Liverpool John Moores University

Title:	DATA STRUCTURES AND ALGORITHMS IN SECURITY
Status:	Definitive
Code:	**5131COMP** (125931)
Version Start Date:	01-08-2019
Owning School/Faculty:	Computer Science
Teaching School/Faculty:	Computer Science

Team	Leader
Aine MacDermott	Y

Academic Level:	FHEQ5	**Credit Value:**	20	**Total Delivered Hours:**	57
Total Learning Hours:	200	**Private Study:**	143		

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	40	
Exam	AS2	Examination	60	2

Aims

This is a practical, applied module with the aim of introducing the student to the fundamentals of Abstract Data Types and complexity of operations on ADTs, using a case study in relation to security problems. Emphasise the importance of ADT and algorithms in building software security solutions. Gain experience implementing common data structures and operations to solve security problems.

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' security implications in a given problem domain
4	Apply library resource-based implementations of common ADTs
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:

Design and implementation of s	3	4	5
Examination	1	2	

Outline Syllabus

Case study of ADTs in relation to security problems
Abstract Data Types and common implementation strategies: Linear ADTs: Lists (Arrays, Linked Lists) Stacks, Queues
Non-Linear ADTs:
Trees, Binary Trees, BSTs
Hashmaps and Dictionaries
Graphs
In-memory representations of data and program; memory protection and security vulnerabilities (e.g. buffer overflows)
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

Learning Activities

Workshops and 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.