COSC 2637/2633 Big Data Processing
Assignment 1

Assessment Type	– Individual assignment. – Submit online via Canvas → Assignment 1. – Marks awarded for meeting requirements as closely as possible. – Clarifications/updates may be made via announcements or relevant discussion forums.
Due Date	23:59, 22 August, 2021
Marks	30/100

Overview
Write MapReduce programs which give your chance to understand the complexity of MapReduce programing, the essential components you learned in lectures.

Learning Outcomes
The key course learning outcomes are:

CLO 1.	Model and implement efficient big data solutions for various application areas using appropriately selected algorithms and data structures.
CLO 2.	Analyze methods and algorithms, to compare and evaluate them with respect to time and space requirements and make appropriate design choices when solving real-world problems.
CLO 4.	Explain the Big Data Fundamentals, including the evolution of Big Data, the characteristics of Big Data and the challenges introduced.

Assessment details
In Labs, you have developed a MapReduce program and run it in Hadoop. It is the basic version of word count. In this assignment, you are asked to extend the function. You should use Java to develop MapReduce program (if you want to use other code language, please contact lecturer for approval).

Task 1 – Count words by lengths (5 marks)
Write a MapReduce program to count number of short words (1-4 letters), medium words (5-7 letters) words, long words (8-10 letters) and extra-long words (More than 10 letters); and the partitioner is implemented such that short words and medium words are processed by the same reducer.
Task 2 – Count word with in-mapper combining without preserving state across documents (7 marks)
Write a MapReduce program to count the number of each word where the in-mapper combining (without preserving state across documents) is implemented rather than an independent combiner.
Task 3 – Count word with in-mapper combining with preserving state across documents (8 marks)
Write a MapReduce program to count the number of each word where the in-mapper combining (with preserving state across documents) is implemented rather than an independent combiner.
Task 4 – Performance Analysis (10 marks)
For Task 2 and 3, the performance may be improved using in-mapper combining with preserving state across documents compared without preserving state across documents. Analyze in which situation the performance can be improved more significant, and in which situation less. Verify your analysis by running both implementations in the situations you identified and show the test results (Tip: the total number of <key, value> pairs transferred to reducers).

Submission

Your assignment should follow the requirement below and submit via Canvas > Assignment 1.
Assessment declaration: when you submit work electronically, you agree to the assessment declaration: https://www.rmit.edu.au/students/student-essentials/assessment-and-exams/assessment/assessment-declaration

Format Requirement

Failure to follow the requirements incurs penalty

(a) The codes for all four tasks are entailed in a single Maven project. (1 mark)
(b) Submit the complete Maven project source code in a .zip file (including a standalone jar file). The zip file should be named as sxxxxx_S2_2020.zip (replace sxxxxx by your student ID). (1 mark)
(c) You need include a "README" file in the zip file. In the README, you are asked to specify how to run each task using the standalone jar in Hadoop. (1 mark)
(d) Include your Task 4 analysis and test results in "README" file. (1 mark)
(e) Paths of input file and output file should not be hard-coded. (1 mark)

Functional Requirement

Failure to follow the requirements incurs penalty

(f) For task 1, 2 and 3, use the text file "Melbourne", "RMIT" and "3littlepigs" as the input files and process them all by running the MapReduce program once~~; output file must be stored in /user/sxxxxx/Task# in HDFS (i.e., /user/sxxxxx/Task1 for Task 1)~~. (4x1 mark)
(g) For task 4, you set up the different situations using the text file "Melbourne", "RMIT" and "3littlepigs" as the input files where each text file may have many copies.
(h) Use "StringTokenizer(String str)" when constructing a string tokenizer for the specified string. The tokenizer uses the default delimiter set, which is " \t\n\r\f": *the space character, the tab character, the newline character, the carriage-return character, and the form-feed character*. Delimiter characters themselves will not be treated as tokens. (4x1 mark)
(i) For each task, using Apache log4j log information: (4x1 mark)
   - In the MAP tasks, the log should be "The mapper task of <Your Name>, <student ID>"
   - In the REDUCE tasks, the log should be "The reducer task of <Your Name>, <student ID>"
(j) Your MapReduce program(s) must be well written, using good coding style and including appropriate use of comments. (4x1 marks)

Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic work. It means acknowledging the work of others while developing your own insights, knowledge and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and/or ideas of others you have quoted (i.e. directly copied), summarized, paraphrased, discussed or mentioned in your assessment through the appropriate referencing methods,
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from Internet sites.

If you do not acknowledge the sources of your material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviors, including:

- Failure to properly document a source
- Copyright material from the internet or databases

- Collusion between students

For further information on our policies and procedures, please refer to
https://www.rmit.edu.au/students/student-essentials/rights-and-responsibilities/academic-integrity

Marking Guide

Task 1 Implementation Correctness	0 marks cannot operate, no output, >1 major logic error in code	1 mark 1 major logic error in code	2-3 marks >1 minor logic error in code	4 marks output incorrect due to 1 minor logic error in code	5 marks output correct and no code error
Task-2 Implementation Correctness	0 marks cannot operate, no output, >1 major logic error in code	1-2 mark 1 major logic error in code	3-4 marks >1 minor logic error in code	5-6 marks output incorrect due to non-logic errors or 1 minor logic error in code	7 marks output correct and no code error
Task-3 Implementation Correctness	0 marks cannot operate, no output, >1 major logic error in code	1-2 mark 1 major logic error in code	3-4 marks >1 minor logic error in code	5-7 marks output incorrect due to non-logic errors or 1 minor logic error in code	8 marks output correct and no code error
Task-4 Analysis Correctness	0 marks analysis incorrect, no test, test results do not make sense.	1-3 mark analysis partially correct (<1/3) and test results are partially reasonable (<1/3)	4-6 marks analysis partially correct (>2/3) and test results are partially reasonable (>2/3)	7-9 marks analysis correct but test results with minor issues to support the analysis.	10 marks analysis correct and verified with consistent test results
Functional requirement	Failure penalty on functional requirements detailed in specification				
Format requirement	Failure penalty on format requirements detailed in specification				