Text Books:

1. Seymour Lipschutz, **Data Structures**, Schaum's Outline Series, TMH, 4th Edition, 2019.
2. Adam Drozdek, **Data Structures and Algorithms in C++**, Cengage Learning, 3rd Edition, 2012.
3. SartajSahni, **Data Structures, Algorithms and Applications in C++**, Universities Press, 2nd Edition, 2011.

Reference Books:

1. D.S Malik, **Data Structure using C++**, Cengage Learning, Second Edition, 2010.
2. Aaron M. Tenenbaum, Moshe J. Augenstein, YedidyahLangsam, **Data Structures Using C and C++**, PHI, 2nd Edition, 2009.
3. Robert L. Kruse, **Data Structures and Program Design in C++**, Pearson, 3rd Edition, 1999.

Semester	**: II**
Course Type	**: DSC**
Course Code	**: CSCDSC152**
Name of the Course	**: Lab on Data Structure**
Learning level	**: Foundation or Introductory Course**
Credits	**: 3**
Contact Hours	**: 90**
Total Marks	**: 100**
End Semester Marks	**: 70**
Internal Marks	**: 30**

Course Objectives:

1. *Help students apply the theoretical concepts of data structures in a practical setting. It should provide exercises and programming assignments that require students to implement and manipulate different data structures.*
2. *Enhancing students' programming skills by providing practical programming exercises.*
3. *It should encourage students to write code, debug, and test their implementations of data structures and associated algorithms.*

*This paper provides practical knowledge of data structure. List of laboratory programming assignments (not limited to these)***:**

1. Write a program to search an element from a list. Give users the option to perform Linear or Binary search. Use Template functions.
2. WAP using templates to sort a list of elements. Give users the option to perform sorting using Insertion sort, Bubble sort or Selection sort.
3. Implement Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list and concatenate two linked lists (include a function and also overload operator +).
4. Implement Doubly Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.

5. Implement Circular Linked List using templates. Include functions for insertion, deletion and search of a number, reverse the list.
6. Perform Stack operations using Linked List implementation.
7. Perform Stack operations using Array implementation. Use Templates.
8. Perform Queues operations using Circular Array implementation. Use Templates.
9. Create and perform different operations on Double-ended Queues using Linked List implementation.
10. WAP to scan a polynomial using a linked list and add two polynomials.
11. WAP to calculate factorial and to compute the factors of a given no. (i)using recursion, (ii) using iteration
12. WAP to display fibonacci series (i)using recursion, (ii) using iteration
13. WAP to calculate GCD of 2 number (i) with recursion (ii) without recursion
14. WAP to create a Binary Search Tree and include following operations in tree: (a) Insertion (Recursive and Iterative Implementation) (b) Deletion by copying (c) Deletion by Merging (d) Search a no. in BST (e) Display its preorder, postorder and inorder traversals Recursively (f) Display its preorder, postorder and inorder traversals Iteratively (g) Display its level-by-level traversals (h) Count the non-leaf nodes and leaf nodes (i) Display height of tree (j) Create a mirror image of tree (k) Check whether two BSTs are equal or not
15. WAP to convert the Sparse Matrix into non-zero form and vice-versa.
16. WAP to reverse the order of the elements in the stack using additional stack.
17. WAP to reverse the order of the elements in the stack using additional Queue.
18. WAP to implement Diagonal Matrix using a one-dimensional array.
19. WAP to implement Lower Triangular Matrix using a one-dimensional array.
20. WAP to implement the Upper Triangular Matrix using a one-dimensional array.
21. WAP to implement Symmetric Matrix using a one-dimensional array.
22. WAP to create a Threaded Binary Tree as per inorder traversal, and implement operations like finding the successor / predecessor of an element, insert an element, inorder traversal.
23. WAP to implement various operations on AVL Tree.

Course outcomes: *After successful completion of the course, the students will be able to*

1. *Students should be able to demonstrate a solid understanding of various data structures such as arrays, linked lists, stacks, queues, trees, graphs, and hash tables. They should be able to explain the characteristics, operations, and applications of these data structures.*
2. *Students should be able to implement data structures and associated algorithms using a programming language.*
3. *Students should be able to analyze the time and space complexity of algorithms associated with different data structures.*