Course: BTech **Semester:** 4

Prerequisite: proficiency in a programming language (e.g., C++, Python) and a strong grasp of data structures and algorithms, with a focus on problem-solving skills and efficient code implementation. Familiarity with common coding platforms (e.g., Codeforces, LeetCode) is also beneficial.

Rationale: Competitive coding sharpens problem-solving skills, enhances algorithmic thinking, and fosters quick and efficient coding practices. It provides a platform for continuous learning, challenges individuals to tackle diverse problems, and fosters a competitive spirit that's valuable in technical interviews and real-world software development.

Teaching and Examination Scheme

Teaching Scheme					Examination Scheme					Total
Lecture Hrs/Week	Tutorial Hrs/Week	Lab Hrs/Week	Hrs/Week	Credit	Internal Marks			External Marks		
					T	CE	P	T	P	
-	-	4	-	2	-	-	20	-	30	50

SEE - Semester End Examination, **CIA** - Continuous Internal Assessment (It consists of Assignments/Seminars/Presentations/MCQ Tests, etc.)

Course Outcome

After Learning the Course the students shall be able to:
After Learning the Course the students shall be able to: 1. Develop strong problem-solving skills, improve algorithmic thinking, and enhance proficiency in coding by tackling a variety of challenging problems. 2. Cultivate the ability to write efficient and optimized code under time constraints, honing the skill of quickly translating algorithmic insights into practical solutions. 3. Gain a competitive advantage in technical interviews and coding assessments, showcasing the ability to tackle diverse coding challenges commonly encountered in job placements and coding competitions. 4. Foster a mindset of continuous learning by regularly engaging with new problems, staying updated on emerging algorithms, and adapting to evolving coding paradigms.

List of Practical

1.	Write a program for implementing a MINSTACK which should support operations like push, pop, overflow, underflow, display 1. Construct a stack of N-capacity 2. Push elements 3. Pop elements 4. Top element 5. Retrieve the min element from the stack
2.	Write a program to deal with real-world situations where Stack data structure is widely used Evaluation of expression: Stacks are used to evaluate expressions, especially in languages that use postfix or prefix notation. Operators and operands are pushed onto the stack, and operations are performed based on the LIFO principle.
3.	Write a program for finding NGE NEXT GREATER ELEMENT from an array.
4.	Write a program to design a circular queue(k) which Should implement the below functions a. Enqueue b. Dequeue

	c. Front d. Rear
5.	Write a Program for an infix expression, and convert it to postfix notation. Use a queue to implement the Shunting Yard Algorithm for expression conversion.
6.	Write a Program for finding the Product of the three largest Distinct Elements. Use a Priority Queue to efficiently find and remove the largest elements.
7.	Write a Program to Merge two linked lists(sorted).
8.	Write a Program to find the Merge point of two linked lists(sorted).
9.	Write a Program to Swap Nodes pairwise.
10.	Write a Program for Building a Function ISVALID to VALIDATE BST.
11.	Write a Program to Build BST.
12.	Write a Program to determine the depth of a given Tree by Implementing MAXDEPTH.
13.	Write a Program to Understand and implement Tree traversals i.e. Pre-Order Post-Order, In-Order.
14.	Write a Program to perform Boundary Traversal on BST.
15.	Write a program for Lowest Common Ancestors.
16.	Write a Program to verify and validate mirrored trees or not.
17.	Write a Program for a basic hash function in a programming language of your choice. Demonstrate its usage to store and retrieve key-value pairs.
18.	Implement a hash table using separate chaining for collision handling. Perform operations like insertion, deletion, and search on the hash table.
19.	

	Write a Program to Implement Two sums using HASHMAP.
20.	Write a Program to Implement Search, insert, and Remove in Trie.
21.	Write a Program to Implement Huffman coding.
22.	Write a Program to find Distinct substrings in a string.
23.	Write a Program to find The No of Words in a Trie.
24.	Write a Program to view a tree from left View.
25.	Write a Program to Traverse a Tree using Level Order Traversal.