|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| D:\UAAR\UIIT\courseOutlineCommittee\CourseContents_Final_V02\New folder\logo4.png | **PMAS Arid Agriculture University Rawalpindi**  **University Institute of Information Technology** | | | | C:\Users\Shahid\Downloads\IMG-20210824-WA0001.jpg |
| CSC-201 Data Structures | | | | | | |
| **Credit Hours:** | | **4(3-3)** | **Prerequisites:** | **CSC-102** | | |
| **Teacher:** | | Mr. Fahad Burhan Ahmad | **Email** | Fahad.burhan@uaar.edu.pk | | |
| **Course Introduction:** | | | | | | |
| The course is designed to teach students structures and schemes, which allow them to write programmer to efficiently manipulate, store, and retrieve data. Students are exposed to the concepts of time and space complexity of computer programs. | | | | | | |
| **Course content:** | | | | | | |
| Abstract data types, complexity analysis, Big Oh notation, Stacks (linked lists and array implementations), Recursion and analyzing recursive algorithms, divide and conquer algorithms, Sorting algorithms (selection, insertion, merge, quick, bubble, heap, shell, radix, bucket), queue, dequeuer, priority queues (linked and array implementations of queues), linked list & its various types, sorted linked list, searching an unsorted array, binary search for sorted arrays, hashing and indexing, open addressing and chaining, trees and tree traversals, binary search trees, heaps, M-way tress, balanced trees, graphs, breadth-first and depth-first traversal, topological order, shortest path, adjacency matrix and adjacency list implementations, memory management and garbage collection. | | | | | | |
| **Course Objective:** | | | | | | |
| This course aims at teaching the students to write programs that not only are correct but also computation and space efficient and optimized for the intended use through appropriate structuring/organization of the related data. Students will learn the standard data structures such as linked lists, stacks, queues, trees, and graphs and the algorithms that manipulate them. Students will also be introduced to the concept of algorithm complexity analysis in order to make them realize the cost of the operations they perform on their data structures. | | | | | | |
| **Teaching Methodology:** | | | | | | |
| Lectures, Written Assignments, Practical labs, Semester Project, Presentations | | | | | | |
| **Courses Assessment:** | | | | | | |
| Mid Exam, Home Assignments, Quizzes, Project, Presentations, Final Exam | | | | | | |
| **Reference Materials:** | | | | | | |
| * Advanced Algorithms and Data Structures by Marcello La Rocca, 2021. * Data Structures and Algorithms Thinking Made Easy 5th Edition by Narasimha Karumanchi, 2017. * Data Structures and Abstractions with Java by Frank M. Carrano & Timothy M. Henry * Data Structures and Algorithms Thinking with Python by Narasimha Karumanchi, 2016. * Data Structures and Algorithms Made Easy: Data Structures and Algorithmic Puzzles 5th ed. Edition,2016. * Data Structures and Algorithms in C++ by Adam Drozdek , 4th edition, 2012 * Data Structures and Algorithm Analysis in C++ by Mark Allen Weiss, 4th edition, 2013. * C++ plus data structures, 6th edition by Dale 2018. * Data Structures and Algorithms in C++ by Michael T. Goodrich 2nd Edition, 2011. * Introduction to Algorithms, Thomas H. Cormen et al, MIT Press, 3rd edition, 2009. * A Common-Sense Guide to Data Structures and Algorithms, Second Edition: Level Up Your Core Programming Skills 2nd Edition, 2020 | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Course Learning Outcomes (CLOs):** | | | |
| At the end of the course the students will be able to: | **Domain** | **BT Level\*** | **PLOs** |
| 1. Implement various data structures and their algorithms, and **apply** them in implementing simple applications | C | 3 | 3 |
| 1. **Analyze** simple algorithms and determine their complexities | C | 5 | 2 |
| 1. **Apply** the knowledge of data structures to other application domains | C | 3 | 1 |
| 1. **Design** new data structures and algorithms to solve problems | C | 6 | 4 |
| \* BT= Bloom’s Taxonomy, C=Cognitive domain, P=Psychomotor domain, A= Affective domain | | | |

|  |  |  |
| --- | --- | --- |
| **Week/Lecture #** | | **Theory/Practical** |
| Week 1 | Lec-I | **Introduction:**   * Introduction to Data Structures- Need and Significance |
| Lec-II | * Review of the Pre-requisite Knowledge |
| Lec-III | **Abstract Data Types:**   * Arrays, Structures and Classes * Abstraction * Concrete and Abstract Data Types * Class invariants and pre-and post-conditions. |
| Practical-I | * Fundamental OO Design and Implementation Practice |
| Practical-II | * Fundamental OO Design and Implementation Practice |
| Practical-III | * Fundamental OO Design and Implementation Practice |
| Week 2 | Lecture-I | **Complexity Analysis:**   * Complexity Analysis * Time and Space Complexity * **Big Oh notation** |
| Lec-II | **Searching Algorithms:**   * Arrays (basic and Object types) * Algorithms on arrays. * Recursion and analyzing recursive algorithms. |
| Lec-III | * Linear Search * Binary Search |
| Practical-I | Pseudocode and Implementation of linear search. |
| Practical-II | Comparison of searching algorithms |
| Practical-III | * Comparison of searching algorithms |
| Week 3 | Lec-I | **Sorting Algorithms:**   * Selection Sort * Bubble Sort * Insertion Sort |
| Lec-II | **Sorting Algorithms:**   * Divide and conquer algorithms * Merge Sort |
| Lec-III | * Quick Sort |
| Practical-I | * Implementation and Comparison of sorting algorithm |
| Practical-II | * Implementation and Comparison of sorting algorithm |
| Practical-III | * Implementation and Comparison of sorting algorithm |
| Week 4 | Lec-I | **Stack Array Implementation:**   * Stack: Basic/primitive functions * Stack supporting functions |
| Lec-II | **Stack Array Implementation:**   * Stack Applications – Arithmetic Expression, * Infix, post-fix and pre-fix notations. |
| Lec-III | * In-to post transformation and post-fix evaluation using algorithm |
| Practical-I | * Implementation of Static Stack. |
| Practical-II | * Infix to Post fix Conversion |
| Practical-III | * Post fix Evaluation |
| Week 5 | Lec-I | **Stack Array Implementation:**   * Parenthesis validation in mathematical expression using stack. |
| Lec-II | * Multiple brackets validation using stack. |
| Lec-III | * Any other example of validation using stack. |
| Practical-I | Parenthesis matching validation implementation |
| Practical-II | Multiple brackets matching validation implementation |
| Practical-III | Multiple brackets matching validation implementation |
| Week 6 | Lec-I | **Queue Array Implementation:**   * Queue: Basic/primitive functions * Queue supporting functions |
| Lec-II | * Circular queue |
| Lec-III | * Priority queue |
| Practical-I | * Implementation of queue operations |
| Practical-II | * Implementation of circular queue operations |
| Practical-III | * Implementation of priority queue operations, SJF |
| Week 7 | Lec-I | **Queue Array Implementation:**   * Double ended queue * Deiqueue * Deoqueue |
| Lec-II | **Link List:**   * Introduction of Link List * Creating Different types of Link List. |
| Lec-III | * Linked List – Operations and Representations |
| Practical-I | * Queue Applications – OS process and Message queues |
| Practical-II | * Implementation of double ended queue. |
| Practical-III | * Linklist implementation using primitive functions. |
| Week 8 | Lec-I | **Stack Link List Implementation:**   * Dynamic Implementation of Stack * Recursion using Linklist * Linklist operations on Stack |
| Lec-II | * Sorted linked list |
| Lec-III | * Searching an unsorted linklist, |
| Practical-I | * Stack Link list implementation using primitive and supporting functions. |
| Practical-II | * Sorted Link list implementation. |
| Practical-III | * Searching Link list implementation. |
| **Mid Term Exam** | | |
| Week 9 | Lec-I | **Queue Linklist Implementation:**   * Linklist implementation of queue |
| Lec-II | * Linklist based Circular queue |
| Lec-III | * Linklist based Variations: deQueue, priority Queue. |
| Practical-I | Dynamic Queue Implementation |
| Practical-II | Circular queue implementation |
| Practical-III | Priority queue implementation |
| Week  10 | Lec-I | **Queue Linklist Implementation:**   * Linklist base Variations: doubly linklist. |
| Lec-II | **Hashing:**   * Hashing and indexing |
| Lec-III | **Hashing:**   * Open addressing and chaining |
| Practical-I | * Implementation of doubly linklist |
| Practical-II | * Implementation of circular doubly linklist |
| Practical-III | * Implementation of hashing |
| Week 11 | Lec-I | **Trees and Traversals:**   * Introduction and terminology, * Binary trees and types, * Tree traversals * Binary Tree representation, basic operations |
| Lec-II | **Binary Search Tree:**   * Binary Search trees and Representation and operations |
| Lec-III | * M-Way tree |
| Practical-I | Tree implementation |
| Practical-II | Tree implementation |
| Practical-III | BST implementation |
| Week 12 | Lec-I | **Balanced Tree and Heaps**:   * AVL tree * Heap (max and min) * Heaps and Associated Algorithms |
| Lec-II | **Huffman Codes:**   * Huffman Algorithm |
| Lec-III | * Huffman Algorithm (cont..) |
| Practical-I | * Heap implementation |
| Practical-II | * Max and Min Heap implementation |
| Practical-III | * Heap Sort Implementation |
| Week 13 | Lec-I | **Graphs:**   * Terminology, operations and representation * Graph traversals and searching algorithms (BFS-DFS) |
| Lec-II | * Weighted Graphs * Dijkstra’s Algorithms |
| Lec-III | * Kruskal’s Algorithm |
| Practical-I | Graph implementation using Adjacency List |
| Practical-II | Graph implementation using Adjacency Matrix |
| Practical-III | DFS and BFS implementation |
| Week 14 | Lec-I | **Shortest Path:**   * Topological order * Shortest path |
| Lec-II | **Shortest Path:**   * Using Adjacency matrix |
| Lec-III | **Shortest Path:**   * Adjacency list implementations |
| Practical-I | * Implementation of shortest path algorithms. |
| Practical-II | * Implementation of shortest path algorithms. |
| Practical-III | * Implementation of shortest path algorithms. |
| Week 15 | Lec-I | **Shortest Path:**   * Unweighted graph * Weighted graph |
| Lec-II | **Shortest Path:**   * Warshal Floyd’s Algorithm |
| Lec-III | **Memory management:**   * Memory Management * Garbage collection. |
| Practical-I | Implementation of Warshal Floyd’s algorithm |
| Practical-II | * Implementation of Warshal Floyd’s algorithm |
| Practical-III | * Implementation of memory management |
| Week 16 | Lec-I | **Project Demos:**   * Project Demos |
| Lec-II | * Project Demos |
| Lec-III | **Final Course Revision:** |
| Practical-I | Revision of programming exercises |
| Practical-II | Revision of programming exercises |
| Practical-III | Revision of programming exercises |
| **Final Term Exam** | | |