Detailed Placement Training Curriculum (40 Hours)

An in-depth program for technical interview preparation with industry-relevant topics

Prepared by **CareerBrook** Training Team

25-11-2025

1 Overview

This 40-hour curriculum is designed to equip candidates with the technical skills required for placements at top tech companies. Each topic is covered on a dedicated page, with detailed subtopics and explanations of their importance in interviews. The program emphasizes practical problem-solving, theoretical foundations, and real-world applications.

2 Data Structures and Algorithms (12 hrs)

Importance and Interview Relevance: Data Structures and Algorithms (DSA) form the backbone of technical interviews at top tech companies. Mastery of DSA demonstrates problem-solving ability, logical thinking, and code optimization skills. Interviewers test candidates on efficient algorithm design, complexity analysis, and implementation under time constraints, making this topic critical for coding rounds.

2.1 Introduction & Complexity (2 hr)

Introduction to algorithm analysis, focusing on asymptotic notations: **Big-O**, **Big-**Ω, and **Big-**Θ. Analyze best, average, and worst-case scenarios for algorithms like loops and recursion.

2.2 Arrays & Strings (2 hrs)

Explore array operations: indexing, linear/binary search, sorting (bubble, insertion, selection). String manipulation includes palindrome checks and substring searches. Master the sliding window technique for problems like maximum sum subarray or longest substring without re- peats.

2.3 Linked Lists (4 hrs)

Study singly, doubly, and circular linked lists. Perform operations: insertion, deletion, traver- sal. Solve problems like list reversal (iterative/recursive) and cycle detection using Floyd's tor- toise and hare algorithm.

2.4 Stacks & Queues (4 hrs)

Implement stacks (LIFO) and queues (FIFO, circular) using arrays or linked lists. Applications include balanced parentheses checking and LRU cache simulation using doubly-linked lists and hash maps.

3 C / C++ / Python Efficient Coding (8 hrs)

Importance and Interview Relevance: Proficiency in C, C++, and Python is essential for coding interviews, as these languages are widely used in industry. Interviewers assess candidates' ability to write clean, efficient, and error-free code under time pressure. Knowledge of language-specific optimizations and best practices is crucial for solving complex problems in coding rounds.

3.1 Pseudo code (2 hrs)

Cover the most frequently asked questions by writing Pseudo code. Become strong in using loops, conditional operators and functions.

3.2 C/C++ Essentials (2 hrs)

Revise all the basic topics and pointers, memory management (malloc/free, new/delete)

3.3 Python for Efficiency (2 hrs)

Revise all the basic topics which helps in writing code for the frequent challenges are given in most of the interviews.

3.4 Coding Practices (2 hrs)

Adopt clean code principles: meaningful naming, small functions. Implement robust error handling and come up with test cases.

4 System Design (4 hrs)

DevOps Tools like JIRA, GitHub, Jenkins, CI/CD

Importance and Interview Relevance: System design is critical for senior roles and interviews at companies, where candidates must architect scalable systems. It tests the ability to balance trade-offs, understand distributed systems, and design real-world applications, making it a key component of technical interviews.

4.1 Introduction to System Design (1.5 hr)

Understand basics of system design terminology, scalability (horizontal/vertical), availability (redundancy, failover), and CAP theorem (Consistency, Availability, Partition tolerance trade-offs).

4.2 SDLC (30 mins)

Understand the basic flow of software development life cycle in most of the domain.

4.3 GitHub (1 hr)

Understand the flow of GitHub operations from creating repo, clone, add, commit, push, branch, etc… to have ideas on how development happens in a collaborative way.

4.4 Jenkins CI/CD (1 hr)

Understand how to define CI/CD pipeline using Jenkins.

5 OS Commands, Concepts, Shell Scripting Tools (6 hrs)

Importance and Interview Relevance: Operating system concepts are vital for low-level programming roles and system design interviews. Companies test candidates on process management, file systems to evaluate their understanding of how software interacts with hardware.

5.1 Basics of OS concepts (2 hrs)

Explore topics related to Operating System like kernel, user, process, threads, list of IPC.

5.2 Linux Commands and shell scripting (2 hrs)

Understand how to debug issues using OS commands and automate using Shell scripting.

5.3 Linux Tools (2 hr)

Explore Linux tools to debug system failures and security issues like WireShark, nmap, lsof, netstat, ptrace, dmesg, etc…

6 RDBMS (2 hrs)

Importance and Interview Relevance: Relational databases are fundamental for back-end roles, especially at companies, which rely heavily on data management. Interviewers test SQL proficiency, database design, and optimization skills to ensure candidates can handle large-scale data efficiently.

6.1 Basics of RDBMS and Case Studies (1 hr)

Understand the use of DB and Relational DB, design databases for banking systems (accounts, transactions)

6.2 SQL Basics (1 hr)

Cover DDL (CREATE, ALTER, DROP), DML (INSERT, UPDATE, DELETE), and queries with joins (INNER, LEFT, RIGHT) and aggregations (COUNT, SUM, AVG).

7 Object-Oriented Programming (OOPs) (2 hrs)

Importance and Interview Relevance: OOP is a cornerstone of software engineering, tested in interviews to assess code modularity and design skills. Companies evaluate candidates' ability to apply OOP principles and design patterns to create maintainable, scalable codebases.

7.1 OOP Basics (1 hr)

Define classes and objects, implement encapsulation with access modifiers (public, private, protected), and use constructors/destructors. Draw UML diagram of class diagrams.

7.2 Inheritance & Polymorphism (1 hr)

Explore inheritance (single, multiple) and polymorphism (overloading, overriding with virtual functions in C++).

8 Networking (2 hrs)

Importance and Interview Relevance: Networking knowledge is crucial for roles involving distributed systems or backend development. Companies test candidates on protocols and socket programming to ensure they can build robust, networked applications.

8.1 Networking Basics (1 hr)

Study TCP/IP stack (4 layers). Explore IP addressing (IPv4/IPv6) and subnetting.

8.2 Network components (1 hr)

Hubs, Routers, switches, gateways, IP-filter. Hands-on commands related to networking on Linux OS.

9 Frequently Asked Interview Questions

1. **Pseudocode: Write pseudocode to print all even numbers from 1 to 100.**
2. **Pseudocode: Write pseudocode to check if a number is prime.**
3. **Pseudocode: Write pseudocode to find the largest number in an array.**
4. **Pseudocode: Write pseudocode for linear search.**
5. **Pseudocode: Write pseudocode to print the first 10 Fibonacci numbers.**
6. **Pseudocode: Write pseudocode to count vowels in a string.**

7. Pseudocode: Write pseudocode to calculate the grade of a student based on marks.

8. Pseudocode: Write pseudocode to reverse a string.

9. Pseudocode: Write pseudocode to check if two strings are anagrams.

10. Pseudocode: Write pseudocode to find the sum of digits of a number.

11. Pseudocode: Write pseudocode for bubble sort.

12. Pseudocode: Write pseudocode to count even and odd numbers in a list.

13. Pseudocode: Write pseudocode to print a star pyramid pattern.

14. Pseudocode (Control Flow): Write pseudocode to check if a year is a leap year.

15. Pseudocode: Write pseudocode to remove duplicates from an array.

16. Arrays: What is an array? Why is it used?

17. Arrays: How do you insert an element into an array?

18. Arrays: How do you traverse an array?

19. Strings: What is the difference between a string and a character array?

20. Linked List (Concept): What is a linked list? Where is it used?

21. Stack: What is a stack? Mention two real-life examples.

22. Queue: What is a queue? Mention two real-life examples.

23. Stack Usage: How do you check balanced brackets using a stack?

24. Searching: What is the difference between linear search and binary search?

25. Sorting: What is bubble sort? Why is it considered inefficient?

26. Python/C/C++ Basics data types ?

27. Python/C/C++ Input: How do you take input from a user?

28. Python/C/C++ Conditions: Write Python code to check if a number is odd or even.

29. Python/C/C++ Loops: How do you iterate through the elements of a list?

30. Python/C/C++ Strings: How do you convert a string to lowercase?

31. Python/C/C++ File Handling: How do you read a file in Python?

32. Python/C/C++ Functions: How do you define a function?

33. Python Dictionary: What is a dictionary in Python?

34. Python/C/C++ code: How to read data from command line arguments?

35. Python/C++ code: Explain about function overloading.

36. OS Basics: What is an operating system?

37. Process vs Program: What is the difference between process and program?

38. Memory: What is RAM?

39. OS Scheduling: What is Round Robin scheduling?

40. Linux Commands: What do these commands do: `ls`, `pwd`, `mkdir`?

41. File Permissions: What does `chmod` command do?

42. Shell Scripting: How do you print "Hello World" in a shell script?

43. Logs: What is the purpose of using `dmesg`?

44. DBMS Basics: What is a database?

45. Tables: What is a table, row, and column?

46. Primary Key: What is a primary key?

47. Foreign Key: Why do we need foreign keys?

48. SQL Select: Write a query to select all records from a table.

49. SQL Where: Write a query to fetch employees with salary > 25,000.

50. SQL Update: Write a query to update a student's marks.

51. SQL Delete: Write a query to delete an employee record.

52. Class vs Object: What is the difference between a class and an object?

53. Encapsulation: What is encapsulation?

54. Inheritance: What is inheritance?

55. Polymorphism: What is method overloading?

56. Abstraction: What is abstraction?

57. Real-Life OOP: Give real-life examples of inheritance and polymorphism.

58. Networking Basics: What is a network?

59. Topology: What is a network topology? Name any 3 topologies.

60. Protocols: What is a protocol in networking?

61. IP Address: What is an IP address?

62. IPv4 vs IPv6: What is the difference between IPv4 and IPv6?

63. Router: What is the function of a router?

64. Switch: What is the function of a switch?

65. Hub vs Switch: What is the difference between a hub and a switch?

66. TCP/IP Layers: What are the layers of the TCP/IP model?

67. Subnetting: What is subnetting? Why is it needed?