CODEBYTE

Mentorship Program

CURRICULUM

CODEBYTE

CURRICULUM

1. Data Structure & Algorithm
2. Database Management System
3. Operating System (OS)
4. Object Oriented Programming
5. Computer Network
6. Projects
7. Interview Preparation

CODEBYTE

Data Structure & Algorithm

Course Overview: Data Structures & Algorithms (DSA) is a fundamental aspect of computer science essential for writing efficient code. Mastering DSA is crucial for aspiring software developers, as it forms the basis of problem-solving skills required in the industry. Most companies include DSA in their technical interview rounds, making it a key component in the placement process.

Topics Covered:

- **Data Structures & Its Types:** Introduction to various data structures, their uses, and differences.
- **Time & Space Complexity:** Analysis of algorithms based on their execution time and memory usage.
- **Basic Math for DSA:** Essential mathematical concepts and techniques for solving DSA problems**.**
- **Arrays & LinkedList:** Fundamental linear data structures for storing collections of elements.
- **Strings:** Handling and manipulating sequences of characters.

- **Stacks & Queues:** Abstract data types for managing ordered collections with specific access rules**.**
- **Sorting & Searching Algorithms**: Techniques for organizing and retrieving data efficiently.
- **Heaps:** Specialized tree-based structures that facilitate priority queue operations.
- **Map & Set:** Data structures for storing unique elements and key-value pairs.
- **Recursion:** Technique where a function calls itself to solve smaller instances of the problem.
- **Trees:** Hierarchical data structures used for representing nested relationships.
- **Backtracking:** Algorithmic technique for solving problems incrementally, abandoning solutions that fail criteria.
- **Dynamic Programming:** Method for solving complex problems by breaking them down into simpler subproblems and storing results.
- **Graphs:** Structures for representing and traversing networks of connected nodes.

CODEBYTE

Object Oriented Programming

Course Overview: Object-Oriented Programming (OOP) is essential for creating organized, reusable code. Mastering OOP is crucial for developers, forming the foundation of modern software practices. Understanding concepts like encapsulation, inheritance, and polymorphism is key for robust applications and job placements. This course covers core OOP principles, design patterns, and practical implementations for a successful software development career.

Topics Covered:

- **Class and Objects:** Blueprints for creating objects with shared attributes and behaviors in OOP.
- **Feature/Characteristics of OOPs:** Includes encapsulation, inheritance, polymorphism, and abstraction for organized and reusable code.
- **Variable Scopes:** Determines the accessibility of variables within different parts of a program.
- **Static (Variables, Functions, Objects):** Elements that persist across instances, shared by all objects of a class.
- **Encapsulation:** Bundles data and methods into a single unit, hiding internal details and exposing a public interface.

- **Polymorphism:** Allows objects to be treated as instances of their parent class, enabling flexibility in method implementation.
- **Inheritance (Type and Mode):** Enables classes to inherit properties and behaviors from parent classes, supporting code reusability.
- **Virtual (Functions and Class):** Functions or classes marked as virtual can be overridden by subclasses, facilitating dynamic binding.
- **Friend Function and Friend Class:** Allows external functions or classes to access private and protected members of a class.
- **Call by Value, Reference:** Methods of passing arguments to functions, either by copying values or passing references for efficiency.
- **This Pointer:** Refers to the current instance of a class, enabling access to its own members within member functions.
- **Exception Handling:** Mechanism for handling errors and exceptional situations gracefully during program execution.
- **Constructor and Destructor:** Special methods for initializing and cleaning up objects, respectively, in OOP languages.
- **Reference Variable:** Variable that holds the memory address of another variable, facilitating efficient memory usage.
- **Abstraction:** Hides complex implementation details, providing a simplified interface for interacting with objects.

CODEBYTE

Database Management System

Course Overview: Database Management Systems (DBMS) are a core component of computer science, crucial for the effective storage, retrieval, and management of data. Understanding DBMS is essential for aspiring software developers as it underpins the ability to build and maintain robust applications. Proficiency in DBMS is often assessed in technical interviews, making it a critical skill for the placement process.

Topics Covered:

- **DBMS Introduction | Set 1:** Overview of the basics and significance of Database Management Systems.
- **DBMS Introduction | Set 2 (3-Tier Architecture):** Explanation of the 3-tier architecture model in DBMS, detailing its layers and functions.
- **DBMS Architecture 2-level 3-level:** Comparison and discussion of 2-level and 3-level DBMS architectures.
- **Need For DBMS:** Reasons why DBMS is essential for efficient data management and organizational operations.
- **Data Abstraction and Data Independence:** Concepts that allow for the simplification of database design and protection from changes in data structure.

- **Database Objects:** Description of various objects in a database, such as tables, views, indexes, and sequences.
- **Disadvantages of DBMS:** Examination of potential drawbacks and limitations of using a DBMS.
- **ER Model Enhanced:** Overview of the Enhanced Entity-Relationship model, which includes additional concepts for more complex database designs.
- **ER Model:** Introduction to the Entity-Relationship model used for database design to visually represent data relationships.
- **Minimization of ER Diagram:** Techniques for simplifying ER diagrams while preserving their essential information.
- **ER Model: Generalization, Specialization and Aggregation**: Explanation of advanced ER concepts for handling hierarchical and complex relationships.
- **Relational Model and CODD Rules:** Overview of the relational database model and the foundational rules defined by E.F. Codd.
- **Relational Model:** Discussion of the relational model structure, where data is organized into tables (relations).
- **Keys in Relational Model (Candidate, Super, Primary, Alternate and Foreign):** Description of different types of keys used to uniquely identify and relate data in a relational model.
- **Number of possible Superkeys:** Explanation of the concept and calculation of all possible superkeys in a relational schema.
- **Anomalies in Relational Model:** Overview of potential issues (anomalies) in relational databases, such as redundancy and update anomalies.
- **Mapping from ER Model to Relational Model:** Process of converting an ER diagram into a relational schema.

- **Database Objects:** Description of various objects in a database, such as tables, views, indexes, and sequences.
- **Strategies for Schema Design:** Techniques for creating efficient and scalable database schemas.
- **Schema Integration:** Methods for merging different database schemas into a unified schema.
- **Introduction to Relational Algebra:** Basics of relational algebra, the theoretical foundation of relational databases.
- **Basic Operators:** Fundamental relational algebra operations like selection, projection, union, and difference.
- **Extended Operators:** Advanced relational algebra operations such as join, intersection, and division.
- **Inner Join vs Outer Join:** Comparison of inner join, which returns matching rows, and outer join, which includes non-matching rows as well.
- **Join Operation vs Nested Query:** Differences between using join operations and nested queries to retrieve related data.
- **DBMS | Tuple Relational Calculus:** Introduction to a non-procedural query language based on specifying properties of desired results.
- **Introduction to Normal Forms:** Overview of database normalization and the different normal forms.
- **Minimum Relations Satisfying 1NF:** Criteria for ensuring a database relation meets the First Normal Form requirements.
- **The Problem of Redundancy in Database:** Issues caused by redundant data in databases, such as inconsistency and inefficiency.

- **How to Find the Highest Normal Form of a Relation:** Steps to determine the highest normal form a database relation satisfies.
- **Domain-Key Normal Form:** Explanation of a normal form that ensures minimal redundancy based on domain and key constraints.
- **Introduction of 4th and 5th Normal Form:** Overview of Fourth and Fifth Normal Forms, addressing multi-valued dependencies and join dependencies.
- **Denormalization in Databases:** Concept and practices of denormalization to improve database performance at the cost of redundancy.
- **DBMS | Data Replication:** Techniques and benefits of replicating data across multiple database systems for reliability and accessibility.
- **Introduction to Transactions and Concurrency Control:** Basics of database transactions and methods to ensure consistency and isolation in concurrent environments.
- **ACID Properties:** Fundamental principles ensuring reliable transactions in databases: Atomicity, Consistency, Isolation, Durability.
- **Concurrency Control - Introduction:** Overview of techniques to manage simultaneous database operations without conflicts.
- **Implementation of Locking in DBMS:** Methods for enforcing locks to manage access to database resources.
- **Concurrency Control Protocols – Lock Based Protocol:** Protocols using locks to ensure proper transaction sequencing and isolation.
- **Introduction to TimeStamp and Deadlock Prevention Schemes:** Methods using timestamps to order transactions and prevent deadlocks.
- **Dirty Read in SQL:** Concept where a transaction reads data that has not yet been committed, potentially causing inconsistencies.

- **HTypes of Schedules:** Different ways of ordering transactions to ensure correctness and consistency in databases.
- **Transaction Isolation Levels in DBMS:** Various levels of transaction isolation to control visibility of changes between transactions.
- **Database Recovery Techniques:** Methods to restore a database to a consistent state after a failure.
- **Starvation in DBMS:** A situation where a transaction is perpetually delayed due to resource contention.
- **Starvation in DBMS:** A situation where a transaction is perpetually delayed due to resource contention.
- **DBMS | OLAP vs OLTP:** Comparison of Online Analytical Processing (OLAP) for complex queries and Online Transaction Processing (OLTP) for routine transactions.
- **Types of OLAP Systems:** Various OLAP systems including MOLAP, ROLAP, and HOLAP, tailored for different analytical needs.
- **Indexing and its Types:** Overview of database indexing techniques to improve query performance.
- **Bitmap Indexing:** A type of indexing that uses bitmaps for efficient query processing in large databases.
- **Inverted Index:** Index structure commonly used in search engines to map content to its locations in a database.
- **SQL Queries on Clustered and Non-Clustered Indexes:** Differences and usage of SQL queries on clustered and non-clustered indexes for performance optimization.
- **SQL | Tutorials:** Educational materials and tutorials to learn and master SQL.
- **Quiz on SQL:** Assessments to test and reinforce understanding of SQL concepts and queries.

CODEBYTE

Operating System

Course Overview: This course delves into the core principles of Operating Systems (OS), covering process management, memory allocation, file systems, and system security. It equips students with the essential knowledge to understand and develop efficient OS functionalities, laying a strong foundation for careers in software development and system administration

Topics Covered:

- **Introduction of Operating System:** Fundamentals of managing hardware and software resources within a computing environment.
- **Types of Operating Systems:** Classification based on usage, such as single-user, multi-user, real-time, and embedded systems.
- **Functions of Operating System:** Management of resources like memory, CPU, and peripherals, alongside providing an interface for user interaction.
- **Real-time systems:** Operating systems designed to process data in real-time, meeting strict timing constraints for critical applications.
- **Tasks in Real-Time Systems:** Handling time-sensitive processes with precise timing requirements, ensuring timely execution.

- **Difference between multitasking, multithreading, and multiprocessing:** Various approaches to concurrent execution, distinguished by their handling of multiple tasks or threads.
- **Types of computer memory (RAM and ROM):** Storage mediums for temporary data access (RAM) and permanent data retention (ROM).
- **Difference between 32-bit and 64-bit operating systems:** Differing memory addressing capabilities, affecting system performance and software compatibility.
- **What happens when we turn on a computer?:** Initialization process involving power-on self-test (POST), bootloader execution, and loading the operating system.
- **Boot Block:** Initial segment of a storage device containing bootloader code for system startup.
- **Microkernel:** Modular approach to kernel design, delegating essential functions while keeping core services minimal.
- **Kernel I/O Subsystem (I/O System):** Component responsible for managing input and output operations between software and hardware.
- **Monolithic Kernel and key differences from Microkernel:** Integrated kernel design with all essential services bundled, contrasting with modular microkernel architecture.
- **Introduction of System Call:** Mechanism for user-level programs to request services from the operating system kernel
- **Privileged and Non-Privileged Instructions: Instructions categorized based on their access to system resources, with privileged instructions reserved for kernel-level operations.**
- **Process | (Introduction and different states):** Overview of processes in an operating system and their various states such as new, ready, running, waiting, and terminated.
- **States of a process:** Detailed explanation of each state a process can be in during its lifecycle.
- **Process Table and Process Control Block (PCB):** Data structures used by the operating system to manage processes and store their information.

- **Process Scheduler:** Component responsible for selecting processes from the ready queue for execution on the CPU.
- **CPU Scheduling:** Techniques for efficiently allocating CPU resources to processes.
- **Preemptive and Non-Preemptive Scheduling:** Different approaches to CPU scheduling where processes can either be interrupted or allowed to run until completion**.**
- **Measure the time spent in context switch?:** Evaluating the time taken to save and restore the state of processes during context switches.
- **Difference between dispatcher and scheduler:** Distinction between components responsible for context switching and process selection.
- **FCFS Scheduling | Set 1 and Set 2:** First-Come-First-Serve scheduling algorithm and its variations.
- **Convoy Effect in Operating Systems:** Phenomenon where short processes get stuck behind long processes in a scheduling queue, causing inefficient resource utilization
- **Belady's Anomaly:** Occurrence where increasing the number of available frames in memory for page replacement algorithms leads to more page faults.
- **Shortest Job First (SJF) scheduling | Set 1 (Non-preemptive):** Scheduling algorithm that selects the shortest job next for execution.
- **Program for Shortest Job First (SJF) scheduling | Set 2 (Preemptive):** Implementation of preemptive SJF scheduling in code.
- **Shortest Job First scheduling with predicted burst time:** Enhancing SJF scheduling by predicting the next CPU burst time for processes.
- **Longest Remaining Time First (LRTF) Program and algorithm:** Variant of SJF where the process with the longest remaining time is selected for execution.
- **Round Robin scheduling:** Time-sharing scheduling algorithm where each process is allocated a fixed time slice.

- **Selfish Round Robin Scheduling:** Variant of Round Robin scheduling where processes attempt to maximize their CPU allocation.
- **Round Robin Scheduling with different arrival times:** Implementing Round Robin scheduling with variations in process arrival times.
- **Priority Scheduling:** Scheduling algorithm where processes with higher priorities are given preference for CPU allocation.
- **Program for Preemptive Priority CPU Scheduling:** Implementation of preemptive priority scheduling in code.
- **Priority Scheduling with different arrival time – Set 2:** Priority scheduling implementation considering variations in process arrival times.
- **Starvation and Aging in Operating Systems:** Issues related to processes not getting sufficient CPU time and aging mechanisms to mitigate starvation.
- **Highest Response Ratio Next (HRRN) Scheduling:** Scheduling algorithm that selects the process with the highest response ratio for execution.
- **Multilevel Queue Scheduling:** Scheduling algorithm that categorizes processes into multiple queues based on their priority.
- **Multilevel Feedback Queue Scheduling:** Extension of multilevel queue scheduling with the ability to move processes between queues based on their behavior.
- **Process Synchronization | Introduction and Set 2:** Techniques to coordinate the execution of multiple processes to avoid race conditions and ensure data consistency.
- **Critical Section:** Code segment where shared resources are accessed and must be executed atomically to prevent race conditions.
- **Inter Process Communication:** Mechanisms for processes to exchange data and synchronize their actions.
- **Interprocess Communication:** Methods: Different techniques for achieving interprocess communication, such as shared memory, message queues, and signals.

- **IPC through shared memory:** Sharing data between processes using a common memory area.
- **IPC using Message Queues:** Communication between processes via message passing using queues.
- **Message-based Communication in IPC (inter-process communication):** Exchanging messages between processes to facilitate communication.
- **Communication between two processes using signals in C:** Interprocess communication in C programming using signals.
- **Semaphores in operating system:** Synchronization primitives used to control access to shared resources and prevent race conditions.
- **Mutex vs. Semaphore:** Synchronization primitives used to control access to shared resources, with mutex allowing only one thread at a time and semaphore permitting a specified number of concurrent accesses.
- **Process Synchronization | Monitors:** High-level synchronization construct allowing mutual exclusion and condition synchronization within concurrent processes.
- **Dekker's algorithm:** Algorithm for mutual exclusion among multiple processes without requiring hardware support.
- **Producer-Consumer Problem using Semaphores | Set 1:** Synchronization problem involving producers producing data and consumers consuming it using semaphores..
- **Dining Philosopher Problem Using Semaphores:** Classical synchronization problem involving multiple philosophers sharing a limited number of resources (chopsticks).
- **Dining-Philosophers Solution Using Monitors:** Solution to the dining philosopher problem using monitor-based synchronization.
- **Readers-Writers Problem | Set 1 (Introduction and Readers Preference Solution):** Synchronization problem involving multiple readers and writers accessing a shared resource, with preference given to readers.
- **Reader-Writers solution using Monitors:** Solution to the readers-writers problem using monitor-based synchronization.

- **Sleeping Barber problem:** Synchronization problem modeling a barber shop where a barber serves customers who arrive and wait if the barber is busy.
- **Lock variable synchronization mechanism:** Technique using lock variables to implement mutual exclusion and ensure thread safety.
- **Interprocess Communication: Methods:** Various mechanisms for communication between processes, including shared memory, message passing, and semaphores.
- **Deadlock Introduction:** Situation in which two or more processes are unable to proceed because each is waiting for the other to release a resource.
- **Deadlock Detection And Recovery:** Techniques for identifying deadlocks and recovering from them by terminating or preempting processes.
- **Deadlock, Starvation, and Livelock:** Concepts related to resource contention, where deadlock involves processes waiting indefinitely, starvation occurs when a process is perpetually denied resources, and livelock involves processes constantly changing their states without making progress.
- **Deadlock Prevention And Avoidance:** Strategies for preventing deadlocks by ensuring that at least one of the necessary conditions (mutual exclusion, hold and wait, no preemption, circular wait) does not hold.
- **Banker's Algorithm:** Resource allocation algorithm ensuring that processes do not enter unsafe states by checking the safety of allocating resources.
- **Resource Allocation Graph (RAG):** Graphical representation of resource allocation and request relationships among processes and resources.
- **Methods of resource allocation to processes by operating system:** Techniques for distributing resources among processes, including banker's algorithm, resource allocation graphs, and priority-based allocation.
- **Program for Banker's Algorithm:** Implementation of the banker's algorithm in code to prevent deadlock.
- **Banker's Algorithm: Print all the safe state (or safe sequences):** Algorithm to identify and print all safe sequences of resource allocations.
- **Deadlock detection algorithm:** Algorithm for identifying the presence of deadlocks in a system.

- **Program for Deadlock free condition in Operating System:** Implementation to ensure that a system remains deadlock-free.
- **Operating System | Thread:** Lightweight processes sharing the same memory space and resources, allowing for concurrent execution.
- **Threads and its types:** Different types of threads including user-level and kernel-level threads.
- **Operating System | User Level thread Vs Kernel Level thread:** Distinction between threads managed by the user-level thread library and those managed by the kernel.
- **Process-based and Thread-based Multitasking:** Comparison of multitasking approaches based on processes and threads.
- **Multi-threading models:** Various models for implementing multithreading, such as many-to-one, one-to-one, and many-to-many.
- **Benefits of Multithreading:** Advantages of multithreading including improved responsiveness, resource utilization, and parallelism.
- **Zombie Processes and their Prevention:** Processes that have completed execution but still have an entry in the process table, and measures to prevent them.
- **Maximum number of Zombie process a system can handle:** Maximum number of zombie processes a system can accommodate before encountering issues.
- **Operating System | Remote Procedure call (RPC):** Mechanism for interprocess communication where a process can execute code in another address space.
- **Memory Hierarchy Design and its Characteristics:** Organization of memory into levels with different access speeds and capacities.
- **Introduction to memory and memory units:** Overview of memory types and units used in computer systems.
- **Different Types of RAM (Random Access Memory):** Various types of RAM including SRAM, DRAM, and DDR RAM.

- **Buddy System:** Memory allocation technique dividing memory into blocks of fixed sizes to satisfy allocation requests.
- **Memory Management | Partition Allocation Method:** Techniques for dividing memory into sections for efficient process allocation.
- **Fixed (or Static) Partitioning in Operating System:** Dividing memory into fixed-size partitions, each assigned to a single process.
- **Variable (or Dynamic) Partitioning in Operating System:** Dividing memory into dynamically sized partitions to fit processes more efficiently.
- **Non-Contiguous Allocation in Operating System:** Allocating memory in separate blocks that are not physically contiguous.
- **Logical vs Physical Address in Operating System:** Logical addresses are generated by the CPU, while physical addresses refer to actual locations in memory.
- **Paging:** A memory management scheme that eliminates the need for contiguous allocation by dividing memory into fixed-size pages.
- **Requirements of Memory Management System:** Ensuring efficient allocation, protection, sharing, and organization of memory.
- **Memory Management – Mapping Virtual Address to Physical Addresses:** Translating virtual addresses to physical addresses using a page table.
- **Operating System | Remote Procedure call (RPC):** Mechanism for interprocess communication where a process can execute code in another address space.
- **Page Table Entries:** Components in a page table that store mapping information between virtual and physical addresses.
- **Virtual Memory:** A technique that extends the available memory by using disk space as additional RAM.
- **Memory Interleaving:** A method to increase memory access speed by spreading data across multiple memory modules.

- **Virtual Memory Questions:** Common queries and explanations related to virtual memory concepts and operations.
- **Operating System Based Virtualization:** Creating multiple virtual environments on a single physical machine using OS-level virtualization.
- **Inverted Page Table:** A space-efficient page table where each entry corresponds to a physical page frame.
- **Swap Space:** Disk space used to extend physical memory and store inactive memory pages.
- **Page Fault Handling:** The process of retrieving data from disk to memory when a requested page is not in RAM.
- **Fixed (or Static) Partitioning in Operating System: Same as above:** Dividing memory into fixed-size partitions, each assigned to a single process.
- **Segmentation:** Dividing memory into variable-sized segments based on the logical divisions of a program.
- **Program for Next Fit Algorithm in Memory Management:** An algorithm that searches for the next available block of memory to allocate to a process.
- **Overlays in Memory Management:** A technique where only essential parts of a program are loaded into memory to save space.
- **Page Replacement Algorithms:** Methods to decide which memory pages to swap out when new pages are needed.
- **Program for Page Replacement Algorithms | Set 1 (LRU):** Implementation of the Least Recently Used page replacement algorithm.
- **Program for Optimal Page Replacement Algorithm:** An algorithm that replaces the page that will not be used for the longest period of time.
- **LFU (Least Frequently Used) Cache Implementation:** An algorithm that removes the least frequently accessed items from the cache first.

- **Second Chance (or Clock) Page Replacement Policy:** A page replacement algorithm that gives pages a second chance before replacing them.
- **Techniques to Handle Thrashing:** Methods to prevent excessive paging, which degrades system performance.
- **Allocating Kernel Memory (Buddy System and Slab System):** Techniques for managing memory allocation within the kernel, using the buddy system and slab allocator.
- **Program for Buddy Memory Allocation Scheme in Operating Systems | Set 1:** Implementation of the buddy system for dynamic memory allocation.
- **File Systems:** Organized methods for storing and retrieving files on a storage device.
- **Unix File System:** A file system used by Unix and Unix-like operating systems, known for its simplicity and efficiency.
- **Implementing Directory Management Using Shell Script:** Writing shell scripts to create, manage, and navigate directories.
- **File Directory | Path Name:** Describing the hierarchy of directories and the path used to locate files.
- **Structures of Directory:** Different ways to organize files within directories, such as single-level, two-level, and hierarchical structures.
- **File Allocation Methods:** Techniques for storing files on disk, including contiguous, linked, and indexed allocation.
- **File Access Methods:** Various methods for accessing data within a file, such as sequential and direct access.
- **Secondary Memory:** Non-volatile storage used for long-term data storage, like hard drives and SSDs.
- **Secondary Memory – Hard Disk Drive:** A magnetic storage device used for storing large amounts of data permanently.

- **Disk Scheduling Algorithms:** Algorithms to determine the order in which disk I/O requests are processed to improve performance.
- **Program for SSTF Disk Scheduling Algorithm:** Implementation of the Shortest Seek Time First algorithm for disk scheduling.
- **What Exactly Spooling is All About?:** Simultaneous peripheral operations online: managing the queue of print jobs or data processing tasks.
- **Difference Between Spooling and Buffering:** Spooling involves queuing tasks for execution, while buffering temporarily stores data in transit.
- **Free Space Management:** Techniques for managing unused space on a storage device, including bitmaps and free lists.

CODEBYTE

Computer Network

Course Overview: This course covers the fundamental principles of Computer Networks, including architectures, protocols, and security. Students will learn about data communication, network topologies, IP addressing, and network management. This foundation prepares students for careers in network administration, cybersecurity, and IT infrastructure management.

Topics Covered:

- **Introduction to Computer Networks:** Overview of network fundamentals and their significance.
- **Basics of Computer Networking:** Core concepts and terminologies in networking.
- **The Internet and the Web:** Understanding the structure and functionality of the Internet and the World Wide Web.
- **Internet and Web Programming:** Programming techniques for web applications and internet services.
- **The New Internet (Internet of Everything):** Exploring the interconnected nature of modern devices.

- **Unknown Facts of Networking:** Lesser-known but important aspects of networking.
- **Network Goals:** Objectives and desired outcomes of networking systems.
- **Line Configuration:** Various configurations for connecting devices in a network.
- **Transmission Modes:** Different methods of data transmission (simplex, half-duplex, full-duplex).
- **Types of Transmission Media:** Physical media used for transmitting data (cables, fiber optics, wireless).
- **Unicast, Broadcast, Multicast:** Methods of data delivery in a network.
- **Network Topologies:** The layout patterns of network connections (star, ring, mesh, etc.).
- **Types of Networks (LAN, MAN, WAN):** Classification of networks based on their scope and scale.
- **Access Networks:** Networks that connect subscribers to their service providers.
- **OSI Model:** A conceptual framework used to understand network interactions in seven layers.
- **TCP/IP vs OSI Model:** Comparison between the TCP/IP and OSI networking models.
- **Flow Control Protocols:** Mechanisms to control data flow between sender and receiver.
- **Active Directory Domain Service:** A directory service for managing network resources in Windows environments.

- **Advantages and Disadvantages:** Pros and cons of different networking technologies and protocols.
- **Data Link Layer:** Layer responsible for node-to-node data transfer and error detection.
- **Local Area Network (LAN) Technologies:** Technologies used to create and manage LANs.
- **Bridges, Internetworking:** Devices and techniques for connecting multiple networks.
- **Framing, MAC Address, Filtering:** Methods for data packet encapsulation and addressing.
- **Multiple Access Protocols:** Protocols that determine how multiple nodes share a communication medium.
- **Byte Stuffing, Bit Stuffing:** Techniques to avoid confusion with control information in data streams.
- **Circuit Switching vs Packet Switching:** Comparison of two data transmission methods.
- **CSMA/CD, CSMA, Collision Avoidance:** Protocols to manage data collisions on a network.
- **Switching Techniques, VLAN:** Methods to manage network traffic and create virtual networks.
- **Sliding Window Protocol, ARQ:** Techniques for reliable data transmission and error handling.
- **Manchester Encoding, Error Detection:** Methods for encoding data and detecting errors.
- **Hamming Code, Wake-on-LAN:** Error-correction code and technology to remotely wake computers.

- **Basics of Wi-Fi, IEEE 802.11:** Wireless networking standards and their basics.
- **Token Ring, Multiplexing:** Network protocols and techniques to combine multiple signals.
- **Network Layer:** Layer responsible for data routing, forwarding, and addressing.
- **ISDN, IPv4, IPv6:** Integrated Services Digital Network and Internet Protocol versions.
- **IP and Classful Addressing, Subnetting, Supernetting:** Techniques for managing IP addresses.
- **Classless Addressing:** Addressing method without the traditional class constraints.
- **Fragmentation, ICMP:** Breaking data into smaller packets and the Internet Control Message Protocol.
- **Routing Protocols:** Protocols used to determine the best path for data transmission.
- **RIP, OSPF, EIGRP, BGP:** Common routing protocols for network data flow management.
- **VLAN, NAT, VRRP, HSRP:** Network techniques for segmentation, translation, and redundancy.
- **Traceroute, ARP, DHCP:** Tools and protocols for network troubleshooting and configuration.
- **DNS, SNMP, MIME:** Protocols for domain name resolution, network management, and email.
- **Transport Layer:** Layer responsible for end-to-end communication and data transfer.

- **TCP, UDP, Congestion Control:** Core transport protocols and techniques to manage network congestion.
- **Leaky Bucket Algorithm, Error Control:** Techniques for managing data flow and ensuring error-free transmission.
- **Sliding Window Protocol:** Protocol to manage data frames between sender and receiver.
- **Application Layer:** Layer providing network services directly to end-users and applications.
- **Protocols (SMTP, DNS, FTP, HTTP):** Common application layer protocols for email, domain resolution, file transfer, and web browsing.
- **P2P File Sharing, Quality of Service:** Methods for direct file sharing and ensuring network service quality.
- **Web Caching, Security:** Techniques for optimizing web content delivery and securing network data.
- **Firewalls, Cryptography, VPNs:** Technologies for network security, data encryption, and virtual private networks.
- **Compression Techniques:** Methods for reducing data size for transmission.
- **LZW, Arithmetic Coding, Shannon-Fano:** Specific data compression algorithms.
- **Network Experiments:** Practical activities for understanding network operations.
- **Socket Programming, Checksum:** Programming for network communication and data integrity checks.
- **Troubleshooting Commands:** Commands used to diagnose and fix network issues.

- **Devices:** Overview of various networking devices and their functions.
- **Networking Devices, Routers, Switches:** Devices used for managing and directing network traffic.
- **Collision Domain, Broadcast Domain:** Network areas affected by data collisions and broadcasts.
- **Spanning Tree Protocol:** Protocol to prevent loops in network topologies.
- **Cloud Computing, VoIP, NFC:** Technologies for remote computing, voice over IP, and near-field communication.
- **USB, Type-C Port:** Universal Serial Bus standards for connecting devices.
- **ARP & DHCP:** Protocols for address resolution and dynamic host configuration.
- **Virtual Memory:** Technique for extending physical memory using disk space.

CODEBYTE

Revision

Objective : To give an overall review of all topics to improve understanding and prepare better for tests or interviews.

Structure :

1. Quick Recap of Core Subjects:

- Summarize key concepts and terminologies from Data Structures & Algorithms, DBMS, OOPs, OS, and CN.
- Include quick-reference guides or cheat sheets for each subject.

2. Topic-Wise Practice Sessions:

- Solve topic-specific questions, e.g., recursion problems, DBMS normalization, or scheduling algorithms.
- Focus on weak areas identified during learning.

3. Quick Recap of Core Subjects:

- Encourage peer learning by forming groups to discuss and solve problems collaboratively.

4. Quiz :

- Conduct quizzes on fundamental concepts to assess knowledge retention.

CODEBYTE

Mock Interviews

Objective : Simulate real-world interview experiences to build confidence and identify improvement areas.

Structure :

1. Technical Mock Interviews:

- Focus on coding problems (DSA), system design, and subject-specific technical questions (DBMS, OOPs, etc.).
- Conduct sessions with experienced mentors or industry professionals.

2. HR Mock Interviews:

- Practice answering common behavioral and situational questions.
- Focus on communication skills, confidence, and storytelling for past experiences.

3. Coding Rounds:

- Organize live coding challenges using platforms like HackerRank or LeetCode.
- Include problems with varying difficulty to simulate actual company coding tests.

4. Feedback and Analysis:

- Provide detailed feedback after every mock interview.
- Share insights on strengths, areas of improvement, and strategies to enhance performance.

5. Panel Discussions:

- Conduct mock interviews with a panel of interviewers to replicate real-world scenarios.

6. Resume Review & Elevator Pitch:

- Refine resumes with mentor feedback.
- Practice delivering a concise and impactful personal introduction.

CODEBYTE

Projects

Course Overview: This course emphasizes the pivotal role projects play in students' educational and professional journeys. Students will delve into various project types, honing their practical skills and bolstering their academic learning. By mastering project execution and presentation, students will be well-equipped to demonstrate their capabilities to prospective employers, fostering confidence and readiness for future career endeavors.

CODEBYTE

Interview Preparation

Course Overview: This course prepares students for job interviews by covering essential interview techniques. Students will learn about common interview questions, behavioral and technical interview strategies, and how to effectively present their skills and experiences. This foundation equips students with the confidence and knowledge needed to succeed in job interviews and secure their desired positions.

Topics Covered:

- **DSA Questions to Crack Product Based Companies:** Key data structures and algorithms questions commonly asked in product-based company interviews.
- **DBMS Interview Questions:** Essential database management system questions for technical interviews.
- **OOPS Interview Questions:** Important object-oriented programming questions frequently encountered in interviews.
- **OS Interview Questions:** Critical operating system questions for interview preparation.
- **CN Interview Questions:** Core computer networking questions asked in technical interviews.

CODEBYTE

- **Pro Tips for Interviews & Coding Rounds:** Expert advice on excelling in technical interviews and coding assessments.

- **Resume Tips & How to Clear ATS:** Guidance on crafting an effective resume and passing applicant tracking systems.

CODEBYTE

FOR FURTHER INFORMATION CONTACT US AT

foundersoffice@thecodebyte.in

growth@thecodebyte.in