COMP8110 Assignment 2
Design and implementation of a new scheduling algorithm
DUE 11:55pm Friday, 31st October (Week 12)

This assessment item accounts for 30% of the total mark (100) of the unit. It is set to be individual work. Note that we will use a code plagiarism check tool.
Resource management, scheduling in particular is of great practical importance in improving data centre efficiency. In job scheduling, there are a number of performance metrics/objectives and constraints, such as execution time, turnaround time, resource utilisation and costs of execution.

Your task in this Assignment is to design and implement one or more new scheduling algorithms that optimises average turnaround time. Your scheduling algorithm as part of your client is required to schedule jobs to servers in order to minimise the average turnaround time without sacrificing too much of other performance metrics, i.e., resource utilisation and server rental cost. In particular, your algorithm should aim to outperform the baseline algorithms with respect to all performance metrics.

As these objectives are incompatible (conflicting performance objectives), the optimisation of one objective might lead to sacrificing the optimisation of other metrics. For instance, if you attempt to minimise average turnaround time by minimising waiting time, you may end up using more resources resulting in high costs (i.e., server rental costs). Therefore, you have to define the scheduling problem clearly indicating your performance objective(s) and justifying your choice in the report. In addition, you must discuss your algorithm and scheduling results in comparison with baseline algorithms (ATL, FF, BF, FC and FAFC) and their results.

Your client should work for any simulation configuration; that is, you shouldn't assume those sample configurations given before are the only ones used for testing and marking. You may even create your own configurations to more effectively demonstrate the performance of your algorithm; these configurations, if used in your discussion (report), must be included in your submission.

The submission shall be ONLY your **5-page** report. You must use LaTeX (e.g., Overleaf) for your report with the LaTeX template provided. You submit only the PDF file of your report. You must use GitHub Classroom (`https://classroom.github.com/a/ZO44tugQ`) to submit your source code. The last commit pushed to the repository will be marked.

Your scheduling algorithm(s) must be runnable by the markers (your TA and the unit convenor) independently. Your submission must meet the following five requirements:

1. The source code of your client-side simulator must compile without errors and be executable using the auto-run script.[1]
2. Your client must be written in Python, Java, or C.[2]
3. Any necessary information and files, such as README.md[3], a *Makefile*, installation shell script, and user guide, must be included in your Git repository to enable the markers to reproduce the results presented in your report.
4. Your source code must be pushed to your GitHub Classroom repository at the time of submission. More detailed instructions, such as configuration files and the auto-run script, will be provided in due course. Note that your client-side simulator must work with any simulation configuration.
5. You must be available during the main examination period for a demo if necessary, e.g., if the marker is unable to run your code.

"Suggested" headings of the Assignment 2 report are as follows. Your report should be no more than five pages excluding references and appendices. Note that the number of pages for each section specified in the following is only indicative.

1. Introduction (1/2 page): What this report is about, including problem definition, the definition of your objective function, and the goal of this assignment.

[1] The auto-run script will be made available separately in iLearn.
[2] If you have a good reason for using a different language you may request an exemption by emailing comp8110@mq.edu.au.
[3] If you have used a port number other than the default one (50000), you must state that in README.md.

2. Algorithm description (1 - 2 pages; one subsection per algorithm if you have more than one); **NB: You need to provide a simple example scheduling scenario, including a sample configuration, the schedule, the description, and discussion**; this is to 'visualise' how your scheduling algorithm works.
3. Implementation details including data structure(s) used (1/2 - 1 page).
4. Evaluation (1-2 pages): Simulation setup including test cases/configurations, results, comparisons with baseline algorithms, and discussion including pros and cons of your algorithm. Use figures and tables effectively and be space-conscious.
5. Conclusion (1/3 page): summary + what you have found and what you suggest.
6. References

*If you have used GenAI, you must include a **"GenAI Usage Disclosure"** section as an appendix, placed after the references. In this section, you are required to provide a full disclosure of any use of GenAI tools at all stages of the assignment, including for data, code, and writing. In particular, you must provide the links to GenAI usage. This section, along with the references, will not be counted toward the page limit.*

Marking rubric (in percentage)

85 to 100

Work in this band presents a full and comprehensive account of all requirements outlined above. In particular, the efficient and elegant design and implementation of one or more fully functional and robust new scheduling algorithms should be evident in (1) the performance of the algorithm, (2) the source code, and (3) documentation. One algorithm at least should demonstrate superiority over baseline algorithms in its design and performance in terms of all performance metrics. The performance of scheduling algorithm(s) needs to be extensively and thoroughly analysed and justified both qualitatively and quantitatively in appropriate forms, such as tables and comparison charts. The solid understanding of Assignment 2 should be evident in the submission, and the project management, e.g., git commit history.

75 to 84

Work in this band presents a clear account of all requirements outlined above. In particular, the efficient and clear design and implementation of one or more fully functional (robust) new scheduling algorithms should be evident in (1) the performance of the algorithm, (2) the source code, and (3) documentation. One algorithm at least should be sufficiently different from baseline algorithms with its performance equivalent to or better than that of baseline algorithms based on one or more performance metrics without sacrificing any performance metric too much compared to the performance gain for the defined objective. The performance of scheduling algorithm(s) needs to be analysed and justified both qualitatively and quantitively in appropriate forms, such as tables and comparison charts. The solid understanding of Assignment 2 should be evident in the submission, and the project management, e.g., git commit history.

65 to 74

Work in this band presents the good design and implementation of one or more new scheduling algorithms with clear algorithm description and discussion. One algorithm at least should demonstrate some performance advantage over one or more baseline algorithms. The performance of scheduling algorithm(s) needs to be analysed and justified both qualitatively and quantitively in appropriate forms, such as tables and comparison charts. The clear understanding of Assignment 2 should be evident in the submission.

50 to 64

Work in this band presents the adequate design and implementation of at least one new scheduling algorithm with appropriate algorithm description. Appropriate discussions are required in documentation with identification and justification of performance of the scheduling algorithm. The good understanding of Assignment 2 should be evident in the submission.