GANPAT UNIVERSITY

FACULTY OF DIPLOMA ENGINEERING

Programme	Diploma in Computer Engineering/ Information Technology		
Semester	III	Version	1.0.0.0
Effective from Academic Year	2026-27	Effective for the batch Admitted in	JULY 2025
Course Code	1CEIT3101	Course Name	Data Structure

I. TEACHING-LEARNING AND ASSESSMENT SCHEME

Course Type	Course code	Course Title	Teaching & Learning Scheme									Examination Scheme							
			Credit				Actual Contact Hrs/week			SLH	Total Learning Hrs/Week	TH			PR			SLA	Total
			CL	TL	LL	Total	CL	TL	LL			CE	SEE	Total	CE	SEE	Total		
DSC	1CEIT3101	DATA STRUCTURE	3	-	1	4	3	-	2	2	7	40	60	100	30	20	50	20	170

Abbreviation:			
	CL- Classroom Learning	**TL -** Tutorial Learning	**LL -** Laboratory Learning
	SLH - Self Learning Hours	**SLA -** Self Learning Assessment	**CE -** Continuous Evaluation
	SEE – Semester End Examination		

II. PRE-REQUISITES

Basic knowledge of algorithms, flowcharts, and C programming concepts is desirable.

III. INDUSTRY /EMPLOYER EXPECTED OUTCOMES

- Apply data structures for efficient software development
- Optimize algorithms for better performance
- Manage real-world data using arrays, linked lists, stacks, queues, trees, and graphs
- Use appropriate data structures to solve programming problems
- Contribute to application development projects
- Support database system development
- Assist in building system software applications

IV. COURSE LEARNING OUTCOMES

At the end of the course, students will be able to achieve the following course learning outcomes:

CO1. Implement fundamental data structures (arrays and strings) and analyze algorithmic complexity.
CO2. Apply stack, queue, and recursion techniques to solve computational problems.
CO3. Develop modular programs using functions, recursion, storage classes, and preprocessors.
CO4. Analyze and apply searching and sorting algorithms for data processing.
CO5. Construct and traverse non-linear data structures like trees and graphs for advanced data handling.

V. THEORY LEARNING OUTCOMES AND ALIGNED COURSE CONTENT:

Name of Unit	Theory Learning outcomes(TLO's) aligned to CO's	Learning Content mapped with Theory Learning outcomes (TLO's) & CO's	Marks	Hours
Unit 1 **Introduction to Data**	**TLO 1.1** Describe primitive and non-primitive data structures with examples.	**1.1** Basic Terminology of Primitive & Non-Primitive Data Structures **1.2** Classification of Linear and Non-Linear Data Structures	**10**	**8**

Structures and Algorithms	**TLO 1.2** Classify data structures into linear and non-linear categories. **TLO 1.3** Describe the characteristics, time complexity, and space complexity of algorithms. **TLO 1.4** Perform array operations like insert, delete, traverse, and explain array types. **TLO 1.5** Implement string operations length, copy, compare, concatenate, and find.	**1.3** Introduction to Algorithms, Characteristics, Time and Space Complexity **1.4** Arrays: Concept and Types Array Operations (Insert, Delete, Traverse) **1.5** String Manipulation: String Length, Copy, Compare, Concatenation, Search (Find)		
Unit 2 **Stack and Queue**	**TLO 2.1** Understand and implement stack operations, expression conversions, and recursion using C. **TLO 2.2** Implement various types of queues and perform standard queue operations using C.	**2.1** Stack: Array and Linked Representation, Operations: Push, Pop Applications: Infix to Postfix and Prefix Conversion, Evaluation of Postfix Expression Recursion: Definition, Implementation in C, Examples (Factorial of a Number) **2.2** Queue: Array and Linked Representation, Operations: Enqueue, Dequeue, Types: Circular Queue, Double-Ended Queue (Deque), Priority Queue	**12**	**8**
Unit 3 **Linked Lists**	**TLO 3.1** Understand pointers and structures for dynamic memory allocation. **TLO 3.2** Represent and perform operations (insert, delete, traverse, search) on singly linked lists. **TLO 3.3** Implement doubly, circular, and header linked lists. **TLO 3.4** Identify appropriate use cases for linked lists in real-world scenarios.	**3.1** Introduction to Pointers and Structures **3.2** Singly Linked List: Representation and Memory Allocation, Operations: Insertion, Deletion, Traversing, Searching **3.3** Doubly Linked List, Circular Linked List, and Header Linked List **3.4** Applications of Linked Lists	**12**	**9**
Unit 4 **Searching and Sorting Algorithms**	**TLO 4.1** Apply sequential and binary search techniques to find elements in data. **TLO 4.2** Implement sorting algorithms: bubble, selection, insertion, quick, merge, radix.	**4.1** Searching Techniques: Sequential Search, Binary Search **4.2** Sorting Techniques: Bubble Sort, Selection Sort, Insertion Sort, Quick Sort, Merge Sort, Radix Sort	**12**	**10**

Unit 5 **Non-Linear Data Structures: Trees and Graphs**	**TLO 5.1** Explain tree terminologies, represent binary trees, and perform tree operations/traversals. **TLO 5.2** Represent graphs using adjacency matrix/list and perform BFS/DFS traversals.	**5.1** Trees: Basic Terminology and Types, Binary Trees: Representation (Array & Linked List), Operations: Insertion, Deletion, Traversals (Inorder, Preorder, Postorder) **5.2** Graphs: Basic Terminologies Representation (Adjacency Matrix and List), Graph Traversals: Breadth-First Search (BFS), Depth-First Search (DFS)	**14**	**10**

VI. LABORATORY LEARNING OUTCOME AND ALIGNED PRACTICAL

SR. NO	PRACTICAL/LABORATORY LEARNING OUTCOME (LLO)	PRACTICAL TITLES	RELEVANT COs
1	**LLO 1.1** Apply array operations such as insertion, deletion, and traversal to manage linear data efficiently.	Implement array operations such as insert, delete, and traverse operations on a 1-D array.	CO1
2	**LLO 1.2** Apply array operations such as search to manage linear data efficiently.	Perform search operations on array.	CO1
3	**LLO 1.3** Perform fundamental string operations including length, copy, comparison, concatenation, and searching.	Perform string operations: length, copy, compare, concatenate, and find.	CO1
4	**LLO 2.1** Implement stack data structures using arrays and linked lists to perform push, pop, and peek operations.	Implement stack using array and linked list with push, pop, and peep operations.	CO2
5	**LLO 2.2** Convert infix expressions to postfix form and evaluate postfix expressions using stack techniques.	Convert infix expression to postfix and evaluate postfix expression using stack.	CO2
6	**LLO 2.3** Design and implement queue data structures using arrays and linked lists with enqueue and dequeue operations.	Implement queue using array and linked list with enqueue and dequeue operations.	CO2
7	**LLO 2.4** Develop circular queue and deque using arrays to manage data efficiently with limited memory.	Implement circular queue and deque using array.	CO2
8	**LLO 3.1** Implement singly linked list operations such as insertion, deletion, traversal, and searching.	Implement singly linked list (insert, delete, traverse, search).	CO3
9	**LLO 3.2** Construct and manipulate doubly linked lists and circular linked lists using basic operations.	Implement doubly and circular linked lists with basic operations.	CO3
10	**LLO 4.1** Apply sequential and binary search techniques to locate data in linear and sorted data structures.	Implement sequential and binary search.	CO4
11	**LLO 4.2** Implement and analyze sorting algorithms such as bubble sort, selection sort, and insertion sort.	Implement sorting algorithms: bubble, selection, insertion.	CO4
12	**LLO 4.3** Implement and analyze sorting algorithms such as Quick Sort, Merge Sort, Radix Sort.	Implement Sorting Techniques: Quick Sort, Merge Sort, Radix Sort	CO4

13	**LLO 5.1** Construct binary trees and perform inorder, preorder, and postorder traversals.	Create binary tree and perform inorder, preorder, and postorder traversals.	CO5
14	**LLO 5.2** Implement graph traversals using Breadth First Search (BFS) and with adjacency matrix or list.	Implement BFS traversals on graphs using adjacency matrix/list.	CO5
15	**LLO 5.3** Implement graph traversals using Depth First Search (DFS) with adjacency matrix or list.	Implement DFS traversals on graphs using adjacency matrix/list.	CO5

VII. SUGGESTED MICRO PROJECT/ASSIGNMENTS/ACTIVITIES FOR SELF LEARNING/SKILL DEVELOPMENT (SELF LEARNING)

Micro Projects (Mini Applications / Use Cases)

- Develop a menu-driven program to implement stack operations using array representation
  Skills: Stack operations (push, pop), array manipulation, menu-driven programming
- Develop a program to convert an infix expression into postfix and prefix expressions
  Skills: Stack application, expression conversion algorithms, operator precedence
- Develop a program to evaluate a postfix expression using stack
  Skills: Expression evaluation, stack-based computation, conditional logic
- Develop a menu-driven program to implement queue operations using array representation
  Skills: Queue operations (enqueue, dequeue), array-based implementation
- Develop a program to implement circular queue operations
  Skills: Circular queue logic, modulo arithmetic, efficient memory utilization

Self-Learning / Skill Building Activities

1. Prepare a comparative chart of stack and queue data structures.
   Goal: Understand structural differences, operations, and use cases.
2. Trace step-by-step execution of infix to postfix conversion using stack.
   Goal: Strengthen algorithmic thinking and stack application skills.
3. Solve multiple postfix expression evaluation problems manually.
   Goal: Improve logical reasoning and expression evaluation accuracy.
4. Prepare a table comparing array and linked list implementations of stack and queue.
   Goal: Analyze memory usage, performance, and implementation trade-offs.
5. Write pseudocode for stack and queue operations before coding in C.
   Goal: Develop structured problem-solving and algorithm design skills.
6. Identify real-world applications of stacks and queues in operating systems and compilers.
   Goal: Relate data structures to practical system-level applications.
7. Implement recursion-based programs for factorial and Fibonacci series.
   Goal: Understand recursion flow, base cases, and function call stack behavior.
8. Prepare a flowchart for circular queue operations.
   Goal: Visualize control flow and improve conceptual clarity.
9. Use an online compiler to practice stack and queue programs with different inputs.
   Goal: Enhance coding proficiency and debugging skills.
10. Analyze time and space complexity of stack and queue operations.
    Goal: Build awareness of algorithm efficiency and performance evaluation.

VIII. LIST OF INSTRUMENTS / EQUIPMENT / TRAINER BOARD

1	C compiler and IDE (GCC, Code::Blocks, Turbo C++)
2	**Python Tutor:** • https://pythontutor.com/

3	**Data Structure Visualizations (USF):** • https://www.cs.usfca.edu/~galles/visualization/

IX. LIST OF REFERENCE BOOKS			
Sr. No	**Title**	**Author**	**Publication**
1	Data Structures Using C	Reema Thareja	Oxford University Press
2	Data Structures Using C	ISRD Group	McGraw Hill, New Delhi
3	Data Structures	Chitra, A Rajan, P T	Tata McGraw Hill, New Delhi
4	Introduction to Data Structures with Application	Paul Trembly, G-Sorenson	McGraw Hill Education

X. LINK OF LEARNING WEB RESOURCE	
1	Geeks for Geeks - Data Structures Tutorial: https://www.geeksforgeeks.org/data-structures/
2	Tutorial Points: https://www.tutorialspoint.com/data_structures_algorithms/index.htm
3	Free tutorials: https://www.w3schools.com/dsa/index.php
4	Learning platform: https://www.coursera.org/specializations/data-structures-algorithms

XI. SUGGESTED WEIGHTAGE TO LEARNING EFFORTS & ASSESSMENT PURPOSE							
Unit	**Unit Title**	**Aligned COs**	**Learning Hours**	**R-Level**	**U-Level**	**A-Level**	**Total Marks**
1	Introduction to Data Structures and Algorithms	CO1	8	2	3	3	8
2	Stack and queue	CO2	8	2	3	3	8
3	Linked Lists	CO3	9	2	3	4	9
4	Searching and Sorting Algorithms	CO4	10	2	3	5	10
5	Non-Linear Data Structures: Trees and Graphs	CO5	10	2	3	5	10
	Grand Total		**45**	**16**	**20**	**24**	**60**

XIII. COs AND POs AND PSOs MAPPING

Course Outcome (Cos)	**Programme Outcomes(POs)**											**Programme Specific Outcomes(PSOs)**		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2	PSO3
CO1	3	3	2	2	0	0	2	1	1	0	1	3	3	2
CO2	3	3	3	2	0	1	2	1	1	1	1	3	3	2
CO3	2	3	3	3	0	1	2	1	1	1	2	3	3	3
CO4	3	3	3	3	0	1	2	1	1	1	2	3	3	3
CO5	3	3	3	3	0	1	2	1	1	1	2	3	3	3

Legends:- *3-High; 2-Moderate/Medium; 1-Slight/Low; 0-None*