DEPARTMENT OF
SOFTWARE TECHNOLOGY

CCDSALG (Data Structures and Algorithms)

Major Course Output 1: Sorting Algorithms

Major Details

Groupings:	At most 4 members in a group
Deadline:	
Percentage:	20% of final grade
Submission Guidelines:	Submit deliverables in AnimoSpace
Filename Format:	`source-CCDSALG-MCO1-`*`Section-GroupNumber`*`.zip`
	`report-CCDSALG-MCO1-`*`Section-GroupNumber`*`.pdf`

Deliverables

You are to submit the following files **separately** in AnimoSpace:

- A `.zip` file containing a folder called `source`, which contains **all** the source codes for the project; and
- A `.pdf` file that contains the project report.

Project Specifications

ID NUMBERS DATASET

- Once you unzip the starter files for the project, you will find a folder called `data`. This folder contains a set of files. Each file contains a set of records. Each record contains the name of a person and his/her corresponding ID number. The ID numbers are all unique, although the names may have duplicates.

- Each file is in the following format:

n <ID number 1> <name 1> <ID number 2> <name 2> <ID number 3> <name 3> ⋮ <ID number n> <name n>	**Explanation:** The file starts with an integer *n* on a single line, denoting the number of records in this file. This is followed by exactly *n* lines. Each line contains the ID number (an integer from 1 to 10000000), followed by a single space, followed by the name of the person. The ID number does not have spaces or commas between them. The name of the person can only consist of alphabetic characters and spaces.

- Each file should be treated independently of one another.

CODE STRUCTURE

- You are to accomplish this project using either C **or** Java programming language. The starter code given to you contains the following files:

    - **For those using C:**
        - `record.c`: Contains a `struct` for a single record. You must use this for your sorting algorithm. You are NOT allowed to modify this file.
        - `filereader.c`: Contains a helper function for reading a text file containing records. You are NOT allowed to modify this file.
        - `timer.c`: Contains a helper function for getting the current time in milliseconds. You are NOT allowed to modify this file.
        - `sortingalgorithms.c`: Contains the functions for sorting records.
        - `main.c`: Contains the main function and is the main entry point of the program.

    - **For those using Java:**
        - `Record.java`: A class to represent a single record. You are NOT allowed to modify this class**.**
        - `FileReader.java`: A helper class containing a method for reading in text files containing records. You are NOT allowed to modify this class.
        - `SortingAlgorithms.java`: A class containing methods for sorting records.
        - `Main.java`: A class containing the main method.

PART 1: SORTING ALGORITHMS

- For this section, your task is to sort the list of records by ascending ID number.

- In `sortingalgorithms.c` / `SortingAlgorithms.java`, complete the implementation of insertion sort, selection sort, and merge sort. All these methods accept an array of `Record` structures (in the case of C) or objects (in the case of Java). Each `Record` object has two properties: the name and the ID number. After the functions / methods are called, it is expected that the array of records will be sorted according to the ID number. **Make sure that the names are still associated with the corresponding ID numbers after sorting!**

- Additionally, you must implement **one (1) more sorting algorithm of your choice**, apart from the ones listed above. Write a method for the implementation of this sorting function in the `sortingalgorithms.c` / `SortingAlgorithms.java` class, under the designated section.

- You are NOT allowed to modify any of the existing function / method headers. You are, however, free to define additional helper functions /methods as well as variables to be used by the sorting algorithms. For Java implementations, helper methods should have an access modifier of `private`.

PART 2: RUNNING THE ALGORITHMS

- Use each sorting algorithm to sort each file from the **`data`** folder. **Make sure that when you call each sorting algorithm, you pass the original, unsorted list of records, not a list that has already been sorted previously.**

- Verify that all your sorting algorithms **work correctly on all the given datasets**.

- Record the **execution time** and the **empirical frequency count** (number of steps) of each algorithm.

- To get the execution time in C, you can use the helper function **`currentTimeMillis()`** provided in **`timer.c`**. Below is an example of how to record the execution time of a certain algorithm:

```
long startTime, endTime, executionTime;
startTime = currentTimeMillis(); // store the current time
/* CALL THE ALGORITHM HERE */
endTime = currentTimeMillis(); // store the current time
executionTime = endTime - startTime;
```

  To get the execution time in Java, you can use the built-in **`System.currentTimeMillis()`** function of Java. Below is an example of how to record the execution time of a certain algorithm:

```
long startTime = System.currentTimeMillis(); // store the current time
/* CALL THE ALGORITHM HERE */
long endTime = System.currentTimeMillis(); // store the current time
long executionTime = endTime - startTime;
```

  To improve empirical rigor, benchmark the algorithm multiple times (at least 5) and record the **average** execution time. Make sure that you are measuring **only** the execution time of the algorithm itself, not including preprocessing and postprocessing steps such as reading the file or printing the results.

- When getting the **empirical frequency count**, make sure you are getting the **actual number of steps performed by the algorithm for each dataset**. You should have one count for every pair of algorithm and dataset.

- **You may use additional datasets for your comparative analysis**, by generating your own data. However, you should at the minimum include the given datasets in your results.

- You can use the main function / method in **`main.c`** / **`Main.java`** to run your experiments. However, the main function / method will not be checked. Keep in mind that your instructor **may use other datasets** to verify the correctness of the implemented functions.

WRITING THE REPORT

- You are to write a report containing, at the minimum, the following information.
    1. A brief introduction of the project and an outline of the contents of the report.

    2. The list of sorting algorithms implemented (including the required algorithms), along with a concise description of each algorithm and its implementation.

    3. A clear description of the process/es used in running the algorithms on the datasets, including the process/es used to verify the correctness of the results for the given data, and the process/es used to benchmark the execution times. **Show supporting code if necessary.**

    4. The execution times and frequency counts of the different algorithms on the given datasets. You can use tables and visualizations to convey the information as clearly as possible.

    5. A comparative analysis of the different algorithms across the different datasets, including theoretical basis that support the observed performance of the algorithms with respect to the datasets. Feel free to use additional datasets to validate the results or generate even more insights.

    6. Summary of findings and the group's learnings, insights, and realizations **with respect to sorting algorithms** after accomplishing this project.

- The beginning of the report should contain the **list of members**, as well as the **group number**.

- At the end of the report, there should be a **table detailing the contributions of each group member**, as well as a **References** section if needed. References are to be cited using APA format.

- There is no required format for the report. However, keep in mind the following:
    1. The report is graded based on substance, not the length or the number of pages. When writing, go directly to the point; avoid long blocks of text without substance.

    2. Make sure that you cover all the points enumerated above in a substantial manner; otherwise, you won't get a perfect grade.

    3. The report should have a clear narrative; its purpose is to convey your group's journey in doing the project and the learnings and insights you gained from it.

Working with Groupmates

For this project, you are encouraged to work in groups of at most 4 members. Make sure that each member of the group has approximately the same amount of contribution to the project. Problems with groupmates must be discussed internally within the group and, if needed, with your instructor.

All members of the group should have a clear understanding of the submitted project, regardless of the division of tasks. Collaboration should be done with a genuine concern for the outcome of the project – not just for the sake of distributing the workload. The instructor may ask **any member** of the group regarding the submitted project. Failure to demonstrate that the member has a substantial understanding of the submitted work will result in **major deductions**.

Academic Honesty Policy

Honesty policy applies. You should explicitly acknowledge the source, i.e., the author (if available) and the URL of any resources that you used for the making of the project. You can write this acknowledgment in the report.

The student handbook states that (Sec. 5.2.4.2):

"Faculty members have the right to demand the presentation of a student's ID, to give a grade of 0.0, and to deny admission to class of any student caught cheating under Sec. 5.3.1.1 to Sec. 5.3.1.1.6. The student should immediately be informed of his/her grade and barred from further attending his/her classes."

The student handbook also states that (Sec. 10.3):

A student caught cheating, as defined in Sec. 5.3.1.1., shall be penalized with a grade of 0.0 in the requirement or in the course, at the discretion of the faculty member, without prejudice to an administrative sanction. In cases of alleged cheating, the faculty member should report the incident to the Student Discipline Formation Office (SDFO).