L	T	P	**C**
2	2	2	4

DATA STRUCTURES

PREREQUISITE KNOWLEDGE: Programming in C

COURSE DESCRIPTION & OBJECTIVES

This course is aimed at offering fundamentals concepts of data structures and explains how to implement them. It begins with the basic concepts of data, data structures and then introduces the primitive and non-primitive data structures in detail. It forms the basis for understanding various ways of representing data and its usage in different computing applications.

MODULE 1

UNIT-1: Data Structures Basics [L:5 T: 6 P:6 = 17 Hours]

Basic Terminology – data, information, datatype; Data Structures – Introduction, storage structures- sequential and linked storage representations; classification of data structures; Applications of data structures.
Sorting: Selection Sort, Bubble Sort, Insertion Sort, Quick Sort and Merge Sort.
Searching: Linear Search and Binary Search

UNIT-2: Linked Lists and Stacks, Queues [L:11 T: 10 P:10 = 31 Hours]

Linked List: Introduction, Types of linked list – Singly linked list, doubly linked list and circular linked list, representation of linked list, Operations of linked list: Traverse forward/ reverse order, searching, insertion and deletion; Applications of linked lists.
Stack – Introduction, array and linked representations, implementation and their applications;
Queue – Introduction, array and linked representations, implementation; Types – Linear, circular and doubly ended queues – operations; Applications of Queues.

Practice Problems on Recursion – Level 1

- Find the product of 2 numbers using recursion
- Find the sum of natural numbers using recursion
- Find the factorial of a number using recursion
- Find the Nth term of Fibonacci series using recursion
- Calculate the power using recursion
- Write a recursive program for checking if a given number is a prime number
- Given two integers write a function to sum the numbers without using any arithmetic operators
- Convert a decimal to binary using recursion
- Print all factors using recursion
- Find the maximum product of digits among numbers less than or equal to N

Practice Problems Recursion – Level 2

- Implement insertion sort recursively
- Write a program to find the numbers less than N that are product of exactly 2 distinct prime numbers - using recursion
- Implement selection sort recursively
- Find the middle of a singly linked list using recursion
- Find the sum of even numbers of an array using recursion
- Check if a given array is in sorted order using recursion
- Print alternate nodes of a linked list using recursion

- Reverse a doubly linked list using recursion
- Write a recursive function that returns all permutations of a given list
- Implement bubble sort recursively

Practice Problems on Sorting and Searching – Level 1

- Implement the insertion sort function
- Implement the bubble sort function
- Implement the quick sort function
- Implement the merge sort function
- Implement the selection sort function
- Implement linear search function
- Implement binary search function

Practice Problems on Stacks – Level 1

- Implement two stacks using a single array
- Given an array replace every element with nearest greater element on the right
- Given a stack reverse the elements using only push and pop functions
- Postfix evaluation using stack
- Balance symbols
- Find middle element in a stack
- Remove middle element from a stack
- Implement push and pop using linked list
- Given an array of characters with the middle marked by X, check if the string is a palindrome
- Maximum sum in sliding window

Practice Problems on Queues – Level 1

- Write a program to accept two numbers as input check if they are equal
- Write a program to accept two characters as input and check if they are equal
- Write a program to accept two numbers as input and print the greater of the 2 numbers
- Write a program to accept two numbers as input and print the lesser of the 2 numbers
- Write a program to accept 3 numbers as input and print the maximum of the 3
- Write a program to accept 3 numbers as input and print the minimum of the 3
- Write a program to accept a number as input and print EVEN if it is an even number and ODD if it is an odd number
- Write a program to accept a number as input and check if it is divisible by 3. If it is divisible by 3 print YES else print NO
- Write a program to accept a number as input and check if it is divisible by both 3 & 5. If it is divisible print YES else print NO
- Write a program to accept a number as input and check if it is positive, negative or zero.

Practice Problems on SLL – Level 1

- Implement the insert function to insert nodes into a singly linked list (ascending order)
- Implement the insert function to insert nodes into a singly linked list (descending order)
- Implement the search node function
- Implement the delete node function
- Display forwards function
- Display backwards function
- Count the number of nodes in a singly linked list
- Swap alternate nodes of a singly linked list

- Move last node to the front of the linked list
- Move first node to the last of the linked list

Practice Problems on DLL – Level 1

- Implement insert function
- Implement display forward function
- Implement display backward function
- Implement search function
- Implement delete function
- Reverse a doubly linked list from M to N
- Find the sum of the odd and even nodes
- Count odd keys of the linked list
- Merge two sorted lists
- Delete adjacent duplicate nodes

Practice Problems on CLL – Level 1

- Insert function (circular doubly linked list)
- Search function
- Display forward
- Display backward
- Delete node (circular doubly linked list)
- Print the middle N nodes of a circular singly linked list
- Move the last node of a circular singly linked list to the beginning
- Delete adjacent duplicate nodes of a circular singly linked list
- Delete nodes greater than a value from a circular doubly linked list
- Find the sum of the nodes of a circular linked list

Practice Problems on Linked List – Level 2

- Given 2 sorted linked lists, print the common elements
- Reverse a list (using Stack)
- Given a pointer to a node (not the last node), delete the node
- Reverse a list (Recursive)
- Reverse a list (Iterative)
- Reverse a singly linked list in pairs (recursive)
- Reverse a singly linked list in pairs (iterative)
- Check if a singly linked list is a palindrome or not
- Remove the loop if exists
- Given 2 linked lists with data in the ascending order, merge them into a single list

MODULE 2

UNIT-1: Trees **[L:8 T: 8 P:8 = 24 Hours]**

Basic Terminology, Types of Trees, Binary Tree – Introduction, properties, array and linked representations; Tree traversals and their implementation; Expression trees; BST – definition and operations, AVL trees – definition and construction; Applications of binary trees.

UNIT-2: Graphs **[L:8 T: 8 P:8 = 24 Hours]**

Basic Terminology, Types of Graphs, Graphs representations – adjacency matric, adjacency list; Traversals - breath first search and depth first search; Applications of graphs.

Hashing: Introduction, Different hash functions, collision: avoidance and handling methods.

Practice Problems on BST – Level 1

- Insert function
- Insert function (recursive)
- Search function
- Pre order traversal
- Post order traversal
- In order traversal
- Level order traversal
- Delete child node
- Delete parent node
- Delete nodes greater than a value from a circular doubly linked list

Practice Problems on Priority Queues – Level 1

- Meeting rooms problem
- Ugly number
- Find median from data stream
- Find the top K frequent elements
- Find K Pairs with smallest sums
- Find the Kth smallest element in a sorted matrix
- Trapping Rain Water
- Rearrange String k distance apart
- Sort characters by frequency
- Solve the maze problem

Practice Problems on Graphs – Level 1

- Implement Graph data structure
- Implement BFS - iterative solution
- Implement BFS - recursive solution
- Implement DFS - iterative solution
- Implement DFS - recursive solution
- Check if given graph is strongly connected or not
- Check if given graph is strongly connected or not - using DFS
- Given a graph find the arrival and departure time of its vertices in DFS. Arrival time is the time when the vertex was explored for the first time, and departure time is the time at which all the neighbours are explored and are ready to backtrack
- Given a directed acyclic graph and a source vertex, find the cost of the shortest path from source vertex to all other vertices present in the graph. If a vertex cannot be reached from given source vertex that distance may be printed as infinite
- Given an undirected graph, check if the graph is 2 edge connected or not

Practice Problems on Hashing – Level 1

- Print a binary tree in vertical order
- Find whether an array is subset of another array
- Given an array A [] and a number x, check for pair in A [] with sum as x
- Minimum operation to make all elements equal in array
- Maximum distance between two occurrences of same element in array
- Check if a given array contains duplicate elements within k distance from each other
- Find duplicates in a given array when elements are not limited to a range

- Most frequent element in an array
- Smallest subarray with all occurrences of a most frequent element
- First element occurring k times in an array

Practice Problems on Graphs – Level 2

- Find the shortest graph distances between every pair vertices in a given path. Assume that the graph does not have any negative edges
- Find the shortest graph distances between every pair of vertices in a given path. The graph can have negative edges
- Detect cycle in DFS
- Count the number of connected components of a graph represented in the adjacent matrix
- Count the number of connected components of a graph represented in the adjacent matrix - using DFS
- Find a spanning tree - not necessarily a minimum spanning tree
- Detect cycle in an undirected graph
- Given an undirected graph, find its depth
- Determine if a directed graph has a unique topological ordering
- Given a directed acyclic graph and two vertices v and w, find the lowest common ancestor

SKILLS:

- Experienced to Store data and various types of data to handle
- Ordering and sorting of data
- Indexing and Searching of required data from large data sequences
- Exposed to various characteristics such as Linear or non-linear, Homogeneous or heterogeneous and Static and Dynamic

COURSE OUTCOMES:

Upon completion of the course, the student will be able to achieve the following outcomes:

No.	**Course Outcome**	**Blooms Level**	**PO**
1	Explore the organization of several ADTs and the manipulation (searching, insertion, deletion, traversing) of data stored in various data structures.	Apply	1
2	Apply different data structures to solve a given problem.	Apply	1
3	Analyze the efficiency of using different data structures and choose the efficient data structure for solving a given problem.	Analyze	2
4	Develop new algorithms to solve various problems.	Apply and Create	3,4

TEXT BOOKS:

1. D Samantha, “Classic Data Structures”, 2nd Edition, Eastern Economic Prentice hall private limited press, 2000.

REFERENCE BOOKS:

1. Ellis Horowitz and Sartaj Sahni, “Fundamentals of Data Structures”, illustrated edition, Computer Science Press, 2006.
2. Mark Allen Weiss, “Algorithms, Data Structures and Problem Solving with C++ illustrated”, 2nd edition, Addison-Wesley publishing company, 2002.

3. R G Dromey and Pearson, “How to solve it by Computer”, 2nd edition, Impression edition, 1998.