COURSE CONTENT

ADVANCED DATA STRUCTURES LAB								
I Semester: CSE								
Course Code	**Category**	**Hours / Week**			**Credits**	**Maximum Marks**		
		L	**T**	**P**	**C**	**CIA**	**SEE**	**Total**
2415831	**Foundation**	**0**	**0**	**4**	**2**	**40**	**60**	**100**
Contact Classes: Nil	**Tutorial Classes: Nil**	**Practical Classes: 60**				**Total Classes: 60**		
Prerequisites: A course on Computer Programming & Data Structures.								

Course Overview:

This is a practical, hands-on laboratory course designed to reinforce and extend the theoretical concepts of advanced data structures through implementation and experimentation. The course focuses on understanding complex data structures, analyzing their performance, and applying them to solve real-world problems efficiently.

Course Objectives:

1. To understand and implement advanced data structures such as trees, heaps, hash tables, tries, and graphs for efficient data organization and processing.
2. To develop programming skills for performing operations like insertion, deletion, searching, sorting, and traversal using advanced data structures.
3. To analyze and compare the performance of different data structures and algorithms in terms of time and space complexity.
4. To apply advanced data structures and pattern matching algorithms in solving real-world computational problems efficiently.
5. To enhance problem-solving and logical thinking abilities through practical implementation of searching, sorting, hashing, and tree-based algorithms.

Course Outcomes: After Completion of the Course, Students should be able to

1. Make use of advanced tree data structures including Binary Search Trees (BST), AVL Trees, Red-Black Trees, and B-Trees, highlighting their practical applications in efficient data storage, searching, and balancing.
2. Explore sorting algorithms based on divide-and-conquer and heap-based approaches to enhance efficiency in data processing and real-time system applications.
3. Simulate the functioning of advanced heap structures including Min-Max Heap, Leftist Tree, and Binomial Heap.
4. Apply hashing techniques to realize dictionary operations for faster data retrieval.
5. Compare various string pattern matching algorithms to determine their suitability for text processing applications.

List of Programs.

1. Write a program to perform the following operations:
   a) Insert an element into a binary search tree.
   b) Delete an element from a binary search tree.
   c) Search for a key element in a binary search tree.

2. Write a program for implementing the following sorting methods:
   a) Merge sort b) Heap sort c) Quick sort

3. Write a program to perform the following operations:
   a) Insert an element into a B- tree.
   b) Delete an element from a B- tree.
   c) Search for a key element in a B- tree.

4. Write a program to perform the following operations:
   a) Insert an element into a Min-Max heap
   b) Delete an element from a Min-Max heap
   c) Search for a key element in a Min-Max heap

5. Write a program to perform the following operations:
   a) Insert an element into a Leftist tree
   b) Delete an element from a Leftist tree
   c) Search for a key element in a Leftist tree
6. Write a program to perform the following operations:
   a) Insert an element into a binomial heap
   b) Delete an element from a binomial heap.
   c) Search for a key element in a binomial heap

7. Write a program to perform the following operations:
   a) Insert an element into a AVL tree.
   b) Delete an element from a AVL search tree.
   c) Search for a key element in a AVL search tree.

8. Write a program to perform the following operations:
   a) Insert an element into a Red-Black tree.
   b) Delete an element from a Red-Black tree.
   c) Search for a key element in a Red-Black tree.
9. Write a program to implement all the functions of a dictionary using hashing.
10. Write a program for implementing Knuth-Morris-Pratt pattern matching algorithm.
11. Write a program for implementing Brute Force pattern matching algorithm.
12. Write a program for implementing Boyer pattern matching algorithm.

TEXT BOOKS:

1. Fundamentals of Data structures in C, E. Horowitz, S. Sahni and Susan Anderson Freed, 2nd Edition, Universities Press
2. Data Structures Using C – A.S. Tanenbaum, Y. Langsam, and M.J. Augenstein, PHI/Pearson education.
3. Introduction to Data Structures in C, Ashok Kamthane, 1st Edition, Pearson.

REFERENCE BOOKS:

1. The C Programming Language, B.W. Kernighan, Dennis M. Ritchie, PHI/Pearson Education
2. C Programming with problem solving, J.A. Jones & K. Harrow, Dreamtech Press
3. Data structures: A Pseudocode Approach with C, R.F. Gilberg and B.A. Forouzan, 2nd Edition, Cengage Learning.

ELECTRONIC RESOURCES:

1. https://www.coursera.org/specializations/data-mining/
2. https://www.classcentral.com/course/mining-stanford-university-mining-massive-dataset-2406/
3. https://www.mygreatlearning.com/academy/learn-for-free/courses/data-mining1/
4. https://www.mygreatlearning.com/academy/learn-for-free/courses/mastering-big-data- analytics/

MATERIALS ONLINE:

1. Course template
2. Open-ended experiments
3. Definitions and terminology
4. Lab Manual