LESSON PLAN 2025-26(WINTER)

NAME OF THE TEACHER : DEEPAK KUMAR BARDA, LECT.(STAGE-II,CSE)

Subject: **DATA STRUCTURES(Course Code: CSEPC 205/TH3)**
Program: Diploma in Computer Science and Engineering
Semester: 3rd
Total Contact Hours: 45
Total Marks: 100
Assessment: Internal Assessment – 30, End Term – 70

After completion of the course, the students will be able to:
CO1-Explain fundamental data structure concepts, classifications, and algorithm analysis.
CO2-Apply linear data structures such as stacks, queues, and their variations.
CO3-Develop linked list structures, including singly, doubly, and circular linked lists.
CO4-Implement non-linear data structures like trees and perform operations such as insertion, deletion, and traversal.
CO5-Describe graph representations and traversal techniques for efficient data organization.

Day	Unit	Topic/Sub-Topic	Learning Objective	Activities	Homework	COURSE OBJECTIVE
UNIT I: Introduction to Data Structures (8 Hours)						
1	I	Introduction to Data Structures: Basic Terminology	Define core concepts like data, data structure, and algorithm.	Interactive discussion on real-world examples of data structures (e.g., a phone book, a playlist).	Research and write down the definitions of data, data structure, and algorithm in your own words.	CO1
2	I	Classification of Data Structures	Differentiate between linear and non-linear data structures.	Use a mind map to visually categorize data structures. Discuss where each type is commonly used.	Find and list two examples of linear and two examples of non-linear data structures.	CO1
3	I	Operations on Data Structure	Identify the fundamental operations performed on data structures (e.g., traversal, insertion, deletion).	Group discussion on which operations are possible on different data structures.	Write a short paragraph explaining the difference between searching and sorting.	CO1
4	I	Asymptotic Analysis of Algorithms	Understand the concept of time and space complexity.	Introduce Big-O notation with simple code examples like linear search vs. binary search.	Analyze the time complexity of a simple loop that prints numbers from 1 to N.	CO1
5	I	Worst-Case Analysis of Algorithms	Analyze the worst-case scenario for an algorithm's performance.	Walk through an example of worst-case analysis for an algorithm like bubble sort.	Practice analyzing the worst-case time complexity of an algorithm of your choice.	CO1

6	I	Best-Case and Average-Case Analysis	Distinguish between best-case, worst-case, and average-case analysis.	Compare the three cases for a given algorithm with different input scenarios.	Given an algorithm, determine its best-case and worst-case scenarios.	CO1
7	I	Review and Mini-Quiz	Consolidate knowledge of Unit I.	Solve practice problems on Big-O notation and basic terminology.	N/A	CO1
8	I	Unit I Assessment	Demonstrate mastery of Unit I.	Quiz on terminology, classification, and algorithm analysis.	N/A	CO1
UNIT II: Linear Data Structures (11 Hours)						
9	II	Stacks: Introduction & Array Representation	Explain the LIFO principle and represent a stack using an array.	Use a physical stack of books to demonstrate LIFO. Draw the array representation on the board.	Write down the steps to implement push and pop operations on an array-based stack.	CO2
10	II	Stacks: Operations on a Stack	Implement push, pop, peek, and isEmpty operations.	Live coding session to build a simple stack class in a programming language.	Implement the stack operations on a provided code template.	CO2
11	II	Applications of Stacks: Infix to Postfix Transformation (1/2)	Convert an infix expression to a postfix expression using a stack.	Work through a step-by-step example on the board, demonstrating the stack's role.	Convert a simple infix expression (e.g., A + B * C) to postfix.	CO2
12	II	Applications of Stacks: Infix to Postfix Transformation (2/2)	Continue practicing complex infix-to-postfix conversions.	Pair programming exercise to solve a more complex expression.	Convert a more complex infix expression with parentheses and multiple operators.	CO2
13	II	Applications of Stacks: Evaluating Postfix Expressions	Evaluate a postfix expression using a stack.	Walkthrough the evaluation process with a sample postfix expression.	Evaluate a given postfix expression and show the state of the stack at each step.	CO2
14	II	Queues: Introduction & Array Representation	Explain the FIFO principle and represent a queue using an array.	Use a queue of students at a counter to explain FIFO. Draw array representation and discuss front and rear pointers.	Write down the steps to implement enqueue and dequeue operations.	CO2
15	II	Queues: Operations on a Queue	Implement enqueue, dequeue, peek, and isEmpty operations.	Live coding session to build a simple queue class.	Implement the queue operations on a provided code template.	CO2
16	II	Types of Queues: Dequeue, Circular Queue	Understand and implement variations of queues.	Draw and explain the concepts of Dequeue and Circular Queue. Discuss their advantages.	Write a brief explanation of why a Circular Queue is more efficient than a linear queue.	CO2

17	II	Applications of Queues: Round Robin Algorithm	Understand how queues are used in CPU scheduling.	Discuss the concept of time slicing and the role of the queue in the Round Robin algorithm.	Explain another real-world application of a queue (e.g., print spooling).	CO2
18	II	Review and Mini-Quiz	Consolidate knowledge of stacks and queues.	Solve practice problems and answer conceptual questions.	N/A	CO2
19	II	Unit II Assessment	Demonstrate mastery of stacks and queues.	Practical coding test and a short quiz.	N/A	CO2
UNIT III: Linked Lists (10 Hours)						
20	III	Linked Lists: Singly Linked List	Understand the structure and concept of a singly linked list.	Draw nodes and pointers to visualize the list. Discuss the advantages over arrays.	Draw a singly linked list with 5 nodes. Label the head and a null pointer.	CO3
21	III	Linked Lists: Representation in Memory & Operations (1/3)	Represent a singly linked list in memory.	Code the basic node structure and a class for the linked list.	Write a function to insert a new node at the beginning of a singly linked list.	CO3
22	III	Linked Lists: Operations (2/3)	Implement insertion and deletion at the end of the list.	Pair programming to implement the functions.	Implement a function to search for a specific node in a singly linked list.	CO3
23	III	Linked Lists: Operations (3/3)	Implement insertion and deletion at a specific position.	Guided coding session to handle edge cases for insertion and deletion.	Implement a function to delete a node at a given position.	CO3
24	III	Circular Linked Lists	Understand the structure and implementation of a circular linked list.	Draw a circular linked list and discuss the operations specific to it.	Convert a singly linked list to a circular linked list in a code example.	CO3
25	III	Doubly Linked Lists	Understand the structure and advantages of a doubly linked list.	Draw and code the Node structure with a prev pointer.	Write a function to insert a new node at the end of a doubly linked list.	CO3
26	III	Linked List Representation of Stacks	Implement stack operations using a linked list.	Compare array-based and linked list-based stack implementations. Discuss the pros and cons.	Write a short paragraph on the benefits of a linked list stack over an array stack.	CO3
27	III	Linked List Representation of Queues	Implement queue operations using a linked list.	Live coding session for a linked list queue.	Write a function to find the length of a linked list queue.	CO3
28	III	Review and Mini-Quiz	Consolidate knowledge of linked lists.	Solve practice problems on all types of linked lists and their applications.	N/A	CO3

29	III	Unit III Assessment	Demonstrate mastery of linked lists.	Practical coding test.	N/A	CO3
UNIT IV: Non-Linear Data Structures (16 Hours)						
30	IV	Trees: Basic Terminologies	Define tree-related terms (e.g., root, node, parent, child, leaf).	Draw a tree structure and label all the parts. Use a family tree analogy.	Draw a tree structure and label the depth, height, and level of each node.	CO4
31	IV	Binary Trees: Definition and Concepts	Understand the properties of a binary tree.	Discuss the rules for a binary tree (max two children per node).	Given a list of numbers, draw the resulting binary search tree.	CO4
32	IV	Binary Tree Representations: Arrays and Linked Lists	Represent a binary tree in memory using arrays and linked lists.	Draw the two representations on the board. Discuss the pros and cons of each.	Explain when a linked list representation of a tree is better than an array representation.	CO4
33	IV	Operations on a Binary Tree: Insertion	Implement the insertion operation in a binary search tree.	Live coding session for a basic binary tree.	Implement the insertion logic on a provided tree class.	CO4
34	IV	Operations on a Binary Tree: Deletion	Implement the deletion operation for various cases.	Walk through the three cases for deletion (no children, one child, two children).	Given a binary search tree, perform a deletion and draw the resulting tree.	CO4
35	IV	Traversals: In-order, Pre-order, Post-order (1/2)	Understand and implement the three main traversal methods.	Trace the path of each traversal method on a sample tree.	Given a tree, write down the pre-order, in-order, and post-order traversals.	CO4
36	IV	Traversals: In-order, Pre-order, Post-order (2/2)	Practice traversal techniques on different binary tree structures.	Work on more complex examples, including trees with an unbalanced structure.	Given a pre-order and in-order traversal, reconstruct the original tree.	CO4
37	IV	Types of Binary Trees	Identify different types of binary trees (e.g., full, complete, perfect).	Discuss the properties of each type. Use visual examples to differentiate them.	Categorize a few provided tree diagrams by type.	CO4
38	IV	Graphs: Graph Terminologies	Define graph-related terms (e.g., vertex, edge, degree).	Use a social media network or a road map as an analogy to explain graph concepts.	Draw a simple graph and label the vertices and edges.	CO5
39	IV	Representation of Graphs: Set, Linked, Matrix	Represent a graph in memory using adjacency lists and matrices.	Draw and explain both representations, and discuss the trade-offs in terms of space and time complexity.	Given a graph, write its adjacency matrix and adjacency list representations.	CO5

40	IV	Graph Traversals: Breadth-First Search (BFS)	Implement and understand the BFS algorithm.	Trace the BFS traversal on a sample graph. Discuss its use in finding the shortest path.	Given a graph and a starting node, show the order of visited nodes using BFS.	CO5
41	IV	Graph Traversals: Depth-First Search (DFS)	Implement and understand the DFS algorithm.	Trace the DFS traversal on a sample graph. Discuss its use in topological sorting.	Given a graph and a starting node, show the order of visited nodes using DFS.	CO5
42	IV	Review and Mini-Quiz	Consolidate knowledge of trees and graphs.	Solve conceptual and practical problems related to trees and graphs.	N/A	CO5
43	IV	Unit IV Assessment	Demonstrate mastery of trees and graphs.	Practical coding test.	N/A	CO5
44	General	Course Review & Problem Solving	Review all key topics from the entire course.	Work through a comprehensive problem that requires integrating multiple data structures (e.g., using a queue and a graph to find a path).	N/A	CO5
45	General	Final Assessment	Final practical exam or comprehensive test.	N/A	N/A	CO5

Deepak kumar Barik
11.07.025
Signature of Teacher

Deepak kumar Barik
11.07.025
Signature of HOD