****

**DEDAN KIMATHI UNIVERSITY OF TECHNOLOGY**

**SCHOOL OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY**

**DEPARTMENT OF INFORMATION TECHNOLOGY**

## CIT 2203- DATA STRUCTURES AND ALGORITHMS

**LECTURER : Dr. Jane Kuria EMAIL : kuria.jane@gmail.com**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Course Purpose:**

The course builds on the programming knowledge by use of data structures and algorithms. It takes the student through designing of data structures and incorporates them in algorithms appropriately for problem-solving.

**Learning Outcomes**

At the end of this unit, the students should be able to:

1. Familiarize the student with programming design methods, particularly top-down design.

2. Develop algorithms for manipulating stacks, queues, linked lists, trees, graphs.

3. Develop the data structures for implementing the above algorithms.

4. Develop recursive algorithms as they apply to trees and graphs.

5. Familiarize the student with the issues of Time complexity and examine various algorithms from this perspective.

**Course outline:**

|  |  |  |
| --- | --- | --- |
| **Week 1** | **Introduction to Data Structures**   * Course overview * Importance of data structures * Abstract Data Types (ADT) * Performance analysis (Big O notation) | 5 hrs |
| **Week 2** | **Arrays**   * Static and dynamic arrays * Insertion and deletion * Searching | 5 hrs |
| **Week 3** | **Linked Lists:**   * Singly linked lists * Doubly linked list * Circular linked lists |  |
| **Week 4** | **Stacks:**   * Definition and applications * Implementation using arrays and linked lists * Infix, postfix, and prefix notation * Expression evaluation | 5 hrs |
| **Week 5** | **Queues:**   * Definition and applications * Implementation using arrays and linked lists * Circular queue * Deque and priority queue | 5 hrs |
| **Week 6** | **Hashing and Hash Tables**   * Hash Functions * Collision Resolution Techniques: chaining, open addressing * Applications of Hash. Tables | 5 hrs |
| **Week 7** | **Continuous Assessment Test** |  |
| **Week 8** | **Sorting and Searching Algorithms**   * Sorting. Algorithms: selection, insertion, merge, quick and heaps sort. * Searching. Algorithms: linear, binary. Search * Analysis. And comparison of sorting and searching algorithms. |  |
| **Week 9** | **Non-linear data structures:**  **Graphs:**   * Graph Terminology and representations * Graph traversal Algorithms | 5 hrs |
| **Week 10** | **Continuous assessment test**  **Graphs**   * Shortest Path algorithms * Minimum Spanning Tree Algorithms |  |
| **Week 11** | **Trees**   * Definition and terminology (root, leaf, etc.) * Binary Trees and Binary Search Trees * Tree Traversals: In-order, Pre-order, Post-order | 5 hrs |
| **Week 12** | **Continuous assessment test**  **Balanced Tree**   * Balanced Trees: AVL trees, Red-Black trees * Heaps and Priority Queues | 5 hrs |
| **Week 13** | **Heaps and Priority Queues**   * Binary heaps * Heap operations * Priority queues and their applications | 5 hrs |

**Teaching Methodology:**

Lectures, interactive tutorials, presentations, demonstrations, and lab sessions.

**Instruction Materials/Equipment**

White-board, LCD/Overhead Projector, handouts, smart board and personal computers and e-learning portal.

**Assessment:**

CAT(s) 20 %

Assignment: 5 %

Practical: 5%

Final Exam: 70 %

**Grading system:**

70 – 100 A

60 – 69 B

50 – 59 C

40 – 49 D

Below 40 E

**Main Textbooks**

1. Goodrich, M.T., & Tamassia R. (2010) *Data Structures and Algorithms in Java* (5th edition)
2. Bruno, P, (2000).*Data Structures and Algorithms with Object Oriented Design Patterns in Java*: John Wiley & Sons Inc.
3. Hill, S, .(2000). *Data Structures, Algorithms, and Applications in Java*(2nd Edition). Amason.com

**Reference Textbooks** .

1. Weiss, (1999*). Data Structures and Algorithm Analysis in Java*: Addison Wesley
2. Shaffer, A. (1998) *Practical Introduction to Data Structures and Algorithm Analysis* (Java Edition): Prentice Hall.
3. Weiss, M,(2000) *Data Structures and Problem Solving using C++* (2nd edition): Amazon.com

**Course Outline:**

**Week 1-2: Introduction to Data Structures and Algorithms**

* Overview of Data Structures
* Overview of Algorithms
* Complexity Analysis: Big O, Big Θ, Big Ω notations
* Recursion and its applications

**Week 3-4: Arrays and Linked Lists**

* Arrays: Definition, operations, and applications
* Linked Lists: Singly linked lists, doubly linked lists, circular linked lists
* Operations on linked lists: insertion, deletion, traversal
* Applications of linked lists

**Week 5-6: Stacks and Queues**

* Stacks: Definition, operations, applications (e.g., expression evaluation)
* Queues: Definition, operations, types (simple, circular, priority, dequeues)
* Applications of stacks and queues

**Week 7-8: Trees**

* Definition and terminology (root, leaf, etc.)
* Binary Trees and Binary Search Trees
* Tree Traversals: In-order, Pre-order, Post-order
* Balanced Trees: AVL trees, Red-Black trees
* Heaps and Priority Queues

**Week 9-10: Hashing and Hash Tables**

* Hash Functions
* Collision Resolution Techniques: Chaining, Open Addressing
* Applications of Hash Tables

**Week 11-12: Graphs**

* Graph Terminology and Representations (adjacency matrix, adjacency list)
* Graph Traversal Algorithms: Depth-First Search (DFS), Breadth-First Search (BFS)
* Shortest Path Algorithms: Dijkstra’s, Bellman-Ford
* Minimum Spanning Tree Algorithms: Kruskal’s, Prim’s

**Week 13-14: Sorting and Searching Algorithms**

* Sorting Algorithms: Bubble sort, Selection sort, Insertion sort, Merge sort, Quick sort, Heap sort
* Searching Algorithms: Linear search, Binary search
* Analysis and comparison of sorting and searching algorithms

**Week 15-16: Advanced Topics**

* Divide and Conquer Algorithms
* Dynamic Programming: Concepts, examples (e.g., Fibonacci, Knapsack problem)
* Greedy Algorithms: Concepts, examples (e.g., Huffman coding, Activity selection)

**Week 17-18: Algorithm Design Techniques and Problem Solving**

* Backtracking
* Branch and Bound
* Problem-solving sessions and case studies