In the name of ALLAH

# Coding Interview Course ( BD Big Tech and FAANG)

## A Complete Guideline for Your Software Engineering Job Interview

**80+** Live Classes, **20+** Industry Specific Classes, **400+** Leetcode Problems, Course Duration: **6 Months**, **Weekly 3 Classes**.

Course Fee: **6000/-**

### **Course Features:**

1. 🧑‍🏫 **80+ Live Classes:** Covering all essential topics comprehensively.
2. 🎯 **20+ Industry-Specific Guideline Classes:** Get insights from industry Engineers.
3. 🧩 **400+ Handpicked LeetCode Problems:** Focused on top interview patterns and FAANG questions.
4. 📽️ **Videos of Each Class:** Access recordings for revision anytime.
5. 🎥 **Solution Videos for All 400+ Problems:** Master problem-solving with detailed explanations.
6. 📝 **MCQ Tests for Each Topic:** Validate your understanding and track progress.
7. 🤝 **Mock Interviews:** Top students will conduct interviews with each other to simulate real-world scenarios.
8. 📄 **CV Reviews and Soft Skills Development:** Build a professional resume and refine interpersonal skills.
9. **Weekly 3 Classes** ( 2 Coding Classes, 1 CSE Fundamental )

### **What Makes This Course Unique?**

The **400 handpicked LeetCode problems** are the most valuable part of this course! These problems cover:

* 🚀 **Top Interview Problems of FAANG** (Meta, Amazon, Apple, Netflix, Google).
* 🔍 **Frequently Asked FAANG Questions** to help you prepare smartly.
* 📘 **Cracking the Coding Interview Problems** aligned with industry standards.
* 💡 **The 23 Problem Patterns of Coding Interviews**, ensuring you master the most recurring patterns.

### **Who Can Join?**

* 🎓 **Final-Year Students:** Preparing for their first job or career shift.
* 💼 **Job Seekers:** Looking to crack interviews and secure roles in top companies.
* 🚀 **Working Professionals:** Aspiring to join BD Big Tech or FAANG.

### **Pre-requisite:**

* ✅ Must have a basic understanding of **C++** programming.

## **Coding Interview Preparation**

### **Core Foundations:**

**Complexity Analysis**, Time and space complexity basics, Big-O notation, **Array**, **Vector**, Array simulations, rotations, and manipulations, **Matrix Operations**, 2D matrix simulations and transformations.

### **Strings and Patterns:**

**String, String Manipulations**, Simulation, Reverse, Parsing, **Palindrome and Anagrams**, string rotation, char count

### **Data Structures with STL:**

Map, Set, Stack, Queue, Deque, Priority Queue (Min-Heap, Max-Heap), LRU Cache, Circular Queue, Custom Comparators in Priority Queues.

### **Mathematical and Bit Manipulation Techniques:**

**Bit Manipulation**, Efficient binary operations for problem-solving, **Mathematical Foundations**, Number theory, modular arithmetic, and digit manipulations, Bit masking, XOR operator and its magics.

### **Greedy Algorithms:**

Solving optimization problems with local decisions.

### **Recursion and Backtracking:**

Problem-solving using recursion and exploring all possibilities, N-Queens problem, Subsets, Permutations, Generate Parenthesis, Fibonacci sequence.

### **Sorting:**

**Sorting Algorithms**, Bubble, Merge, Quick, and Insertion Sor, Difference between Merge Sort and Quick Sort, Counting Sort, Heap Sort

### **Searching:**

**Binary Search Variations**, Lower/Upper bounds, Bisection.

### **Two Pointers and Sliding Windows:**

**Two pointers and sliding windows**

### **Linked Lists:**

Singly, Doubly, and Circular Linked Lists, Reverse, Rotate, Merge, and Detect Loops, Reverse a Linked List, Detect and Removing Loops in a Linked List, merge two Linked Lists, Find the Middle Element of a Linked List, Intersection of Two Linked Lists, Clone a Linked List with Random Pointers, Rotate a Linked List, Add Two Numbers Represented by Linked Lists

### **Graphs and Trees:**

DFS, BFS, Topological Sort, Cycle Detection, Island, Dijkstra, Bellman-Ford, Floyd-Warshall.

### **Binary Tree and Binary Search Tree:**

Binary Search Trees, Balanced Trees, Heap Sort, Balanced binary search tree, Tree Construction from Traversals, Lowest Common Ancestor (LCA), Tree Diameter, Tree Balancing Techniques, Depth-First Search (DFS) and Breadth-First Search (BFS), Maximum Path Sum, Introduction to Heap, Types of Heaps (Min-Heap, Max-Heap), Heap Operations (Insert, Extract, Peek), Heap Applications (Priority Queues, Heap Sort), Introduction to Binary Search Tree (BST), BST Operations (Insertion, Deletion, Search), Tree Traversal (In-order, Pre-order, Post-order)

### **Dynamic Programming:**

0-1 Knapsack, Coin Change, Longest Increasing Subsequence (LIS), Longest Common Subsequence (LCS). Longest Palindromic Substring ( Manachers algorithm )

### **Advanced-Data Structures:**

**Trie**: Insert, Search, and Applications (Autocomplete, Spell Checker), **Segment Trees**: Range queries and updates, KMP string algorithm

## 

## **CSE Fundamentals**

### **Object-Oriented Programming (OOP) Using C#:**

1. **Fundamentals**:
   * Classes and Objects
   * Encapsulation, Inheritance, Polymorphism, and Abstraction
2. **Advanced Topics**:
   * Constructor and Destructor
   * Method Overloading and Overriding
   * Abstract Classes and Interfaces
   * What is runtime and compile time polymorphism

#### **Practical Exercises**

* Hotel Booking System
* Parking Lot
* Chat Server

### **Database Management System (DBMS):**

* **Basics**:
  + SELECT, INSERT, UPDATE, DELETE.
  + WHERE, GROUP BY, HAVING, and ORDER BY.
* **Intermediate Concepts**:
  + Joins: INNER, LEFT, RIGHT, FULL OUTER JOIN.
* **Advanced Topics**:
  + ACID Properties.

### **Design Principles:**

1. **Basics**:
   1. DRY, KISS, YAGNI
2. **Intermediate Concepts**:
   1. SOLID Principles

### **System Design:**

#### **Core Concepts**

1. **Basic Components**:
   * Load Balancers, Caching, Proxies, and Databases.
   * Horizontal and Vertical Scaling.
2. **Key Topics**:
   * Designing Scalable Systems: Consistency, Availability, and Partition Tolerance (CAP Theorem).
   * Microservices vs. Monoliths.
   * Distributed Systems and Databases.

#### **Case Studies and Practical Exercises**

* Design a **URL Shortener System** (e.g., TinyURL).
* Create a **Messaging System** (e.g., WhatsApp).
* Design **Instagram's Newsfeed System**.

### **Operating Systems (OS):**

#### **Core Concepts**

1. **Basics**:
   * Process vs. Thread.
   * CPU Scheduling Algorithms: FCFS, SJF, Round Robin, Priority Scheduling.
   * Memory Management: Paging, Segmentation, Virtual Memory.
2. **Intermediate Concepts**:
   * Synchronization: Mutex, Semaphore, Deadlock Avoidance (Banker’s Algorithm).
   * File Systems and Disk Scheduling.
   * Interprocess Communication (IPC).

#### **Practical Exercises**

* Implement a simple **Multithreading Program** in C++/Java.
* Solve deadlock problems (e.g., detect or avoid deadlocks).