University of Brighton

School of Computing, Engineering and Mathematics

Assessment Brief

Module Title:	Data structures and operating systems
Module Code:	CI583
Author(s)/Marker(s) of Assignment	Jim Burton

Assignment No:	1
Assignment Title:	Huffman coding and reflective report
Percentage contribution to module mark:	100% of the overall mark
Weighting of component assessments within this assignment:	n/a
Module learning outcomes covered:	1. Assess how the choice of data structures and algorithm design methods impacts the performance of programs. 2. Choose the appropriate data structure and algorithm design method for a specified application. 3. Demonstrate an understanding of the limitations of/merits of an operating system as a manager of normally scarce resources, 4. Describe and propose solutions to issues arising from the interactions between system and/ or user level components.

Assignment Brief and Assessment Criteria*:* See pages 2 onwards

Date of issue:	29 October 2021
Deadline for submission:	15:00, 13 January 2022
Method of submission:	e-submission via studentcentral. Submit your programming assignment as a **zip file** containing both **your edited version of the** project and **your reflective report** as a Word document or PDF file.
Date feedback provided	10 February 2022

CI583 – Assessment 1, Huffman coding implementation and report

Author: Jim Burton
Version: 1 Date: 16 September 2021

1 Requirement

Download or clone the Java project from the URL below. Implement your solutions to the problems given in the README file in the top level of the project. **You must maintain the project structure as it is. If you submit individual Java source files, marks will be deducted.** The project contains unit tests which you should use to test your progress. **Note that passing the tests may not be a guarantee of correctness or of full marks for correctness.** See the slides from this module and the recommended reading for more information on hashtables, hash functions and prime numbers.

1. **Huffman coding.** Get a copy of the code from https://github.com/jimburton/huffman-java/ by cloning the repository or downloading a zip file that contains it. Follow the instructions in README.md to complete the implementation.

Implementation assessment criteria: 60% of overall mark

1. Correctness [40 marks]

You can test your progress towards a correct solution by running the unit tests supplied. However, a set of tests which all pass is **not a guarantee of full marks** in this area. Firstly, it is possible to make unit tests pass by "cooking the books", i.e. by hard-coding certain values into the application. Secondly, you may create an implementation which "works", i.e. which passes all tests, but which is not a true Huffman coding because you aren't following the algorithm correctly.

To achieve **30-40 marks** your implementation will be correct in all aspects and show a thorough understanding of the theory behind these data structures and algorithms, as well as showing strong programming skills.

To achieve 2**0-29** marks your implementation may pass some but not all unit tests but will show an understanding of some aspects of this problem. Methods that you weren't able to get working may contain pseudo-code in comments, and some credit will be given for this.

If your code does not pass the unit tests and does not show an understanding of the problem you will receive **fewer than 20 marks**, but I will award credit for code that shows some understanding of the theory and implementation of these data structures and algorithms.

2. **Coding style** [20 marks]

To achieve these marks your code must conform to the **Java style guidelines** linked to on studentcentral, and show good ability with the Java programming language. That means that your code will be consistently and conventionally formatted, comments used appropriately and not over-used, variables named appropriately, and the code will be well-structured. You may have shown independent learning by using features of the language not discussed in the module to structure the code more elegantly.

2. **Report**. Write a report not longer than **four A4 sides** that includes at least **two sections**. In Section 1, **describe the complexity of your encode and decode methods** and discusses the applications for this kind of compression technique. (Note that to describe the complexity of these methods you need to consider the complexity of all methods called within them.) Comment on what you would need to add to this implementation in order to produce a fully working compression/decompression tool that stores compressed data in binary codes?

   In Section 2 of the report, **discuss the application of some of the data structures and algorithms you learned about in the first part of the module in the context of operating systems**. This will require independent learning and research on your part. Some of the best places to find information will be the lecture slides from this module and the books on operating systems that can be found in the library, especially *Operating Systems in Depth* (Doeppner) and *Modern Operating Systems* or *Operating Systems: Design and Implementation* (both by Tanenbaum et al). A few non-exhaustive examples of the applications you could discuss are the use of the stack and stack frames in assembly code, the use of a binary algorithm in the Buddy System, the “disc map” data structure in the S5FS filesystem or the use of hash functions in page tables, file systems and directory listings.

Report assessment criteria: 40% of overall mark

1. Discussion of complexity [20 marks]

To achieve **15-20 marks** you must show an excellent understanding of both the theory behind this data structure and of your actual implementation. You will have taken into account the complexity of all methods that are used by the encode and decode methods. You will use and demonstrate an understanding of the correct terminology regarding complexity. **5-15 marks** will be awarded to work that provides less detail but in a way which is nevertheless accurate and which demonstrates an understanding of the question.

2. Data structures **and operating systems** [20 marks]

To achieve **15-20 marks** you will have shown some good independent research and made thorough use of the module material and/or other sources to discover several applications of data structures and algorithms in the domain of operating systems. You will have explained these in a thorough and convincing way in your own words. **5-15 marks** will be awarded to work that provides less detail but in a way which is nevertheless accurate and which demonstrates an

understanding of the question.

In addition to these module-specific marking criteria, I will apply the University of Brighton Grading Descriptors for Undergraduate Work when marking your submission. A copy of this document can be found in the Assessment area for this module on studentcentral.

Any non-original code must be clearly identified using comments within the submitted code and an accurate reference with an appropriate license must be included within the references section of the report

2 Submission

The work must be submitted through the module area on StudentCentral. Submit the **implementation and report** as a single Zip file.

Any work submitted later than 15:00 on **13 Jan 2021** will be treated as late.

3 Deliverables

The deliverables consist of:

3. A functioning Java program meeting the requirements defined above.
4. Report.