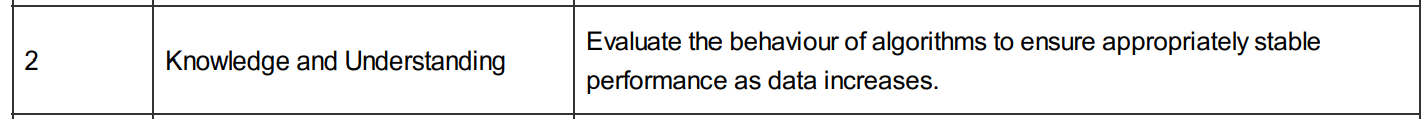


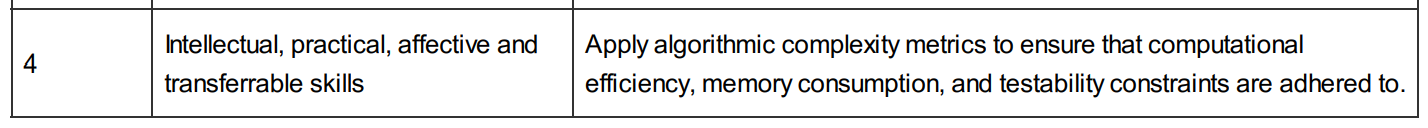
**MOD008310 and MOD008369**

**Algorithm Analysis and Data Structures – Submission 9th April 2024**

A white rectangular box with black text

Description automatically generated





For this assignment part, you will need to complete the following:

1. Choose 2 sorting algorithms and complete the following tasks:
   1. Write, fully commented, code in C# for each algorithm.
   2. Using timer functions, demonstrate the difference in operation time on the following:
      1. 10 elements
      2. 100 elements
      3. 1000 elements
      4. 10000 elements
      5. 100000 elements
   3. Create trace tables for each algorithm showing the sorting of the same 10 elements in the same order
   4. Compare and contrast your findings, including timing and steps required, critically analysing both of the algorithms for efficiency (500 words)

**Grading Criteria**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Insufficient–Grade 0 – 39** | **Basic – Grade 40+** | **Satisfactory – Grade 50+** | **Good – Grade 60+** | **Excellent – Grade 70+** |
| **Program Functionality and Execution 40%** | Majority of required functionality not working. Confusing user experience | Most basic functionality working. User experience inconsistent. Messages/instructions unclear or not enough of them. Frequent exceptions (crashes) and/or unhelpful error messages | All basic functionality working. User experience generally easy to follow but with some inconsistencies. Some useful instructions. Occasional exceptions (crashes) and/or unhelpful error messages. | All listed intermediate functions implemented working. Usable program. Majority of errors handled without crashing. Clear messages and instructions to the user. | All listed advance functions implemented and working. Simple and intuitive to use. Clear meaningful instructions and error messages. Robust and reliable, with all exceptions handled without crashing. |
| **Program Architecture and Authorship 20%** | Little or no attempt at solving the problem or most code submitted is auto generated. Lacking in basic topics as covered in module. | Code confused and difficult to follow intended behaviour. Limited adherence to good coding practices or code which is otherwise difficult to maintain. Adequate attempt at implementing taught material but with significant flaws or inefficiencies. | Code well laid out and presented though might be somewhat difficult to follow intended behaviour. Some adherence to good coding practices. Attempt at implementing taught material with limited success. Sound implementation of sorting algorithms with some adherence to algorithmic efficiency. | Code easy to follow. Maintainable and well presented. Sound application of taught material with some evidence of researching around the subject to develop skills. Good implementation of sorting algorithms with some attention to efficiency and algorithmic correctness. | Excellent adherence to coding standards i.e., indentation, use of comments, naming and capitalisation. Code easily maintainable and easy to follow. Appropriate use of ‘new’, ‘untaught’ features. Excellent implementation of sorting algorithms demonstrating a deep understanding of algorithmic principles and efficiency. Highly optimised code. |
| **Documentation (40%)** | Little or no adherence documentation | Basic documentation and little or inappropriate commenting | Satisfactory documentation. Appropriate commenting and variable names. | Good documentation. With intuitive commenting and variable names. | Professional style documentation with excellent commenting. Very readable code that is easily understood. |