss ideas on how to easily integrate the repository to your code base. For example: let us suppose there is a new Sorting algorithm added to the Sorting code repository called RadixSort, then your code base should still function without major design changes to the structure of the code.

It is acceptable for you to reuse the Sorting code written in Java directly from in-class discussions or refer to other online resources for the Sorting algorithm implementations. There is a creative freedom for you to pursue these different paths, but it is required for you to properly cite the resources from where the code was taken through your documentation (in the README file)!

The major functional part of the Playlist Sorter, is to be able to sort your playlist by:

- name of the song - alphabetically a–z
- name of the singer - alphabetically a–z
- ranking - numerically 1–50
- year - numerically
- genre and ranking/name: this is slightly more complicated. First you need to create the personalized preference list (ranking) of the genres depending on what you like to listen to (for example, if I had three genres to rank from 'pop', 'alternative' and 'hip hop', I'd list them as: 'pop' - 2, 'alternative' - 1 and 'hip hop' - 3). Then you sort the list based on your preferences of genres. But since you may have several songs within the same genre (or perhaps all songs in your playlist are within the same genre), you will need to sort these songs according to their ranking/popularity or the artist name or the song name or the year (your choice) as well. An open ended design question is what data structure do you use to represent the genre and ranking/name?

Playlist Sorter Interface

Extend your program so that it asks the user what kind of algorithm he/she wants to use (you can restrict the responses to the algorithms you have implemented) and what kind of sorting he/she wants to do (by song name, by year, etc.), and then perform the specified sorting using the specified algorithm and save the results in the separate text file (name it appropriately). Although the sorting repository is considered as a black box to the end user, for testing purpose, it is required to prompt the user to provide the algorithm related information such as the Algorithm Type. Based on the user input, an appropriate Sorting algorithm should be triggered for execution and the entire flow should gets kick started from there. The input unsorted file (data created at the first step) and the output sorted file should be placed under the data sub-directory through your program. In this way, it is easier to the end user to see the results. Although not a requirement, the team can make a creative decision to display the final sorted data in the console additionally.

Submission Details

For this assignment, please submit the following to your GiHub repository by using the link shared to you by the course instructor:

1. Commented source code from the “PlaylistPopulation” program under the src sub-directory if you had automated the dataset creation process through code.
2. Commented source code from the “PlaylistProcessor” program under the src sub-directory.
3. Commented source code from the “PlaylistSorter” program under the src sub-directory.
4. Commented source code from the “PlaylistSorterInterface” program in the src sub-directory, which is the Driver program that would hold the main method.
5. A readme file explaining how to compile and run your program. In addition, include a detailed self reflection of the lab exercise in the readme file. The self reflection may include answers to questions such as:
   - ”What challenges did you face as a team in this lab?”
   - ”What challenges did you face as an individual in this lab?”
   - ”What did you like in this lab as a team?”
   - ”What did you not like in this lab as a team?”
6. It is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllabus. Make sure to add the statement ”This work is mine, unless otherwise cited.” in all your deliverables such as source code and other textual documents.

Grading Rubric

1. The lab is worth 50 points. It is expected that the submission has the complete solution to the problem above. The details regarding point deductions will be provided through the grading report file.
2. An important part of this lab is participating in the group meetings. It is required for each team to meet with the Professor and discuss design level issues, current progress, getting started ideas, and steering a brainstorming session. There is 10 points awarded for participation in the group meetings out of the total 50 points for the lab. There will be two group meetings conducted during the lab sessions, that is on Feb 28th and Mar 7th. There is a total of 5 points awarded for participating in each of those meetings. Failure to participate in the group meeting, will lead an individual team member to appropriate point deductions. The first meeting should be based on getting started ideas, workload distribution among team members, and so on. The second meeting

should be more detailed, where the team is required to share their partial implementation through a demo and demonstrate a clear plan for completing the lab assignment. To raise appropriate questions about the team's challenges and getting it addressed/clarified is another important part of the second meeting.

3. There will be full points awarded for the lab, if all the requirements in the lab specification are correctly implemented. Partial credits will be awarded if deemed appropriate.

4. Failure to upload the lab assignment code to your git repo, will lead you to receive "0" points given for the lab submission. In this case, there is no solid base to grade the work.

5. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in github. In this way, an updated version of team's submission will be used for grading. If the team did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the team had not submitted any code, then in this case, there is 0 points given to the team automatically for the lab work. All team members will get the same grade for the actual lab part except the points allocated to the group meetings.

6. If you needed any clarification on your lab grade, talk to the instructor. The lab grade may be changed if deemed appropriate.