DSA – Placement Supreme Batch

20 Weeks to 12+ LPA Dream Job 🚀

A Step-by-Step DSA Plan in C++/Java to Land High-Paying Roles

🗓️ **Duration: (Module 1) – DSA -** 12 Weeks
💼 **Goal:** Secure packages up to 12+ LPA in 12 Weeks DSA

Week 1: Basics + Patterns

- **Step-by-step Learning**:
    - C++/Java Basics: Input/Output, Data Types, Operators, Loops (for, while), Conditional Statements.
    - Functions: Introduction to functions, Pass by Value vs. Reference, Function Overloading.
    - Patterns: Triangle, Diamond, Pascal's Triangle, Hollow Shapes.
- **Practice**:
    - Simple mathematical problems (e.g., prime numbers, GCD, factorial).
    - Solve 10+ pattern problems.
- **Outcome**:
    - Comfort with basic syntax, control flow, and problem-solving.

Week 2: Arrays + 2D Arrays + Strings

- **Step-by-step Learning**:
    - Arrays: Introduction, Operations (Insertion, Deletion, Traversal), Common Problems (e.g., Max/Min Element, Reverse Array).
    - 2D Arrays: Matrix Initialization, Row/Column-wise Operations, Basic Problems (e.g., Matrix Transpose, Diagonal Sum).
    - Strings: Basics, Standard Library Functions (length (), substr(), etc.), String Reversal, Palindromes.
- **Practice**:
    - 15 problems: Rotating arrays, Matrix problems, and String manipulations.
- **Outcome**:
    - Strong foundation in arrays and strings.

DSA – Placement Supreme Batch

Week 3: Time and Space Complexity + HashMap's

- **Step-by-step Learning**:
  - Complexity Analysis: Big-O Notation, Common Complexities (O(1), O(n), O(n^2)).
  - Space Complexity: How to analyze memory usage.
  - HashMap's: Basics, Implementation, Applications (Frequency Count, Two Sum).
- **Practice**:
  - 10 problems: Frequency counting, Subarrays with a given sum, Two Sum.
- **Outcome**:
  - Ability to analyze the efficiency of code and use hashmaps effectively.

Week 4: Recursion - Part 1

- **Step-by-step Learning**:
  - Basics of Recursion: Base Case, Recursive Case.
  - Problems: Factorial, Fibonacci, Sum of Digits, Power Calculation.
  - Recursion Tree Visualization.
- **Practice**:
  - 10 beginner problems (simple mathematical recursion).
- **Outcome**:
  - Clear understanding of recursion fundamentals.

Week 5: Recursion - Part 2

- **Step-by-step Learning**:
  - Backtracking: Introduction, Problems like N-Queens, Rat in a Maze.
  - Advanced Recursion Problems: Permutations, Subsets, Word Search.
- **Practice**:
  - 10+ intermediate problems on backtracking.
- **Outcome**:
  - Confidence in solving recursion and backtracking problems.

Week 6: Complete OOP's + STL

- **Step-by-step Learning**:
  - Object-Oriented Programming:
    - Classes and Objects.
    - Encapsulation, Abstraction, Inheritance, Polymorphism.
  - STL (Standard Template Library): Basics (Vectors, Pairs, Sets, Maps).
- **Practice**:
  - Write basic programs using OOP principles.
- **Outcome**:
  - Solid understanding of OOP and STL basics.

Week 7: Linked Lists

- **Step-by-step Learning**:
  - Basics of Singly Linked List: Insertion, Deletion, Reversal.
  - Doubly Linked List and Circular Linked List.
  - Problems: Detect Cycle, Intersection of Two Lists.
- **Practice**:
  - 10+ problems on Linked Lists.
- **Outcome**:
  - Mastery over Linked List operations and common problems.

Week 8: Stacks & Queues

- **Step-by-step Learning**:
  - Stack: Implementation using Arrays/Linked Lists, Applications (Balanced Parentheses, Infix to Postfix).
  - Queue and Deque: Implementation, Applications (Sliding Window Maximum).
  - Priority Queue (Heap): Introduction, Min-Heap, Max-Heap.
- **Practice**:
  - 10 problems on Stacks and Queues.

- **Outcome**:
    - Clear understanding of stack and queue operations.

Week 9: Trees & Binary Trees

- **Step-by-step Learning**:
    - Binary Trees: Basics, Traversals (Preorder, Inorder, Postorder, Level Order).
    - Binary Tree Problems: Height, Diameter, Symmetry Check.
- **Practice**:
    - 10+ problems on trees.
- **Outcome**:
    - Strong grasp of tree structures and traversal methods.

Week 10: Binary Search Trees

- **Step-by-step Learning**:
    - Binary Search Tree (BST): Basics, Operations (Insertion, Deletion, Search).
    - Problems: LCA in BST, Validate BST, Kth Smallest Element.
- **Practice**:
    - Solve 10 problems on BST.
- **Outcome**:
    - Confidence in handling BST problems.

Week 11: Priority Queues & Tries

- **Step-by-step Learning**:
    - Priority Queues: Applications in real-world scenarios (e.g., Huffman Coding).
    - Tries: Basics, Insert/Search Words, Prefix Matching.
- **Practice**:
    - 10+ problems on priority queues and Tries.
- **Outcome**:
    - Proficiency in advanced data structures.

DSA – Placement Supreme Batch

Week 12 (Optional-Advanced): Dynamic Programming + Graphs

Step-by-step Learning:

- Basics of DP: Memoization, Tabulation.
- Classic Problems: Knapsack, Subset Sum, Longest Common Subsequence.
- **Graphs**: Graph Representation (Adjacency Matrix/Adjacency List), Traversal (BFS, DFS), Shortest Path Algorithms (Dijkstra's).

Practice:

- 10 problems from beginner to intermediate level.

Outcome:
Ability to break complex problems into smaller subproblems and solve graph-related challenges.

Explanation of Package Ranges

1. **Up to Week 6**: **(Up to 6 LPA)**
    - Students are prepared for **service-based companies** (TCS, Infosys, Wipro, Accenture, Capegmini etc.).
    - Typical problems include array manipulations, basic recursion, and basic hashmaps.
2. **Weeks 7–10**: **(Upto 10 LPA)**
    - Students gain the skills to tackle **mid-level product companies** (e.g., Paytm, Ola, Swiggy).
    - Proficiency in linked lists, stacks, queues, trees, and STL.
3. **Weeks 11–12**: **(12+ LPA)**
    - Mastery of advanced topics like graphs, dynamic programming, and tries prepares students for **top-tier product companies** (e.g., Amazon, Microsoft, Adobe, Google).
    - Solving graph problems and optimizing solutions are essential for salaries **12+ LPA**.

Key Highlights of the 12 Weeks to 12 LPA Dream Job Plan 🚀

- **Structured Curriculum**: Step-by-step learning from basics to advanced DSA.
- **Assignments After Every Class**: Practice problems to reinforce every concept.
- **Total Problems Solved**: Solve 200–300 problems during the course.
- **Placement-Oriented**: Tailored to crack service-based and product-based interviews.
- **Focus on Real-World Skills**: Learn optimization with time and space complexity.

Week 13: Node.js & Express.js (Module 2 – Backend)

- **Node.js Basics:**
  - Introduction to Node.js and event-driven architecture.
  - Working with modules and npm.
  - Asynchronous programming with callbacks, Promises, and async/await.
- **Express.js Basics:**
  - Setting up an Express.js server.
  - Creating routes and middleware for request handling.
- **Minor Project:**
  - Create a basic HTTP server using Node.js.
  - Build a simple Express.js server with basic routing.

Outcome: Understand Node.js fundamentals and how to build simple web servers with Express.js.

Week 14: APIs & Postman

- **API Concepts:**
  - RESTful API architecture and best practices.
  - HTTP methods (GET, POST, PUT, DELETE).
  - Building RESTful APIs with Express.js.
- **Postman for API Testing:**
  - Introduction to Postman.
  - Sending requests and testing API endpoints.
  - Automating tests and workflows using Postman.
- **Minor Project:**
  - Build a RESTful API for a simple user management system.
  - Test the API using Postman.

Outcome: Learn API design principles, build and test APIs using Postman.

Week 15: PostgreSQL & Prisma

- **PostgreSQL Basics:**
  - Introduction to PostgreSQL: querying, joins, and CRUD operations.
  - Setting up and interacting with a PostgreSQL database.
- **Prisma ORM:**
  - Introduction to Prisma and how to set it up with Node.js.
  - Performing CRUD operations with Prisma.
  - Integrating Prisma with PostgreSQL.
- **Minor Project:**
  - Build a RESTful API using Prisma for database interaction with PostgreSQL.

Outcome: Understand relational databases (PostgreSQL), use Prisma for ORM integration.

Week 16: Authentication & Security

- **User Authentication:**
  - Implement JWT authentication.
  - Passport.js for handling different authentication strategies.
- **Security Best Practices:**
  - Preventing XSS, CSRF, SQL Injection.
  - Encrypting passwords and securing sensitive data.
- **Minor Project:**
  - Secure API with JWT and Passport.js.
  - Apply security best practices to your API.

Outcome: Learn how to implement secure user authentication and protect APIs from common vulnerabilities.

Week 17: Backend with Frontend + Testing

- **Integrating Frontend with Backend:**
  - Connecting React with your RESTful API using Axios.
  - Handling responses, errors, and displaying data on the frontend.
- **Testing:**
  - Introduction to unit and integration testing.
  - Using Jest and Supertest to test APIs.
  - Writing test cases for backend functionality.
- **Minor Project:**
  - Build a full-stack application (frontend with React and backend with Node.js and Express).
  - Write test cases for the backend API.

Outcome: Learn how to connect a backend with a frontend and ensure the functionality is working with tests.

Week 18: Deployment

- **Version Control with Git & GitHub**:
  - Learn Git basics (commits, branches, merges).
  - Using GitHub for code collaboration and version control.
- **Deployment:**
  - Deploying Node.js apps with GitHub Pages, Heroku, and MongoDB Atlas.

  - Continuous Integration/Continuous Deployment (CI/CD) basics.
  - Deploy to AWS (EC2, S3, RDS, etc.).
- **Minor Project:**
  - Deploy a personal website on GitHub Pages.
  - Deploy a full-stack application to Heroku.

Outcome: Learn how to deploy applications and collaborate using Git/GitHub.

Projects Overview:

- Minor Projects: Small web applications focused on individual backend topics (e.g., user management, API testing, authentication).
- Major Projects:
  - Full-stack web applications (e.g., blogging platform, user management system, social media platform).

Week 18: Aptitude Basics + Logical Reasoning

• **Quantitative Aptitude**:

- **Number Systems**: Understanding types of numbers (odd, even, prime), divisibility rules.
- **Percentages & Profit and Loss**: Calculating profit, loss, discount, and percentage changes.
- **Time, Speed, and Distance**: Solving relative speed, boats & streams, and train problems.
- **Ratios & Proportions**: Solving problems with simple and compound ratios.
  • **Logical Reasoning**:
- **Syllogism**: Identifying relationships and making deductions.
- **Blood Relations**: Solving family tree-based problems.
- **Number Series**: Identifying and completing patterns in number series.
  • **Practice**:
- Solve 20-30 practice problems each day for reinforcement.
  Outcome: Strengthen basic concepts in quantitative aptitude and logical reasoning.

Week 19: Advanced Aptitude + Mock Practice

• **Advanced Quantitative Aptitude**:

- **Permutations & Combinations**: Learn and practice basic counting principles, and combination/permutation problems.
- **Probability**: Basic probability theory, and solving problems on dice, cards, and coins.

- **Work and Time (Advanced)**: Solving complex work-related problems involving pipes, cisterns, and efficiency.
- **Averages & Mixtures**: Solving problems with weighted averages and mixture/allegation rule.
  • **Logical Reasoning**:
- **Critical Reasoning**: Deductive and inductive reasoning, and analyzing arguments and conclusions.
- **Puzzles & Data Interpretation**: Solving complex seating arrangement and data-based puzzles.
  • **Mock Tests**:
- Take 2-3 timed mock tests to practice real exam conditions.
- Analyze the results and focus on weaker areas.
  Outcome: Master advanced topics in aptitude and practice solving problems in mock tests under time constraints.

Week 20: System Design Fundamentals + Case Studies

• **Introduction to System Design**:

- **System Design Basics**: Learn the fundamentals of system design (scalability, reliability, fault tolerance).
- **Key Concepts**: Understand high-level system design concepts like load balancing, sharding, and distributed systems.
  • **Design Patterns & Principles**:
- **Common Design Patterns**: Study design patterns like Singleton, Factory, Observer, etc.
- **Database Design**: Learn about normalization, indexing, and ER diagrams.
  • **Case Studies**:
- Design real-world systems such as an e-commerce platform, URL shortening service, or social media application.
- Focus on scalability, performance, and fault tolerance.
  • **Practice**:
- Work through 2-3 real-world system design case studies, explaining your thought process and design decisions.
  Outcome: Build a foundational understanding of system design and be able to design scalable systems.

DSA – Placement Supreme Batch

Week 21: Mock SDE Sessions (Aptitude, System Design & DSA Coding)

• **Mock System Design Interviews**:

- Conduct 2-3 mock system design interviews with peers/mentors.
- Focus on breaking down requirements, designing scalable systems, and explaining trade-offs.
  • **Coding Interviews**:
- **Data Structures & Algorithms**: Solve problems related to arrays, linked lists, trees, graphs, and dynamic programming.
- **Practice**: Take 2-3 mock coding interviews to simulate real-time problem-solving.
- Focus on writing clean code and explaining the solution in detail.
  • **Mock Aptitude Sessions**:
- Take 2-3 full-length timed mock aptitude tests.
- Focus on time management and analyze performance.
  • **Feedback & Iteration**:
- Review mock interview results and focus on areas of improvement.
- Iterate on problem-solving approaches based on feedback.
  Outcome: Refine interview skills through mock system design, coding, and aptitude sessions to prepare for real interviews.