**1. Java Basics & Fundamentals**

1. Write a Java program to print "Hello, World!".
2. Write a program to swap two numbers without using a third variable.
3. Convert an integer to binary, octal, and hexadecimal without built-in functions.
4. Reverse an integer without converting it to a string.
5. Check if a number is even or odd without using an if-else statement.
6. Convert a floating-point number to a string without using .toString().
7. Find the largest prime factor of a number.
8. Calculate the factorial of a number using recursion and iteration.
9. Check if a number is a power of two.
10. Convert Roman numerals to integers.
11. Find the greatest common divisor (GCD) of two numbers using recursion.
12. Generate a Fibonacci sequence using iteration and recursion.
13. Check if a string is a palindrome (case-sensitive and case-insensitive).
14. Reverse words in a string without using built-in functions.
15. Implement a function to check for anagrams.
16. Find the first non-repeating character in a string.
17. Implement a function to find the longest common prefix in an array of strings.
18. Implement a simple calculator using switch-case.
19. Write a Java program to calculate the sum of digits of a number using recursion.
20. Implement the power function without using Math.pow().
21. Find the maximum and minimum numbers in an array without using sorting.
22. Write a Java program to find the missing number in an array of 1 to N.
23. Find the second largest and second smallest elements in an array.
24. Implement a program to count vowels and consonants in a given string.
25. Convert a number to words (e.g., 123 → "One Hundred Twenty-Three").
26. Implement a simple number guessing game in Java.
27. Implement the Collatz Conjecture in Java.
28. Convert a decimal number to binary without using built-in functions.
29. Implement a function to check if a given string contains only digits.
30. Find all the prime numbers up to N using the Sieve of Eratosthenes.

**2. Control Flow & Loops**

1. Print numbers from 1 to 100 without using a loop.
2. Print the Fibonacci series up to N using recursion.
3. Generate Pascal’s Triangle for a given number of rows.
4. Print Floyd’s Triangle using nested loops.
5. Implement FizzBuzz (replace multiples of 3 with "Fizz" and 5 with "Buzz").
6. Find all prime numbers between two numbers.
7. Find the sum of digits of a number using loops.
8. Implement a program to print the multiplication table of a given number.
9. Print a pattern of numbers in a pyramid format.
10. Find the longest palindrome substring in a given string.
11. Implement a program to find Armstrong numbers within a given range.
12. Print a diamond pattern using loops.
13. Count the occurrences of each character in a string.
14. Implement a number reversal program using loops.
15. Implement the LCM (Least Common Multiple) of two numbers.
16. Check if a number is a palindrome using loops.
17. Implement the Tower of Hanoi problem using recursion.
18. Implement a menu-driven program for a simple ATM system.
19. Print numbers in a spiral matrix format.
20. Implement a function to generate random numbers without using Random class.
21. Find all subsets of a given set (Power Set).
22. Print a butterfly pattern using nested loops.
23. Implement the N-Queens problem using backtracking.
24. Implement a function to generate a magic square.
25. Find the sum of all digits in an integer recursively.
26. Implement a program to count words in a sentence.
27. Implement a function to check if a number is a perfect square.
28. Implement a function to print all Armstrong numbers up to 1000.
29. Implement a function to find the number of trailing zeroes in factorial.
30. Implement a function to generate a Collatz sequence for a given number.

**3. Object-Oriented Programming (OOPs) in Java**

**Classes & Objects**

1. Create a Person class with properties like name, age, and gender.
2. Implement a Student class with methods to display student details.
3. Implement a BankAccount class with deposit and withdrawal functionality.
4. Implement a Car class with attributes like brand, model, and price.
5. Implement method overloading in a Calculator class.

**Inheritance & Polymorphism**

1. Create a Vehicle class and extend it to Car and Bike.
2. Demonstrate method overriding using a Shape class.
3. Implement an interface Animal with a speak() method.
4. Implement polymorphism in a Payment class (CreditCard, UPI, PayPal).
5. Demonstrate upcasting and downcasting in Java.

**Encapsulation & Abstraction**

1. Implement encapsulation in an Employee class with getters and setters.
2. Create an abstract class Shape with area() and perimeter() methods.
3. Implement a real-world example of abstraction (ATM system).
4. Demonstrate the final keyword with a class and method.
5. Implement a singleton class in Java.

**Composition & Association**

1. Implement a Library class containing a list of Book objects.
2. Implement an Order class that contains multiple Item objects.
3. Demonstrate aggregation using a Department and Employee relationship.
4. Implement a program to simulate a college system (students, teachers, subjects).
5. Demonstrate static methods and variables in a Java class.
6. Implement a simple hotel reservation system using OOPs concepts.
7. Implement a program for railway reservation using encapsulation.
8. Implement a program to track employee attendance using OOPs.
9. Create a shopping cart system using an Item and Cart class.
10. Implement a movie ticket booking system using Java classes.
11. Implement an employee payroll system using Java OOP concepts.
12. Create a restaurant menu ordering system using OOP principles.
13. Implement an online banking system with deposit, withdraw, and transfer methods.
14. Implement a flight reservation system using multiple classes and relationships.
15. Design an inventory management system for a warehouse.

**4. Java Collections & Data Structures**

1. Implement an ArrayList from scratch.
2. Implement a LinkedList from scratch.
3. Implement a Stack using an array and a LinkedList.
4. Implement a Queue using an array and a LinkedList.
5. Implement a circular queue using an array.
6. Implement a Priority Queue using a MinHeap.
7. Find the kth largest element in an array using a PriorityQueue.
8. Implement a HashMap from scratch.
9. Implement a LRU (Least Recently Used) cache using LinkedHashMap.
10. Implement a Trie (Prefix Tree) for storing words.
11. Implement a binary search tree (BST) with insert, delete, and search operations.
12. Find the height of a binary tree.
13. Implement BFS and DFS traversal of a binary tree.
14. Implement a graph using an adjacency list and perform BFS and DFS.
15. Detect a cycle in a directed and an undirected graph.
16. Implement Dijkstra’s algorithm for shortest path.
17. Implement the Floyd-Warshall algorithm for all-pairs shortest path.
18. Implement Kruskal’s algorithm for minimum spanning tree.
19. Implement Topological Sorting using Kahn’s Algorithm.
20. Find the shortest path in a maze using BFS.
21. Implement a max heap and min heap using an array.
22. Implement a Bloom filter in Java.
23. Find the intersection of two sorted arrays efficiently.
24. Implement a disjoint set (union-find) with path compression.
25. Find the longest consecutive subsequence in an unsorted array.
26. Implement a sliding window maximum problem using a Deque.
27. Implement a LFU (Least Frequently Used) cache in Java.
28. Implement a KMP algorithm for pattern matching.
29. Implement a Rabin-Karp algorithm for substring search.
30. Implement a Ternary Search Tree.

**5. Java Multithreading & Concurrency**

1. Create a simple thread in Java using Thread and Runnable.
2. Implement a producer-consumer problem using wait-notify.
3. Implement a producer-consumer problem using BlockingQueue.
4. Implement a thread-safe Singleton class.
5. Implement an executor framework to run multiple tasks concurrently.
6. Use Callable and Future to return results from threads.
7. Implement a thread-safe counter using Atomic variables.
8. Implement a reentrant lock instead of synchronized.
9. Implement a deadlock scenario and resolve it using ordering.
10. Implement a read-write lock mechanism.
11. Implement a dining philosophers problem using semaphores.
12. Implement a thread pool from scratch.
13. Implement a custom blocking queue using wait-notify.
14. Implement a parallel merge sort using ForkJoin framework.
15. Implement parallel processing using Java Streams.
16. Implement a multi-threaded web crawler.
17. Implement a multi-threaded file reader and writer.
18. Implement a bank transaction system with thread safety.
19. Implement a ticket booking system with multiple users accessing seats.
20. Implement a distributed task queue using threads.
21. Implement the sleeping barber problem using Java Concurrency.
22. Implement a thread-safe circular buffer.
23. Implement a web scraping tool using multithreading.
24. Implement a concurrent data pipeline using Java Streams.
25. Implement a stock trading system where multiple users place orders concurrently.
26. Implement an in-memory cache with automatic eviction policies.
27. Implement a parallel file processing system using Java 8 Streams.
28. Implement a flight booking system where multiple users book seats.
29. Implement a thread-safe rate limiter using a token bucket algorithm.
30. Implement a batch processing system using multi-threading.

**6. Java 8 Features (Lambdas, Streams, Functional Programming)**

1. Write a program using lambda expressions to filter even numbers from a list.
2. Implement a functional interface and use it with lambda expressions.
3. Convert a list of strings to uppercase using Java Streams.
4. Find the sum of all even numbers in a list using Streams.
5. Convert a list of integers into a list of squares using Streams.
6. Find the second highest number in a list using Streams.
7. Implement a program to sort a list of employees based on salary using Streams.
8. Find the frequency of each word in a given string using Streams.
9. Convert a List<String> to a Map<String, Integer> using Collectors.
10. Implement parallel stream processing on a large dataset.
11. Find the most repeated word in a list of strings using Streams.
12. Find all transactions of a particular year from a list of transactions.
13. Implement a custom collector for grouping elements in Streams.
14. Implement a program to check if all elements in a list satisfy a condition.
15. Implement a method reference for sorting a list.
16. Implement an optional value handling in Java.
17. Implement a program to partition a list of integers into even and odd.
18. Implement a program to find the longest string in a list using Streams.
19. Implement a function to remove duplicates from a list using Streams.
20. Implement a function to convert a list of maps to a single map.
21. Implement a program to calculate the average salary of employees using Streams.
22. Implement a flatMap example to flatten a list of lists.
23. Implement an example of reducing elements using Streams.
24. Implement an infinite stream to generate Fibonacci numbers.
25. Implement a program to merge two lists using Streams.
26. Implement a custom comparator using a lambda function.
27. Implement a program to process a large CSV file using Streams.
28. Implement a real-time event processing pipeline using Streams.
29. Implement a word count program using Streams and Collectors.
30. Implement a program to paginate results from a list using Streams.

**7. Java Design Patterns**

1. Implement a Singleton pattern.
2. Implement a Factory pattern.
3. Implement an Abstract Factory pattern.
4. Implement a Builder pattern.
5. Implement a Prototype pattern.
6. Implement an Adapter pattern.
7. Implement a Bridge pattern.
8. Implement a Decorator pattern.
9. Implement a Facade pattern.
10. Implement a Proxy pattern.
11. Implement a Flyweight pattern.
12. Implement a Chain of Responsibility pattern.
13. Implement a Command pattern.
14. Implement an Interpreter pattern.
15. Implement an Iterator pattern.
16. Implement a Mediator pattern.
17. Implement a Memento pattern.
18. Implement an Observer pattern.
19. Implement a State pattern.
20. Implement a Strategy pattern.
21. Implement a Template Method pattern.
22. Implement a Visitor pattern.
23. Implement an MVC architecture.
24. Implement a Publish-Subscribe model.
25. Implement an Event-Driven Architecture.
26. Implement a Dependency Injection pattern.
27. Implement a Circuit Breaker pattern.
28. Implement a Rate Limiter pattern.
29. Implement a Service Locator pattern.
30. Implement an Event Queue pattern.

**8. Spring Boot & Spring Framework**

1. Create a Spring Boot REST API with CRUD operations.
2. Implement a Spring Boot REST API with JWT authentication.
3. Implement Role-Based Access Control (RBAC) in Spring Security.
4. Implement OAuth2 authentication with Spring Security.
5. Create a Spring Boot application with a PostgreSQL database.
6. Implement caching in Spring Boot using Redis.
7. Implement soft delete functionality using Hibernate.
8. Implement a global exception handler using @ControllerAdvice.
9. Implement a request rate limiter in Spring Boot.
10. Create a Spring Boot application with multiple data sources.
11. Implement a file upload and download API using Spring Boot.
12. Implement API versioning in Spring Boot.
13. Implement Spring AOP for logging method execution.
14. Implement an email notification system in Spring Boot.
15. Implement database connection pooling using HikariCP.
16. Implement WebSockets in Spring Boot for real-time chat.
17. Create a Spring Boot API with Circuit Breaker using Resilience4j.
18. Implement a job scheduling service using Spring Batch.
19. Implement pagination and sorting in a REST API using Spring Data JPA.
20. Implement a distributed locking mechanism using Redis in Spring Boot.
21. Implement a GraphQL API using Spring Boot.
22. Implement a REST API that supports filtering using QueryDSL.
23. Implement a microservices-based architecture using Spring Cloud.
24. Implement asynchronous processing in Spring Boot using @Async.
25. Implement a Kafka producer-consumer using Spring Boot.
26. Implement an event-driven architecture with Spring Boot and RabbitMQ.
27. Implement a dynamic configuration service using Spring Cloud Config.
28. Implement an audit logging system using Spring Boot.
29. Implement a Swagger documentation API for a Spring Boot project.
30. Implement a custom starter in Spring Boot.

**9. Microservices Architecture**

1. Implement a microservices architecture using Spring Cloud.
2. Implement API Gateway with Spring Cloud Gateway.
3. Implement a distributed tracing mechanism using Zipkin.
4. Implement service discovery using Eureka.
5. Implement centralized configuration using Spring Cloud Config.
6. Implement circuit breaker functionality using Resilience4j.
7. Implement a message queue-based communication using Kafka.
8. Implement an event-driven architecture using RabbitMQ.
9. Implement a rate limiter in API Gateway.
10. Implement inter-service communication using Feign Clients.
11. Implement authentication & authorization using Keycloak in microservices.
12. Implement log aggregation using ELK Stack.
13. Implement an order management system using microservices.
14. Implement a Saga pattern for handling distributed transactions.
15. Implement a bulkhead pattern for failure isolation.
16. Implement service-to-service authentication in microservices.
17. Implement an API documentation system for microservices.
18. Implement a distributed caching strategy in microservices.
19. Implement a blue-green deployment strategy for microservices.
20. Implement a canary deployment strategy in Kubernetes for microservices.
21. Implement Kubernetes service autoscaling for microservices.
22. Implement a microservices authentication system using JWT and OAuth2.
23. Implement a distributed logging solution for microservices using Loki.
24. Implement database-per-service architecture for microservices.
25. Implement a microservices security policy using Istio.
26. Implement a secure gRPC communication between microservices.
27. Implement a cloud-native microservices architecture using AWS Lambda.
28. Implement an API versioning strategy in a microservices ecosystem.
29. Implement Kubernetes network policies for microservices.
30. Implement a microservices-based fraud detection system.

**10. System Design (High-Level Architecture & Scalability)**

1. Design a URL shortening service like Bitly.
2. Design a social media news feed system.
3. Design a real-time chat application like WhatsApp.
4. Design an e-commerce platform architecture.
5. Design a scalable notification service.
6. Design a distributed caching system.
7. Design a stock trading platform.
8. Design a ride-sharing service like Uber.
9. Design a video streaming platform like YouTube.
10. Design a real-time bidding system like Google Ads.
11. Design a cloud storage system like Google Drive.
12. Design a multi-tenant SaaS architecture.
13. Design an event-driven order processing system.
14. Design a payment gateway system like Stripe.
15. Design a real-time collaboration tool like Google Docs.
16. Design a recommendation system for an e-commerce platform.
17. Design a banking transaction system with fraud detection.
18. Design a hotel booking system like Airbnb.
19. Design a search autocomplete system like Google Search.
20. Design a high-throughput email delivery service.
21. Design a distributed queue system like Apache Kafka.
22. Design a clickstream data analytics platform.
23. Design a decentralized identity management system.
24. Design a feature flagging system for A/B testing.
25. Design a news aggregation service like Google News.
26. Design a traffic monitoring system for smart cities.
27. Design a scalable IoT data ingestion pipeline.
28. Design a cloud-based document processing system.
29. Design an AI-driven customer support chatbot.
30. Design a zero-trust security model for a SaaS platform.

**11. Kubernetes & Cloud (AWS, GCP, Azure)**

1. Deploy a Spring Boot application on Kubernetes.
2. Set up auto-scaling in Kubernetes for a microservice.
3. Implement a Kubernetes-based CI/CD pipeline.
4. Deploy a multi-container application using Kubernetes.
5. Implement Kubernetes service mesh using Istio.
6. Deploy a high-availability PostgreSQL database using Kubernetes.
7. Implement a Kubernetes logging system using Fluentd and Elasticsearch.
8. Implement a Kubernetes monitoring system using Prometheus and Grafana.
9. Implement a Kubernetes secrets management system using HashiCorp Vault.
10. Implement a Kubernetes-based data pipeline.
11. Deploy a microservices architecture in AWS EKS.
12. Implement an AWS Lambda function in Java.
13. Implement a serverless application using AWS Lambda and DynamoDB.
14. Implement an S3 file processing pipeline using AWS Lambda.
15. Deploy a Java-based REST API on AWS Fargate.
16. Set up an AWS API Gateway with authentication using Cognito.
17. Implement a distributed NoSQL database architecture in AWS.
18. Implement a cloud-native application using Google Cloud Functions.
19. Implement a secure multi-region deployment in AWS.
20. Implement a real-time analytics pipeline using AWS Kinesis.
21. Deploy a containerized Java application in Google Kubernetes Engine.
22. Implement an AWS-based fraud detection system.
23. Deploy a Java-based AI/ML model using AWS SageMaker.
24. Implement an end-to-end DevOps CI/CD pipeline using Jenkins and AWS.
25. Implement a disaster recovery plan for a cloud-based system.
26. Implement a FinOps strategy for cost optimization in AWS.
27. Implement a Kubernetes-based machine learning deployment.
28. Deploy an AI-powered recommendation engine in the cloud.
29. Implement an AWS-based blockchain system.
30. Implement a hybrid cloud architecture for an enterprise.