SC1007 Data Structures and Algorithms

Academic Year	AY2020		**Semester**	1				
Course Code	SC1007							
Course Title	Data Structures and Algorithms							
Pre-requisites	SC1003 Introduction to Computational Thinking and Programming							
Pre-requisite for	SC2001 Algorithm Design and Analysis							
No of AUs	3							
Contact Hours	Lectures	26 without LAMS lectures, 13 with LAMS lectures	Tutorials	6	Lab	16		
Proposal Date	12 February 2020							

Course Aims

This course aims to (i) teach the concepts, implementations and applications of data structures such as arrays, linked lists, stacks, queues and trees that are important for building efficient algorithms; (ii) provide an introduction to algorithm analysis and design. These are essential for future computer science and computer engineering courses.

Intended Learning Outcomes (ILO)

By the end of this course, the student would be able to:

1. Select appropriate data structures such as arrays, linked lists, stacks, queues and trees and implement algorithms to solve real world problems using these data structures in C.
2. Conduct complexity analysis of simple algorithms.
3. Select and implement appropriate search algorithm (sequential search, binary search, search using hash tables) as part of a problem solution in C. Compare the efficiencies of these search algorithms.
4. Select and implement appropriate graph traversal algorithm (DFS, BFS) as part of a problem solution in C. Analyse the complexities of these graph traversal algorithms.

Course Content

	Topics
1	**Introduction & dynamic memory allocation** Static vs dynamic memory allocation. 'Heap' management and 'garbage collection'. Run-time memory protection. Overview of node-based data structures.
2	**Linked Lists** Linked List structures, doubly linked lists, circular lists. Their implementations in C and examples that use lists.
3	**Abstract Data Types and Their Implementation** Stacks, queues, priority queues. Their implementations using linked lists. Use of stacks to evaluate arithmetic expressions and use of queues, priority queues to schedule jobs.
4	**Tree Structures** Overview of hierarchical/non-linear data structures. Tree structures and their implementations. Binary vs general trees. Tree traversal: pre-order, in-order, post-order. Expression trees: representation and evaluation. AVL trees and their balancing.

5	**Introduction to algorithms** What is an algorithm? Important problem types in computing. Algorithm design strategies.
6	**Analysis of Algorithms** Time and space complexities of algorithms. Analyzing basic program constructs. Best case, worst case and average case time complexity analysis. Deducing recurrence relations for time complexity of recursive algorithms. Solving elementary recurrence relations. Asymptotic time complexity analysis. Big-Oh, big-Omega, and big-Theta notations. Common Complexity Classes. Basic techniques for proving asymptotic bounds. Space Complexity. Faster computer or faster algorithms.
7	**Searching** General exhaustive search. Iterative and recursive sequential search algorithms. Binary search, its invariance, and complexity. Open and closed address hashing using linear probing and double hashing collision resolution techniques. All algorithms covered are accompanied by asymptotic complexity analysis.
8	**Graph Representations and Searching** Basic graph representation methods, adjacency lists, and adjacency matrix, Systematic graph traversals with breadth-first search (BFS) and depth-first search (DFS) algorithms. A generic backtracking algorithm and its complexity. Eight-queen, Maze-Search.
	Check for Hours

Assessment (includes both continuous and summative assessment) CE/CZ1107

Component	Course LO Tested	Related Programme LO or Graduate Attributes	Weightage	Team/ Individual	Assessment Rubrics
1. Lab assignments	1,3,4	a, b, c, d, e, l	40%	Individual	See appendix 1
2. Lab tests	1,3,4	a, b, c, d, e, l	40%	Individual	
3. Quiz	1, 2, 3, 4	a, b, j	20%	Individual	
Total			100%		

Appendix 1: Assessment Criteria

Lab assignments and tests. Lab assignments involve the submission of programming code for grading. First, programming questions related to LO 1, LO 3 and LO 4 from lab assignments will be set and given to individual students. Lab test problems will be given to students at the beginning of the tests. Then, you will submit your answer code within a specified duration. For grading, we focus on marking based on the correctness of the programming logic to solve the given problem. Questions will be designed carefully to test the different concepts in each topic of the course. In addition, it is also important to note that programs will only be meaningful if they can run correctly. The marking will be carried out using the automated marking mechanism in the online system according to test cases. As such, the assessment rubrics are given as follows.

Score	Interpretation
5	Correct code – The submitted code is able to compile and run correctly on all the test cases
2-4	Partially correct code – The submitted code is able to compile and run correctly on some test cases. The score will be allocated proportionally according to the number of test cases which can be run correctly.
1	Incorrect code or non-compilable code - The submitted code is unable to run correctly on all the test cases, or the submitted code is unable to compile.

Note: If you were found cheating or plagiarism in assignments or lab tests, your score for that assignment or lab test will be zero. In addition, you will also be subjected to disciplinary action according to University regulations.

Quizzes. Written quizzes will be marked in similar way to examinations.