

Liverpool John Moores University

Title: DATA STRUCTURES AND ALGORITHMS
Status: Definitive
Code: **5133COMP** (126288)
Version Start Date: 01-08-2021

Owning School/Faculty: Computer Science and Mathematics
Teaching School/Faculty: Computer Science and Mathematics

Team	Leader
Sorren Hanvey	Y
Aine MacDermott	

Academic Level: FHEQ5 **Credit Value:** 20 **Total Delivered Hours:** 55
Total Learning Hours: 200 **Private Study:** 145

Delivery Options

Course typically offered: Semester 1

Component	Contact Hours
Workshop	55

Grading Basis: 40 %

Assessment Details

Category	Short Description	Description	Weighting (%)	Exam Duration
Artefacts	AS1	Design and implementation of software	60	
Report	AS2	Report on the efficiency of algorithms	40	

Aims

This is a practical, applied Software Engineering module with the aim of introducing the student to the fundamentals of Abstract Data Types and complexity of operations on ADTs followed by an implementation-based exploration of common data structures and operations, their implementations and applications

Learning Outcomes

After completing the module the student should be able to:

- 1 Explain a range of fundamental data structures and their operations
- 2 Analyse fundamental algorithms' complexity as applied to a range of ADT implementations
- 3 Evaluate data structures in a given problem domain
- 4 Implement standard ADTs using both primitive language and library resources
- 5 Synthesise appropriate algorithms and data structures to fulfil a problem specification

Learning Outcomes of Assessments

The assessment item list is assessed via the learning outcomes listed:

Implementation of software	3	4	5
Report	1	2	

Outline Syllabus

Abstract Data Types and common implementation strategies:

Linear ADTs:

Lists (Arrays, Linked Lists)

Stacks, Queues

Non-Linear ADTs:

Trees, Binary Trees, BSTs

Maps (ListMaps, BSTMaps, HashMaps)

Graphs

Algorithms for structure operations; insert, remove, retrieval

Algorithms for structure navigation; search and sort

Algorithm types: iterative and recursive

Relationship between ADTs and computing fundamentals (e.g. Stack, Queue)

Use of Big O notation to specify time complexity for simple algorithms

Using a program debugger to monitor program state, and halt/control execution as required.

Use of a program debugger to inspect the call stack and stack frames

Learning Activities

Workshops, Directed Study Tasks

Notes

This module is a technical, skills-focused module. It will require previous experience in programming. It will build on existing programming-based skills such as problem / functional decomposition and the use of an IDE to develop and test programs. Basic

operational familiarity with a debugger will be assumed but reinforced and built on during this module.