Renaissance Plus Detailed Curriculum

1. Begin With Programming

C++

- Parts of a Computer Program, Header Files, Libraries etc.
- Compilers, executables and IDE.
- Different types of Data types.
- Integer Overflows, Type-Casting, Precision Issues and ASCII codes.
- Reading and Writing to the console.
- Arithmetic, Assignment, Logical and Relational Operators.
- Expression evaluation.
- Swapping the variables : with and without extra variable.
- Simple-If, If-else, Nested If-else and Else-If ladder.
- Finding maximum and second maximum
- Loops: While loop, For loop, Nested loops.
- Printing natural numbers, odds and evens, Multiplication Tables, Sum of Digits.
- Printing varieties of patterns, Pyramids, Series like Fibonacci and Strong numbers.
- Mathematical : Finding GCD, LCM, Factors, Factorial, Primality Tests etc.
- Break and Continue Keywords, Math Library and Important functions.
- Arrays : Declaring and Defining, Memory level understanding.
- Array operations: Copy, Print, Insert, Delete.
- Frequency of elements, Finding Pairs and Triplets with desired property.
- Sorting Algorithms : Bubble, Selection and Insertion Sort.
- Multi-dimensional arrays, Matrix operations: Addition, Multiplication and Transpose.

- Strings, character array, operations: length, upper and lower casing, copying, sorting, comparing, pattern-matching.
- Using in-built string functions.
- Pointers and Addresses, Pass-by-reference and value, Pointer Arithmetic.
- Constant Pointer, pointer to pointer, Array of Pointers, Pointer and Strings.
- User Defined Functions : Arguments, Return Types etc.., Passing an array and 2-d array.
- Structures, Array of structures and Nested Structures, problem solving using structs.
- Helpful Constructs of C++ and their ease of usage.
- Challenging coding exercises.
- Standard Template Library : Iterators, Maps, Sets, Vectors and other important data structures.

Python

- What is Python?.
- Difference b/w compiled and interpreted language.
- Data Types and Type-conversion.
- Assignment and Arithmetic Operators.
- Bitwise, Relational and Logical Operators.
- Simple-If, If-else, Nested If-else and Else-If ladder.
- Calculator Program.
- String and Membership Operators.
- Identity Operators.
- Associativity and Precedence.

- While loops, printing odds and evens, factorial, prime testing, sum of digits.
- Nested Loops, printing Pyramid, predicting output, GCD and LCM.
- Return, Yield and Generators, Pass by Object Reference.
- Scope and LEGB principle.
- Global and Non-Local Keywords.
- Function call Stack and Recursion.
- Functional Programming, High order functions, Currying, Lambda Functions and Nested Lambdas.
- Map, Filter, Reduce, Any and All, Comprehensions.
- OOPS : setup, modules, packages, errors, Exception Handling, Errors, Classes and Objects.
- Encapsulation, Inheritance, Polymorphism, Access Modifiers, Static variables and Methods.
- Challenging coding exercises and way ahead…

2. Problem Solving in Data Structures and Algorithms

Arrays and Dynamic Arrays

- Introduction to Static and Dynamic Arrays
- Pre-computational Techniques
- Bucketization & sliding window technique
- Reverse lookups & array rearrangements

Mathematics

- Calculating GCD and LCM
- Primality test on a number & in a range

- Prime factorisation & fast factorisation technique
- Sieve of Eratosthenes & Segmented Sieve
- Fundamental of Modular Arithmetics
- Permutations & Combinations

Time and Space Complexities

- Introduction to Time complexity
- Understanding Space complexity
- Order of expressions & recurrence relations
- Finding time & space complexity of code snippets

Recursion

- Recursion tree & recursion stack diagram
- Generating Recurrence Relations and termination conditions
- Divide and Conquer algorithmic paradigm
- Subset problem and its variations: Handling Lexicographical order, duplicates, etc.
- Generating all valid paths, valid parentheses and other recursion based problems
- Breaking a problem into multiple sub-problems

Backtracking

- Backtracking algorithmic paradigm
- Concept of Do and Undo in Backtracking
- Variations of Permutations of a String problem
- Problems on N-Queens & Sudoku solver

Searching

- Binary Search Algorithm, its use to find first & last occurrences
- Concept of separating condition and its use to find single, repeated, peak elements
- Concept of Monotonic Space and introduction to Binary Search on Answer Technique
- Popular Binary Search on Answer problems like Place the cows, Allocate the books & Good base

Sorting Algorithms

- Introduction to comparators and stability in sorting
- Insertion sort & bubble sort
- Classical problems like next greater permutation and smallest subarray
- Merge process and its applications like inversion count
- Quick sort and space-efficient partition
- Sorting in linear time using count sort, radix sort, etc.

Two Pointers

- Two pointer technique and its applications
- Pair/Triplet/Quadruplet sum/difference problems and their patterns
- Two Pointers problem on strings like Longest substring without repetition
- Specific prime factorisation using 2 pointer

Bit Manipulation

- Introduction to bitwise operators
- Properties of XOR operator and its related problems
- Problems on finding minimum XOR, OR of subarrays, etc.
- Bit level hacks

Hashing

- Understanding hashing, hash tables and tree maps
- Classical problems on checking for pair & subarray with given sum/differences
- Geometry related Hashing problems, handling slopes
- String related hashing problems like Substring with all Words, Making Palindromes etc

Stacks

- Introduction to stack data structure
- LIFO principle & stack overflows
- Problems related to Next/Previous Greater/Smaller elements
- Design problems on Stack like Min Stack, etc

Queues

- Introduction to Queue data structure & FIFO principle
- Introduction to Circular queues & Double Ended Queues
- Design problems related to Queues
- Use of Queues in Level Order Traversal and related problems
- Problems related to Deque like Sliding window Max

Linked List

- Introduction to linked lists
- Basic operations like finding middle node, reversing & deleting
- Cycle detection problems
- Merging linked lists iteratively & recursively
- Cloning & flattening a linked lists
- Implementing LRU cache
- Classical problems on rearrangement of nodes & palindrome evaluation

Binary Trees

- Understanding trees, nodes and leaves
- Symmetricity & mirror image of binary tree
- Height & diameter of a binary tree
- Balanced binary tree & related problems
- Conversions from Binary Trees to other Data Structures
- Conversions of Other Data Structures to Binary Trees
- Recursive and Iterative Tree Traversals
- Level order traversal and related problems
- Left & right, top & bottom views of binary tree
- Vertical order traversals
- Serialisation & deserialisation of tree
- Tree re-construction from special traversals
- Concept of LCA and related problems

Binary Search Trees

- Introduction to Binary search tree
- Basic operations like searching, insertion & deletion in BST
- Checking if a Binary tree is a BST
- Conversions of BST to other Data Structures
- Binary Search Tree Iterators and related problems
- LCA in BST

Tries

- Introduction to Tries data structures
- Insertion, deletion & prefix searching in Tries
- String problems like pattern prefix matching and palindrome pairs
- Storing binary numbers in Tries
- Binary Number related problems in Tries

Heaps

- Understanding and building heaps
- Basic heap operations
- Priority queues & Comparators
- Heap sort algorithm
- K Min/Max Problems Related to Heaps
- Task Scheduling Problems, Tying Ropes Problems and other Classical Heap Problems

• Greedy Algorithm

- Introduction to Greedy algorithm: How to prove your greedy
- Classical Greedy Problems like activity selection, job sequencing etc with their proof of correctness.
- Concept of Data compression using Huffman Encoding
- Median in Running Stream Problem

Dynamic Programming

- Concept of States, overlapping Subproblems, optimal Substructures
- Concept of Memoization to Optimise Recursion and form DP
- Concept of Tabulation and Converting Recursive DP to Iterative DP
- Classical 1D DP Problems like Staircase, Maximum Sum Non Adjacent Problems
- Coin Change Problem and Related Variations
- Bounded/Unbounded Knapsack Problems and related Variations
- Longest increasing subsequence in 1D, 2D, 3D
- String Related DP Problems like Longest Common Subsequence, Edit Distance, Regex Matching, etc
- Matrix Chain Multiplication and related problems
- DP on Trees and Bit masking DP

Graph Algorithms

- Introduction to Graphs
- Modelling problems into graphs
- Breadth First Search algorithm
- Shortest Path in an undirected graph

- Classical problems like Knight tour, closest zero, rotten oranges, word zeroes etc.
- Breadth First Search algorithm
- Depth First search algorithm
- Number of connected components & Bipartite graph detection
- Detecting cycles in directed & undirected graphs
- Topological sorting
- Strongly Connected Components
- Articulation Points and Bridges in a Graph
- Shortest Path Algorithms in Weighted Graphs
- Minimum Spanning Tree Algorithms

N-array and Generic Trees

- Hierarchical data structures & Generic N-ary trees
- Height of generic tree from parent tree
- Level order traversal in generic trees
- DFS & BFS based problems on generic trees

Range Queries

- Processing Range queries efficiently
- Square root decomposition technique
- Range sum query using Sparse table
- Segment Trees and Lazy Propagation

3. Computer Science Fundamentals

Operating Systems

- Functions of Operating Systems: CPU Virtualisation, Memory Virtualisation, etc
- CPU Virtualisation using Processes, Process Lifecycle
- Process Scheduling Algorithms
- Context Switch and Interprocess Communication
- Memory Management by OS: Concept of Paging, TLB, Virtual Memory
- Page Replacement Algorithms
- Threads vs Processes: Concurrency vs Parallelism
- Process Synchronisation Solutions: Mutexes, Semaphores and Monitors
- Problems like Producer Consumer, Dining Philosophers & Readers Writers
- Concept of Deadlocks and ways to deal with it

Database Management Systems

- Introduction to Databases and Types of DBs
- ER Diagram Modelling
- CRUD Operations using SQL
- Grouping, Aggregations, Joins and Subqueries in SQL
- Transaction schedules & ACID properties
- Bad Vs Good DB design
- Functional dependencies & Normal Forms

Computer Networks

- OSI Model Layers
- Physical Layer & Data Link Layer
- Network Layer
- IP Addressing Subnet
- Domain Name System
- Routing Algorithms
- Transport Layer, TCP vs UDP
- Application Layer and HTTP

4. Fundamentals of Low Level Design and System Design

Low-Level Design

Clean Coding Principles

- Attributes of a clean code.
- Naming Conventions.
- Clean and Modular Functions, Levels of abstraction.

Principles of Object Oriented Programming

- Classes and Objects.
- Access Modifiers.
- Static variables and methods.
- Constructors and chaining.
- This Keyword.
- Final Keyword and Immutability.

- Encapsulation.
- Polymorphism: Compile and Runtime.
- Inheritance.

SOLID principles

- Single Responsibility Principle.
- Open Closed Principle.
- Liskov Substitution Principle.
- Interface Segregation Principle.
- Dependency Inversion Principle.

Design Patterns and Case Studies

- Builder, Decorator, Factory, Strategy and other useful patterns.
- Design of Tic Tac Toe.
- Steps to break down a design problem.
- Design of Chess.
- Design of Library Management System.
- Design of Food Delivery System.
- Design of Parking Lot.

High-Level Design

Understanding Web architecture & Intro to HLD concepts

- Understanding web architecture components
- Frontend, Databases & Backend
- Understanding reliability, scalability & maintainability

Databases

- Database Replication and Replication Models
- CAP Theorem
- Federation
- SQL vs NoSQL
- Indexing and Indexes
- Sharding

Backend and Frontend System Design

- REST APIs
- Load Balancing & Consistent Hashing
- Microservice Architecture & Service Discovery
- Forward & Reverse Proxies
- Message Queues
- Polling & Web sockets
- Content Delivery Networks

Cache

- Caching and Cache Locations in Web Architecture
- Cache Invalidation & Eviction
- Cache Patterns

Case Studies

- Real world system design case studies like tiny url, Youtube, Google Drive, etc

Electives

5. Frontend Development with ReactJS & Capstone Project

HTML/CSS/JS

- **HTML:** Tags and Attributes, Document Structure, Metadata, links, Images etc, Multimedia, Best Practices, and Intro to Responsive Designs, DOM Tree and using divs for creating.
- **CSS :** Role of CSS, Rules, properties, values, CSS Box Model, Text styling, Color and Background, Flexbox and Grid, Media Queries, SEO Impact, Bootstrap, Tailwind.
- **Core Javascript:** JavaScript Internal Working using Event Loop, Web APIs, JavaScript Specific Concepts like Variable Hoisting, Closures, Arrow Functions, Map/Filter/Reduce, Deep vs Shallow Cloning, Debouncing & Throttling, Objects & Proto - Why is everything an object in JS?, Async JS: Promises, async, await, then, Error Handling, Side project: Tic Tac Toe Game using JS. Integrating timer and timer-based results.

Technologies

- Build Automation Tool.
- Version control and Git.

React Dev

- Intro : Need for a framework like React, Environment Setup, Writing our first app using create-react-app, Structure of a React project.
- Components and Containers : How to write industry standard react code, Functional and Class based, Component Lifecycle and Updates.
- State and Props : Concept of State, Passage of Data across components, PropTypes and Default props.
- Event Handling : Internals of Event Handling in React, Passing arguments to Handlers, Event Pooling and Synthetic event.

- React Component Lifecycle Methods : What are lifecycle methods, Lazy and Suspense, React Context.
- React Hooks: Why need hooks and their types, useState and useEffect Hooks, Custom Hooks.
- React Router : What is Routing, Client side vs Server side, Router setup, Nested Routes, Redirect routes, Pathless routes, Rules for routing.
- Redux : Why redux, Redux actions, reducers and store.
- Local Storage and Cookies : Difference between local storage and cookies, their use-cases.
- Mini and Major projects with deployment.

6. Low Level Design and Concurrency for Experienced Professionals

Clean Coding

- Fundamentals of clean coding principles.
- Best practices for naming variables and functions.
- Modularity and levels of abstraction.

Object Oriented principles & their applications

- Why OOPs? Structuring data and algorithms together.
- Objects and Classes. Constructors, this keyword.
- Final keyword and concept of immutability.
- Pillars of OOPS and their practical relevance: Abstraction, Encapsulation, Polymorphism, Inheritance.
- Static variables and methods.
- 2.6 Problems with Inheritance.
- Making generic functionalities with Polymorphism.
- Access modifiers.

Deciphering Design principles

- Single Responsibility principle.
- Open close principle.
- Liskov Substitution principle.
- Interface Segregation principle.
- Dependency Inversion principle.

Design Patterns and Case Studies

- Top-Bottom flow of thinking before attempting any design problem.
- Design a Library Management System.
- Strategy Design Pattern.
- Builder Design Pattern.
- Factory Design Pattern.
- Abstract Factory Design Pattern.
- Design a Food delivery system like Zomato.
- Decorator Design Pattern.
- Design a Game of Chess.
- Design a Parking Lot.
- State Design Pattern.
- Design an ATM.
- Design a Json Parser Module.
- Observer Design Pattern.
- Design a Stock Trading App.
- Chain of Responsibility Design Pattern.
- Design a Logging System.
- Design an Elevator System.
- Command Design Pattern.
- Design an e-commerce like Amazon.
- Design an in-memory MySQL Server.

Design Patterns and Case Studies

- Design an extensible Caching library.
- Singleton Design Pattern.
- Design a social media Platform like Facebook.
- Remaining important Design Patterns and their practical applications.

Testability

- What is Unit Testing?
- Mocking and Assertion.
- Why need Testing Frameworks: Junit, Mockito?
- Writing robust tests.
- Dependency Injection and Inversion of Control.
- Great Testability is a consequence of Good LLD.
- Things to avoid.
- Dealing with worst cases: PowerMock.

Concurrency vs. Parallelism, Thread safety

- Context switches, task scheduling, Thread stack.
- Race conditions and Locks.
- Reentrant locks and Thread safety measures.

Volatile, Monitors, Thread Synchronization and signaling

- Memory visibility issues.
- Busy waiting vs wait/notify.
- Challenging Exercises, Problem solving with threads.

Deadlocks, Producer Consumer problem, Dining Philosophers problem

- Deadlock, Livelock.
- Coding Dining Philosophers problem, Producer Consumer problem and its variants.

Explicit Locks, ReadWrite Locks Blocking Queues, Synchronizers

- Thread safe Data structures, Hand over Hand locking.
- Creating our own locks and ensuring reentrancy.
- Latch, Semaphore, FutureTask, Barrier.

Thread Pool, Executor Service, Non Blocking Algorithms

- Building our own thread pool, understanding Executor Service.
- Exercises and Problem solving, Building a thread-safe hashmap, lock-striping.
- Atomic Data Types.

7. Backend development in Java Spring Boot & capstone Project

Java Concepts

- Clean Coding and OOPS Concepts (Refer to LLD).
- Reflection : Meta-class, Determining methods, fields and constructors at run-time, Building generic libraries, exploring supertypes and inheritance hierarchy, Building a robust general-purpose serializer and deserializer, class loading, designing a mini dependency injection module.
- Exception Handling : Checked and Unchecked exceptions, exception hierarchy, User defined exceptions, exception chaining, stack-traces, try-catch-finally, try-with-resources and best practices of exception handling.
- Database Programming and JDBC : JDBC Architecture, Drivers, Driver Manager, Using 3P drivers, Creating Connections, Statements, Transactions, Connection pool management, Result sets, Scrollable and Updatable types, Datasource, Closing of resources and other best practices.
- Concurrency : Threads, Race conditions, Memory Visibility, thread signaling and synchronization, Deadlocks, Thread Pools and Executor Service.
- Annotations : Role of Annotations, Meta Annotations(Target, Retention, Inherited etc), creating own annotations
- Generics : Generic classes, methods and expressions, Bounded Generics, Multiple bounds, Wild cards with generics
- Streams and Functional Interface : Need of functional interface, Lambda expressions, Consumer, Predicate, Function etc..., Stream creation, methods, pipeline, lazy loading, order of execution, stream reduction
- Socket Programming : Networking Basics, TCP and IP, Creating Sockets, URLs, Implementing our own client and server.

Technologies

- Build Automation Tool : Maven Commands, Pom file structure, Tags, Dependency Management, Conflict Resolution, Scopes, Local, Remote and Central repository
- Version Control : Intro to Git, Git Commands, Branches, Staging Environment, Merging conflicts
- ORMs : Difference b/w object and relational paradigm, Pros and Cons of using any ORM
- Message Queues and Caches : Concept of Batch and Stream Processing, Message Brokers vs Kafka, Caching techniques and Redis.
- Monoliths and Microservices : Pros and Cons of both architectures, Some real world systems implementing these architectures.
- Unit Testing : Junit, Mockito and Power Mockito, Best practices for testing, Understanding of Coverage metrics.

Core Development

- Intro to Spring Boot : Understanding the need for this framework.
- What is Inversion of Control and Dependency Injection : ApplicationContext, IoC Container, Beans, Types of Autowiring e.g. Field-based, Constructor-based etc and their pros and cons, Component Scanning, XML based configuration and its drawbacks.
- Web Server and MVC : Tomcat Server and its role, Servlets, Servlet Containers, Dispatcher Servlet, MVC architecture
- Writing Controllers : Networking basics, CRUD operations, REST APIs, Important annotations like @Controller , @RequestMapping and many more, Request Handling and URL mapping, Message Converters.
- Databases : Understanding Databases, SQL vs NoSQL, Entity-Relation Design, Normalisation, SQL queries.

- Hibernate, JPA : How ORMs work, Mapping classes, value-types, collections etc, Repository, FindBy Methods and Custom Queries, Pagination, Sorting, Named Queries, Asynchronous processing, Lazy and Eager loading, Transactions, Bulk and Batch Processing, Drawbacks of ORM
- Security : Authentication, Authorization and Access Controls, JWT Tokens
- Serialization and Deserialization : Jackson, Important Annotations like @JsonGetter etc
- Logging : Logging libraries, log-levels and best practices for logging
- Spring Boot Actuator : Why need it, Enabling endpoints, customisation
- Mini and Major Projects