Liverpool John Moores University

Title:	DATA STRUCTURES AND ALGORITHMS
Status:	Definitive
Code:	**5550NCCG** (129513)
Version Start Date:	01-08-2021
Owning School/Faculty:	Computer Science and Mathematics
Teaching School/Faculty:	Nelson Campus

Team	Leader
Silvester Czanner	Y
Robert Askwith	

Academic Level:	FHEQ5	**Credit Value:**	20	**Total Delivered Hours:**	60
Total Learning Hours:	200	**Private Study:**	140		

Delivery Options
Course typically offered: S1, S2 and NS2 (S2 for Jan)

Component	Contact Hours
Lecture	60

Grading Basis: 40 %

Assessment Details

Category	Short Description	Description	Weighting (%)	Exam Duration
Practice	Exercise	Algorithm Implementation Exercise	50	
Exam	Exam	Written Exam (90 mins)	50	

Aims

This module introduces students to data structures and how they are used in algorithms, enabling them to design and implement data structures. It introduces the specification of abstract data types and explores their use in concrete data structures. Based on this knowledge, students should be able to develop solutions by specifying, designing and implementing data structures and algorithms in a variety of programming paradigms for an identified need.

Learning Outcomes

After completing the module the student should be able to:

1 Examine abstract data types, concrete data structures and algorithms.
2 Specify abstract data types and algorithms in a formal notation
3 Assess the effectiveness of data structures and algorithms
4 Design, implement and test an algorithm to meet a given specification

Learning Outcomes of Assessments

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

Algorithm Implementation	4		
Written Exam	1	2	3

Outline Syllabus

Abstract Data Types. Formal specification of ADTs. Data structures: e.g. Array; set; stack; queue; list; tree; types e.g. active, passive, recursive.

Algorithm types and examples.

Design specification: Specify ADTs using formal notation, Issues e.g. complexity in software development; design patterns, parallelism; interfaces; encapsulation, information hiding, efficiency. Creation: Pre-conditions, post-conditions, error-conditions

Implementation: Data structures; e.g. multidimensional arrays, linked lists, stacks, queues, trees, hash table, heap, graph

Algorithms; sorting, searching, tree traversal, list traversal, hash functions, string manipulation, scheduling and recursive algorithms; using handle, pointer, class, methods; using an executable programming language.

Use of data structure libraries; selection of data structures; theoretical analysis; asymptotic analysis; size of N, Big O notation. Algorithm effectiveness: Run time benchmark, compiler/interpreter dependencies, resource usage, degree of parallelism, time, space, power performance, efficiency of garbage collection.

Learning Activities

Lectures
These will not normally be traditional didactic lectures in which the student plays little active part, but will be delivered in small groups of up to 20

Notes

-