University of Central Lancashire UCLan **School of Psychology and Computer Science**	**UCLan Coursework Assessment Brief**	2023 - 2024
	Module Title: Computational Thinking Module Code: CO2412	Level 5
	Sort the Data!	This assessment is worth 60% of the overall module mark

THE BRIEF/INSTRUCTIONS

Module Learning Outcomes

On successful completion of this module a student will be able to:	
1.	Use appropriate methods including logic and probability to reason about algorithms and data structures.
2.	Compare, select and justify algorithms and data structures for a given situation
3.	analyse the space and time complexity of simple algorithms
4.	Use a range of appropriate notations to represent and analyse problems
5.	Implement and test algorithms and data structures

The Background Story

You have recently joined the research team at a prestigious computer science institute known for its contributions to algorithm analysis. The institute has embarked on an exciting project to compare and analyse the performance of three fundamental sorting algorithms—Selection Sort, Merge Sort, and QuickSort—across different dataset sizes. Your role in this project is to conduct a comprehensive analysis and provide valuable insights to help determine the most efficient sorting algorithm.

You are also given the task to combine two separate datasets containing first names and last names provided in text files (firstNames.txt & lastNames.txt) together in line with the instructions outlined in step 1 of the task description below. You decide to leverage the power of Python to assist you with this task.

Task Description:

Learning Materials: Weeks 1-5
Further Reading Metaheuristic Search Concepts (Chapter 2) (Zapfel et Al)

Your task is to create and analyse three different-sized random arrays containing 100, 1,000, and 10,000 elements. You will apply Selection Sort, Merge Sort, and QuickSort to these arrays to compare their efficiency in terms of sorting speed and resource usage.

Instructions:

1. **Name List Shuffler:**
   a) Write a Python program to open the provided text files (firstNames.txt & lastNames.txt) and trim each file so that they only hold 4000 lines of data. The program should check each file and output the number of lines.
   b) Your Python script should be able to randomly generate a list of full names including first names and last names (for example: James Smith) with the size of 4000 entries from the provided text files (firstNames.txt & lastNames.txt) and save the generated list into a text file (FullNames.txt).

c) Your Python script should be able to output the longest full name (name with the highest number of characters) entered in the list generated from section **1b**.

2. **RNG Data Sorter:**
    a) Create three random arrays with the following sizes: 100, 1,000, and 10,000 elements. These arrays will simulate real-world datasets. Each generated number should have 6 digits ranging from 100,000 to 999,999.
    b) Implement all three sorting algorithms: Selection Sort, Merge Sort, and QuickSort.
    c) Apply each sorting algorithm to the three arrays separately. Ensure that you are sorting copies of the original arrays to maintain data integrity.

3. **Performance Analysis of sorting algorithms:**
    Measure and analyse the following performance metrics for each sorting algorithm and dataset size including:
    a) The execution time (in milliseconds) for sorting each array.
    b) Number of comparisons made during the sorting process.

4. **Graphical Comparison of sorting algorithms:**
    a) Generate and create graphs (e.g., line charts or bar charts) to visually compare the performance of the three sorting algorithms across various dataset sizes.
    b) Present the execution time data versus the array sizes in a clear and concise manner to facilitate easy comparison.

5. **Report Writing (1,000 words limit) *:**
    a) Explain what is meant by Big O Notation.
    b) Explain which sorting algorithm is the fastest and why, based on your analysis in terms of Big O Notation of each algorithm.
    c) Create graphs (e.g., line charts or bar charts) to visually compare the performance of the three sorting algorithms across various dataset sizes.

***Ensure that your report includes proper references and a bibliography. As well as a tittle page including the title of the assignment and your G number.**

Deliverables

- A Python .py program file for the Name List Generation.
- A Python .py program file for the Random Array Generation.
- A text file containing the full name output for part 1.
- A text file containing the sorted array list (optional for 2.1 criteria)
- A Microsoft Word document for the report (1,000 words maximum).

Marking Scheme

Marking bands are indicative and can be overridden at the marker's discretion with justification.

40 – 49% (Pass Criteria)

- All the developed Python .py scripts (as specified in sections **1** and **2**) are executable.

- The programme for full name generation (as specified in section **1** - **Name List Shuffler**) functions as expected.
- The programme for random number generation operation (as specified in section **2** - **RNG Data Sorter**) functions as expected.
- The Report contains parts **5a** and **5b** specified in section **5** of the instruction to a reasonable standard.

50 – 59% (2.2 Criteria)

- As above.
- The implemented sorting algorithms (as specified in section **2** - **RNG Data Sorter**) are using the same copy of the generated random arrays to maintain data integrity.
- The report discusses all criteria (**5a**, **5b**, **5c**) specified in section 6 of the instruction to a reasonable standard.
- All the developed Python .py scripts (as specified in sections **1** and **2**) have suitable data structure.
- Any evidence of an acceptable code commenting is demonstrated.
- All the requirements from section **3** of the instruction (Performance Analysis) are implemented.

60 – 69% (2.1 Criteria)

- As above.
- All criteria (**5a**, **5b**, **5c**) specified in section **5** of the instruction are covered to a high standard using technical language.
- The sorted arrays for each sorting algorithm are saved in a separate output text file.
- The script specified in section **2** - **RNG Data Sorter** demonstrates that the sorting algorithms use the least amount of memory resources including variables and avoiding magic numbers.
- Good programming practice is evident to a high standard in all the developed Python .py scripts (e.g., commenting, naming conventions for variables and correct indentation)
- Any evidence of using “Functions” is demonstrated in all the developed Python .py scripts.

70 – 100% (1st Criteria)

- As above.
- All criteria (**5a, 5b, 5c**) specified in section 5 are covered to an excellent standard with competent use technical language.
- Evidence of any mechanism for ensuring that all the numbers in the randomly generated arrays are unique. i.e., there are no duplicated numbers generated.
- Algorithmic efficiency is given using appropriate notation in the report.
- The number of references is adequate, and a corrected format used for both in-text and bibliography citing.
- The generated graph in the report presents the execution time data versus the array sizes in a clear and concise manner to facilitate easy comparison.
- The graph is generated during the runtime using Python programming language.

Commented [BC(oEaC1]: Seems similar to the criteria for 60%. Make the wording consistent with the 60% criteria, e.g. “All criteria (**5a**, **5b**, **5c**, **5d**) specified in section **5** are covered to an excellent standard with competent use technical language.

PREPARATION FOR THE ASSESSMENT

Before attempting this assessment, you should complete the exercises in weeks 1 – 5 on Blackboard

RELEASE DATES AND HAND IN DEADLINE

Assessment Release date: 24/10/2023

Assessment Deadline Date and time: 22/01/2024, 17:00 hours.

Please note that this is the latest time you can submit – not the time to submit!

SUBMISSION DETAILS
You should compress your source code work into a single zip file and submit using the assignments tab on Blackboard. You should also upload your report in a Microsoft Word document and submit in the appropriate section using the assignments tab on Blackboard. **Do not use other compression formats**. For each part of your submission, clearly identify the question you are answering.

HELP AND SUPPORT

- Questions regarding this assessment should be asked through the CO2412 Module Teams Channel "Lab Work and Assessment"
- For support with using library resources, please contact Bob Frost, RSFrost@uclan.ac.uk or SubjectLibrarians@uclan.ac.uk. You will find links to lots of useful resources in the My Library tab on Blackboard.
- If you have not yet made the university aware of any disability, specific learning difficulty, long-term health or mental health condition, please complete a Disclosure Form. The Inclusive Support team will then contact to discuss reasonable adjustments and support relating to any disability. For more information, visit the Inclusive Support site.
- To access mental health and wellbeing support, please complete our online referral form. Alternatively, you can email wellbeing@uclan.ac.uk, call 01772 893020 or visit our UCLan Wellbeing Service pages for more information.
- If you have any other query or require further support you can contact The <i>, The Student Information and Support Centre. Speak with us for advice on accessing all the University services as well as the Library services. Whatever your query, our expert staff will be able to help and support you. For more information , how to contact us and our opening hours visit Student Information and Support Centre.
- If you have any valid mitigating circumstances that mean you cannot meet an assessment submission deadline and you wish to request an extension, you will need to apply online prior to the deadline.

Disclaimer: The information provided in this assessment brief is correct at time of publication. In the unlikely event that any changes are deemed necessary, they will be communicated clearly via e-mail and a new version of this assessment brief will be circulated.	Version: 1