ASSESSMENT AND INTERNAL VERIFICATION FRONT SHEET
(Grouped Criteria)

(Note: This version is to be used for an assignment brief issued to students via Classter)

Course Title	**BSc. Software Development** **BSc. Creative Computing** **BSc. Applied Data Science** **BSc. Digital Games Development**		**Lecturer Name & Surname**	**Andrew Cortis** **Kassandra Calleja**	
Unit Number & Title	ITSFT-506-1608 – Data Structures & Algorithms				
Assignment Number, Title / Type	2 – Advanced Algorithms Implementation & Evaluation / Home				
Date Set	12/05/2025	**Deadline Date**	09/06/2025		
Student Name	Hayden Zammit	**ID Number**	008106L	**Class / Group**	SWD6.1B

Assessment Criteria	Maximum Mark
KU2.7 : Explain the process known as the Fisher Yates Shuffle. *KU3.1: Show analysis of estimate running times and compare implementation of efficient algorithms with inefficient algorithms.* *AA2.3: Produce an algorithm for Graphs or Tree structures. Also compute the best, worst and average case times.* *AA2.4: Produce an algorithm using the Binary Tree structure. Also compute the best, worst and average case times.* *AA2.5: Produce an algorithm using the queue data structure to prioritize data. Use a data structure such as a Heap and the Heapsort algorithm.* *SE2.6: Evaluate the applications of pseudo random number generator.* *SE2.8 : Implement three different sorting algorithms. Predict the rate of processing and evaluate and justify application for each algorithm.* *SE4.1 : Evaluate the features algorithms in relation to their correctness, proof and intractability.* ***Total Mark***	61

Notes to Students:

- This assignment brief has been approved and released by the Internal Verifier through Classter.
- Assessment marks and feedback by the lecturer will be available online via Classter (http://mcast.classter.com) following release by the Internal Verifier
- Students submitting their assignment on Moodle/Turnitin will be requested to confirm online the following statements:

Student's declaration prior to handing-in of assignment

- I certify that the work submitted for this assignment is my own and that I have read and understood the respective Plagiarism Policy

Student's declaration on assessment special arrangements

- I certify that adequate support was given to me during the assignment through the Institute and/or the Inclusive Education Unit.
- I declare that I refused the special support offered by the Institute.

MCAST

ITSFT-506-1608
Data Structures & Algorithms

First Year BSc. 2024-2025

Assignment 1
Advanced Algorithms Implementation & Evaluation

Assignment Guidelines

Read the following instructions carefully before you start the assignment. If you do not understand any of them, ask your lecturer.

- This is a **HOME** Assignment to be completed by the **DEADLINE SPECIFIED BY LECTURER ON VLE**.
- The assignment consists of **2 Sections and CARRIES 61marks**; all tasks must be attempted.
- Please note that **ALL WORK** must be handed in by the stipulated deadlines. **LATE ASSIGNMENTS WILL NOT BE ACCEPTED.**
- The assignment sheet and assignment coversheet should be fully completed with all the necessary details. Note that **assignments handed in without the assignment cover sheet are considered as not submitted.**
- Assignments ***must* be handed in as a soft copy uploaded on Moodle** by the stipulated deadline.
- Any **references should be listed and quotes should be paraphrased *properly***. Unless listed and paraphrased properly the assignment will be regarded as plagiarized. **Referencing should be carried out using IEEE Style** Referencing Notation.

- **<u>Copying is strictly prohibited and will be penalized</u>** in line with the College's disciplinary procedures.

Task 1 31marks

Empirical Analysis of Sorting Algorithms

(AA2.3, AA2.4, AA2.5, SE2.8)

Scenario:

A simple Order Management System presents users with the following 3 options:

1. View all Orders (sorted in Different Ways)
2. Search Order by ID
3. Print Most Urgent Orders

The following system has already been partly implemented in the part solution provided, in the project named Task 1 - Searching and Sorting Algorithms. All required classes have been created and Program.cs contains display menu options and to call the necessary objects and their methods to test your implementation. You must implement the methods that throw new NotImplementedException() with the required code. <u>The given structure and requirements must be adhered to for all implementations. No variations from these are accepted, even if the implementations work.</u>

A detailed description of the functionalities to be implemented can be found below:

1. View All Orders (SE2.8, 10marks)

Upon selecting this option, users are presented with a sub-menu allowing them to view a list of all the orders placed, sorted in one of 3 different ways. The options provided by the sub-menu are as follows:

A. SORT BY ORDER ID

Displays orders sorted by their ID in ascending order.

The sorting must be carried out using Quick Sort using the left-most element as the pivot. *(2 marks)*

B. SORT BY DATE PLACED

Displays orders sorted by date placed in most recent first order.

The sorting must be carried using Merge Sort. *(2 marks)*

C. SORT BY DELIVERY DATE

Displays orders sorted by their delivery date in most recent first order. *(2 marks)*

The sorting can be implemented using any other sorting algorithm, including in-built sorting algorithms.

WRITTEN TASK FOR CHOSEN SORTING ALGORITHM IN (C)

This is a written task to be submitted via Word Document or PDF along with your coversheet and any other written tasks. No implementation is required for this part, apart from the implementation of the sorting algorithm, described in point (c) above.

Specify the sorting algorithm chosen for (c) and state its asymptotic speed in terms of the Best, Average and Worst Case. *(1 mark)*

Compare the asymptotic speeds above to those of the other 2 sorting algorithms implemented i.e. Merge Sort and Quick Sort and state which algorithm, would be the most ideal one for sorting objects *(3 mark)*

IMPORTANT NOTE:

The above options require the implementation of the specified sorting algorithms within the Sort() method in the provided classes MergeSort.cs, QuickSort.cs and CustomSort.cs.
This method must not sort the original array supplied as input, instead it must return a sorted copy of this array.
Additional methods may be added but these must be private and called within the specified Sort() method.

2. Search Order by ID (AA2.4, 7marks)

Upon selecting this option, users will be prompted to search for an order by its ID; the details of the order matching the ID supplied are displayed to the user. If no order is found for the given ID, an error message is displayed.

Implement the required Search functionality in an efficiency manner using a Binary Search Tree by implementing the code for the following methods in the BinarySearchTree.cs class provided:

A. VOID BUILD(LIST<ORDER> ORDERS) (3 marks)

This method should build a Binary Search Tree based on the list of orders passed as parameters; the first order in the list should be placed at the root of the tree being built.

B. ORDER GET(GUID ORDERID) (3marks)

This method should search for the order object in the tree having the Guid passed as parameter and return the object if found.

The method should return null if no order object matching the given id is found in the tree and guarantee an average case speed of O(log n), assuming that orders data is randomly generated and that orders are not sorted in any way. No additional work (written or otherwise) is required for this but your implementation must be written in such a way that this asymptotic speed is observed. (1 mark)

3. Print Most Urgent Orders (AA2.3 & AA2.5, 14marks)

Upon selecting this option, the user will be presented with the 5 most urgent orders i.e. the 5 orders having the earliest delivery.

This functionality is to be implemented by building a Priority Queue which gives priority to elements having the earliest delivery date. The Priority Queue must be built in line with the following specifications :

A. Using an Appropriate Heap Structure for the Priority Queue

The Priority Queue must use a Heap (minheap or maxheap, whichever is most suitable) as the underlying data structure where order elements are stored based on their delivery date. (AA2.5, 1mark)

B. Implementing Necessary Heap Operations

For the heap to store its elements as required, the implementation of the following 2 methods in the provided Heap.cs classis required.

i. void Insert(Order order) (AA2.3, 3marks)

This method should insert the Order object passed as parameter in the correct position in the heap, based on its delivery date.

ii. Order Remove() (AA2.3, 4marks)

This method should return the Order currently at the top of the Heap and re-organize the Heap so that the required heap-order is maintained. You may create additional methods besides the given Insert() method but these method must be private and called within the Remove() method provided.

C. Using Heap Operations to Implement the Priority Queue

The two operations required by the Priority Queue, must use the Insert() and Remove() operations defined within the Heap as follows:

i. void Build(List<Order> orders) (AA2.5, 2marks)

This method should call the appropriate method from the Heap class to populate the Priority Queue with orders contained within the list passed as parameters.

It is important that the orders are stored in such a way that orders having the most earliest delivery date are stored towards the start (head) of the Priority Queue.

Ensure that even in the worst case, populating the priority queue does not exceed O(n log n) time. No additional work (written or otherwise) is required for this task, but your implementation must be written in such a way that this asymptotic speed is observed.

ii. List<Order> GetMostUrgentOrders(List<Order> orders) (AA2.5, 4marks) This

method should call the appropriate method from the Heap class to get the order with the earliest delivery date i.e. the one currently at the start (head) of the Priority Queue

This should be done 5 times in total and each time, the order retrieved should be added to a list of orders which is to be returned at the end.

Ensure that even in the worst case, retrieving the most urgent order from the priority queue does not exceed O(log n) time. No additional work (written or otherwise) is required for this task, but your implementation must be written in such a way that this asymptotic speed is observed.

Task 2 30marks

Random Number Generation & Empirical Analysis

(KU2.7, KU3.1, SE2.6, SE4.1)

Scenario:

The implementation and evaluation of a pseudo random number generator (PRNG) is required for this part of the assignment. A structure for this has already been partly implemented in the part solution provided, in the project named Task 2 - Random Number Generation; this is to be used as specified below. Note that the given structure and requirements must be adhered to for all implementations. No variations from these are accepted, even if the implementations work.

1. Implement a Pseudo Random Number Generator (PRNG) (SE2.6, 10marks)

Implement the SplitMix64 algorithm to implement a PRNG capable of generating a random number within a given range (both min and max values included). Pseudocode for implementing the SplitMix64 algorithm can be found below.

Your implementation must reflect the pseudocode provided, no other variations will be accepted, even if the PRNG works.

```
Class SplitMix64


Initialize state = current time in milliseconds // state is a 64-bit unsigned integer



Function Next(min,max):          // both min and max are 64-bit unsigned integers

   z  = state +  a large constant   // large constant is 0x9E3779B97F4A7C15

   Update state to be equal to z


   Update z to be (z XOR (z shifted right by 30 bits)) * 0xBF58476D1CE4E5B9


   Update z to be (z XOR (z shifted right by 27 bits)) * 0x94D049BB133111EB


   Update z to be z XOR (z shifted right by 31 bits)

        Modify z so that it is in the range min to max
   return z
```

2. Demonstrate the PRNG's correctness & whether it is intractable or not (KU3.1, SE4.1, 15marks)

A. Prove the PRNG's correctness (SE4.1, 7marks)

In Program.cs write code to generate 100 random numbers in the range 1 to 1000 (both numbers included) using the implemented SplitMix64 algorithm and store them in a list (or array).
(1mark)
Prove that the implemented SplitMix64 algorithm works correctly by writing further code to ensure that all numbers generated and stored within the list are:

- in the range 1 to 1000.
  (2marks)
- not sorted in ascending order.
  (2marks)
- not sorted in descending order.
  (2marks)

B. Demonstrate whether the PRNG is intractable or not (KU3.1, 5marks)

Use Empirical Analysis to demonstrate whether the implemented SplitMix64 algorithm is intractable or not.

To do so, time how long it takes the RNG to produce the following amount of random numbers:
1000, 10000, 100000, 1000000. (2marks)

Ensure that all necessary care has been taken to obtain accurate timings. (1mark)

Use the timings obtained to plot a Log-Log Graph with Excel. (2marks)
Ensure that you include a screenshot of the timings obtained when running empirical analysis.

C. Written Task for PRNG Correctness and Intractability (SE4.1, 3marks)

Answer the following questions:

i. Is your PRNG implementation correct? (SE4.1, 1mark) Base your answer on the results obtained for code requested in point A above.

ii. Is your PRNG implementation intractable? (SE4.1, 2marks) Base your answer on the Log-Log Graph produced in point B above.

This is a written task to be submitted via Word Document or PDF along with your coversheet and any other written tasks. No implementation is required for this part, apart from the implementation of the proofs described in points A and B above.

D. Design Task for PRNG IMPROVEMENT (KU2.7, 5marks)

PRNGs are said to suffer from 2 main problems: they are both periodic and deterministic. Shuffling the sequence of pseudo-random numbers generated can help to reduce this.

Draw a flow chart which illustrates how the Fisher Yates algorithm would shuffle an array of random numbers. The algorithm receives as input an array of random numbers and returns a shuffled version without affecting the original array supplied.

The flow chart created must:

i. Reflect the Fisher-Yates Shuffle Algorithm Correctly (3 marks) ii.
Use correct Flow Chart Notation (1 mark) iii.
Be neatly drawn and easy to read and interpret (1 mark)

Use draw.io or an equivalent diagramming tool to create the flow chart and submit the flowchart created as both an image file (.png) and as a draw.io file (or equivalent). Also include the flowchart in the Word Document or PDF along with your coversheet and any other written tasks.

End of Assignment