| | | | | | |
|---|---|---|---|---|---|
| Course Title | Data Structures and Algorithms | | | | |
| Course Code | CSC615 | | | | |
| Course Type | Compulsory | | | | |
| Level | Master (2nd Cycle) | | | | |
| Year / Semester | 1st Year / 1st Semester | | | | |
| Teacher's Name | TBA | | | | |
| ECTS | 10 | Lectures / week | 3 Hours / 14 weeks | Laboratories / week | N/A |
| Course Purpose and Objectives | The course will introduce students to the notation, terminology, and techniques underpinning the study of algorithms. To introduce basic data structures and associated algorithms. To introduce standard algorithmic design paradigms and efficient use of data structures employed in the development of efficient algorithmic solutions.<br>Structures like arrays, stacks, queues, linked lists, trees and graphs will be discussed and analyzed. Algorithms will be developed that operate and manipulate these structures efficiently. Analysis of time-space complexity of algorithms is discussed. | | | | |
| Learning Outcomes | Upon successful completion of the course, students will be able to:<br>• Analyze program time complexity and express it in big-Oh, Omega and Theta notation.<br>• Classify and evaluate different data structures, both linear and non-linear.<br>• Generate programs that use abstract data structures to solve computational problems.<br>• Choose and apply appropriate algorithms to solve computational problems.<br>• Describe the principles of and apply a variety of data structures and their associated algorithms;<br>• Describe standard algorithms, apply a given pseudo code algorithm in order to solve a given problem, and carry out simple asymptotic analyses of algorithms;<br>• Choose and justify the use of appropriate data structures to enable efficient implementation of algorithms;<br>• Use critical thinking, critical analysis and problem-solving to provide efficient solutions of real life problems.<br>• Design and develop efficient algorithms for real problems that incoporate appropriate data structures with suitable algorithmic methods | | | | |
| Prerequisites | None | Co-requisites | CSC600 | | |
| Course Content | **Objective:**<br>The course will introduce students to the notation, terminology, and techniques underpinning the study of algorithms. To introduce basic data structures and associated algorithms. To introduce standard | | | | |

| | |
|---|---|
| | algorithmic design paradigms and efficient use of data structures employed in the development of efficient algorithmic solutions.<br>Structures like arrays, stacks, queues, linked lists, trees and graphs will be discussed and analyzed. Algorithms will be developed that operate and manipulate these structures efficiently. Analysis of time-space complexity of algorithms is discussed.<br><br>**Description:**<br><br>**Introduction:** Software Quality, Correctness, Reliability, Robustness, Usability, Maintainability, Reusability , Portability, Efficiency. Data Structures: A Physical Example<br><br>**Analysis of Algorithms:** Algorithm Efficiency, Growth Functions and Big-OH Notation, Comparing Growth Functions, Determining Time Complexity, Analyzing Loop Execution.<br><br>**Linked Structures:** A linked list Abstract Data Type (ADT) References as Links, Managing Linked Lists, Accessing Elements, Insertion, Deletion, Doubly Linked Lists.<br><br>**Lists**: A List ADT, Iterators, Adding Elements to a List, Using Ordered Lists: Tournament Maker, The Josephus Problem, Implementing Lists: With Arrays, with linked lists<br><br>**Stacks**: A Stack ADT, Accessing a stack. The push, pop and peek Operations. Using stacks. Implementing stacks: With Links, with arrays.<br><br>**Queues**: A Queue ADT, Using Queues. Accessing a queue: the enque Operation, the dequeue Operation, Implementing Queues: With Links, With Arrays.<br><br>**Recursion**: Recursive Thinking, Infinite Recursion, Recursion in Math, Recursive Programming, Recursion versus Iteration, Direct versus Indirect Recursion. Using Recursion: Traversing a Maze, The Towers of Hanoi. Analyzing Recursive Algorithms.<br><br>**Sorting and Searching**: Searching, Static Methods, Linear Search, Binary Search, Comparing Search Algorithms.<br>Sorting: Selection Sort, Insertion Sort, Bubble Sort, Quick Sort, Merge Sort. Greedy and Divide and Conquer algorithmic techniques, Hashing.<br><br>**Trees**: Trees, Tree Classifications, Strategies for Implementing Trees. Tree Traversals: Preorder Traversal, Inorder Traversal, Postorder Traversal, Level-Order Traversal,<br>Using Binary Trees: Expression Trees, Implementing Binary Trees with Links.<br><br>**Binary Search Trees**: A Binary Search Tree, Implementing Binary Search Trees: With Links, arrays: The addElement Operation, the removeElement.<br>Balanced Binary Search Trees: Right Rotation, Left Rotation, Rightleft Rotation, Leftright Rotation. |

| | |
|---|---|
| | Implementing Binary Search Trees: AVL Trees, red-black trees.<br><br>**Priority Queues and Heaps**: A Heap, The addElement Operation, The removeMin Operation, The findMin Operation. Using Heaps: Priority Queues. Implementing Heaps: With Links: the addElement Operation, The removeMin Operation, The findMin Operation.<br>Multi-way Search Trees: Combining Tree Concepts, 2-3 Trees, 2-4 Trees, B-Trees.<br><br>**Graph Theory:** What a graph, what a PATH and a CIRCUIT are, directed and undirected graphs, networks.<br>Common Graph Algorithms: Traversals, breadth- and depth-first search in graphs, Testing for Connectivity, Minimum Spanning Trees, Determining the Shortest Path.<br>Strategies for Implementing Graphs: Adjacency Lists, Adjacency Matrices.<br><br>**Hashing**: Hashing, Hashing Functions, The Division Method, The Folding Method, The Mid-Square Method, The Radix Transformation Method, The Digit Analysis Method, The Length-Dependent Method. Hashing Functions in the Java Language.<br>Resolving Collisions: Chaining, Open Addressing, Deleting Elements from a Hash Table 419<br>Deleting from a Chained Implementation, Deleting from an Open Addressing, Implementation.<br>Hash Tables in the Java Collections API.<br><br>Recent developments and contemporary issues pertaining to the subject-matter of the course. |
| Teaching Methodology | Face- to- face |
| Bibliography | Lewis, J. & Chase J., JAVA SOFTWARE STRUCTURES: DESIGNING & USING DATA STRUCTURES, Addison Wesley<br><br>Goodrich, M. & Tamassia, R., DATA STRUCTURES AND ALGORITHMS IN C, Wiley<br><br>Sartaj Sahni, DATA STRUCTURES, ALGORITHMS AND APPLICATIONS IN JAVA, McGraw-Hill<br><br>Dale, N., Weems, C and Richards, T. C++ Plus Data Structures, Jones and Bartlett Publishing<br><br>Mark Allen Weiss, DATA STRUCTURES AND ALGORITHM ANALYSIS, Addisson Wesley<br><br>Drozdek, DATA STRUCTURES & ALGORITHMS IN JAVA, Thompson Course Technology<br><br>Clifford, A., PRACTICAL INTRODUCTION TO DATA STRUCTURES AND ALGORITHMS, Prentice Hall |

| | |
|---|---|
| | Bailey, JAVA STRUCTURES: DATA STRUCTURES IN JAVA FOR THE PRINCIPLED ROGRAMMER” , McGraw-Hill |
| Assessment | Coursework 35%<br>Examinations 55%<br>Class Participation and Attendance 10%<br>100% |
| Language | English |