Module 8: Programming and Data Structures

Stage	1
Semester	2
Module Title	Programming and Data Structures
Module Number/Reference	8
Module Status (Mandatory/Elective)	Mandatory
Module ECTS credit	5
Module NFQ level (only if applicable)	8
Pre-requisite Module Titles	None
Co-requisite Module Titles	None
Is this a capstone module? (Yes or No)	No
List of Module Teaching Personnel	Mr Barry Denby Mr John Hannon

Contact Hours				**Non-contact Hours**			**Total Effort (Hours)**
Lecture	**Practical**	**Tutorial**	**Seminar**	**Assignment**	**Placement**	**Independent work**	
24	18			36		22	100

Allocation of Marks (Within the Module)

	Continuous Assessment	**Project**	**Practical**	**Final Examination**	**Total**
Percentage contribution	50%			50%	**100%**

Intended Module Learning Outcomes

Upon successful completion of this module, you should be able to:

1. implement and use data structures introduced on the course;
2. apply object-oriented methods when designing data structures;

3. implement both recursive and non-recursive solutions to classical data structure problems;
4. implement both linear and non-linear data structures;
5. analyse simple algorithms using asymptotic analysis;
6. compare the efficiency of algorithms solving similar problems;
7. implement algorithms on data structures and relate these to realistic problems.

Module Objectives

As in all programming modules, a key objective is the acquisition, on behalf of the learner, of good software engineering skills and the application of these skills to the design and implementation of software components. At the heart of all software design is the implementation of appropriate data structures that provide efficient data models for the problem at hand. Learners develop an in depth knowledge of the standard data structures: stacks, queues, sets, bags and maps; and also learn to implement these using both linear (linked lists, arrays) and non-linear (binary search trees) data structures.

Module Curriculum

Recursion

- Concept of recursion, recursive functions, recursion over sequences, tail recursion.
- Divide and conquer algorithms.

Analysis of algorithms

- Counting and calculating the time complexity of an algorithm.
- Basic ideas and definitions of asymptotic analysis.
- Big O notation and its application to the evaluation of temporal cost of algorithms.
- Comparing performance using big O notation.
- Basic time and space analysis.

Sorting

- Simple sorting algorithms: insertion, selection, bubble.
- QuickSort. Merge Sort. Heapsort.
- Analysis of sorting algorithms using big O notation.
- Sorting in linear time.

Dynamic Data structures

- Constructing lists using singly linked lists and doubly linked lists.
- Using these linear data structures to implement classes that encapsulate: stacks, queues, priority queues and sets.
- Problem solving with these data structures.

- Concept and definition of Hashtables.
- Implementing classes that encapsulate a hash table and also the use of hash tables in implementing sets.
- Concept and definition of Trees: representing rooted trees, binary search trees, query, insertion, deletion, traversal.
- Optimising the performance of binary trees with avl trees and black-red trees and B-trees. Implementing data structures with trees.

Primary Reading

Mullins, A. Data Structures and Algorithms in Java, Griffith College, 2012.

Goodrich, M.T. & Tamassia, R. *Data Structures and Algorithms in Java (4th Edition)*, Wiley, 2005

Additional Reading

Dasgupta, S., Papadimitriou, C. & Vazirani, U. *Algorithms (1st Edition)*, McGraw-Hill, 2006

Weiss, M. A. *Data Structures and Algorithm Analysis in Java (2nd Edition),* Addison Wesley, 2006

Wirth, N. *Algorithms + Data Structures = Programs*, Prentice Hall, 1976

Module Learning Environment

Accommodation
Lectures are carried out in classrooms / lecture halls in the College. Lab tutorials are carried out in computer labs throughout the Campus. All have the language software required to deliver the programme.

Library

All learners have access to an extensive range of physical and electronic (remotely accessible) library resources. The library monitors and updates its resources on an on-going basis, in line with the College's Library Acquisition Policy. Lecturers update reading lists for this course on an annual basis as is the norm with all courses run by Griffith College.

Module Teaching and Learning Strategy

The module is delivered through a combination of lectures and practical lab programming sessions. The learners complete a series of worksheets throughout the module that are directly related to the material covered in lectures. The emphasis is on developing sound software engineering skills in practical programming based on theoretical knowledge.

Module Assessment Strategy

The module assessment consists of a series of continuous assignments and a final examination. Each week learners are required to complete a series of programming tasks that relate to the material covered in lectures. The practical lab sessions are used to enforce concepts covered in the lectures and the worksheets are used to ensure that learners are keeping up with the material as it is delivered. Lab sessions are also used to deal with issues emerging from the worksheets. All work submitted by learners is assessed and comments are given to individual learners. All assessment goes to giving a final grade for the work completed by the learner.

Element No	Weighting	Type	Description	Learning Outcome assessed
1	50%	Weekly Lab Book Submission	A series of weekly lab books designed to teach programming concepts	1-7
2	50%	Closed Book Examination	End of Module Examination	1-7