Problem Solving & Data Structures : (60 mins)

We expect you to tackle fundamental problems related to basic data structures,such as Arrays, Stacks, Queues, Linked Lists, Maps etc. Your performance will be assessed based on several criteria:

- **Problem-Solving Skills:** Demonstrate your ability to effectively address and solve these problems.
- **Algorithmic Approach:** Show your proficiency in designing efficient algorithms, supported by appropriate pseudocode.
- **Data Structure Selection**: Justify your choice of data structures and explain how they contribute to the efficiency of your solution.
- **Complexity Analysis:** Provide a thorough understanding of the time and space complexities associated with your chosen algorithms.

Your solutions should reflect a clear grasp of these concepts and their practical applications.
Ability to convert DS&A into code and describe run time of solution and run time analysis.
The platform used during the interview is : CodeSignal

Platform Round : (60 mins)

iOS & Swift

  - Overview of iOS architecture and lifecycle
  - Key components of iOS development (e.g., ViewControllers, Delegates)
- **Swift Fundamentals**
  - Core syntax and language features
  - Advanced Swift concepts (e.g., Generics, Protocol-Oriented Programming)
- **Threading and Memory Management**
  - Concurrency in Swift (e.g., GCD, Operation Queues)
  - Memory management principles (e.g., ARC, memory leaks, retain cycles)
- **UI & Auto Layout**
  - Auto Layout principles and constraints
  - Dynamic UI adjustments and responsive design
  - Use of SwiftUI and its integration with UIKit
- **Security Best Practices**
  - Secure data storage and encryption
  - Authentication and authorization mechanisms
  - Network security (e.g.HTTPS/TLS/TCP/IP secure communication practices)

Feel free to discuss any specific iOS topic or area of interest that you are passionate about during our conversation.

Android

- **Android Architecture**
  - Overview of the Android OS architecture
  - Components and lifecycle management

- **Core Android and UI Framework**
  - Lifecycle and state management
  - Communication between Activities and Fragments
- **Services**
  - Types of services (e.g., Foreground, Background)
  - Service lifecycle and inter-process communication
- **Content Providers**
  - Data sharing between applications
  - Implementing and using Content Providers
- **Broadcast Receivers**
  - Handling and broadcasting system-wide or application-specific intents
- **Database**
  - SQLite database management
  - Room Persistence Library for data storage
- **Networking**
  - Making network requests (e.g., using Retrofit, Volley)
  - Handling network responses and data parsing
- **Threading and Memory Management**
  - Concurrency and threading techniques (e.g., Coroutines)
  - Memory management practices (e.g., avoiding memory leaks, garbage collection)
- **Security**
  - Best practices for securing data and communications
  - Authentication and authorization strategies
  - Securing sensitive information and preventing common vulnerabilities

Feel free to discuss any specific Android topic or area of interest that you are passionate about during our conversation.

Machine Coding : iOS (2 hours) and Android (1.5 hours)

- **Design Patterns**
  - Use of appropriate design patterns
  - Adherence to best practices and proper SOLID principles
  - Code readability and maintainability

- **Class Definitions and Naming Conventions**
  - Proper class definitions
  - Consistent and descriptive naming conventions
  - Clear and logical class structure

- **Extensibility**
  - Code flexibility for future enhancements
  - Modularity and separation of concerns

- **Edge Cases and Exception Handling**
  - Comprehensive handling of edge cases
  - Robust exception handling mechanisms
  - Graceful debugging, error recovery and reporting

Evaluation Metrics

- **Code Style & Modularity (60%)**
  - Overall code style
  - Code organization and modularity
- **Extensibility and Abstraction (20%)**
  - Code extensibility
  - Proper use of abstraction
- **Functionality (20%)**
  - Fulfillment of expected functionality
  - Correctness and performance of implemented features

Design Round : (60 mins) (Recruiter will mention if this round is applicable)

- **Architecture Design**
  - High-level system architecture and component interactions
  - Design patterns and principles (e.g., MVC, MVVM, Clean Architecture)
  - Scalability and maintainability considerations

- **App Design**
    - User experience (UX) and user interface (UI) design
    - Handling Device Fragmentation
    - Integration of system components (e.g., Activities, Fragments, Services)
- **API Design**
    - Designing REST APIs for use cases with appropriate considerations to payload size optimisation, authentication etc.

Exploring Edge Cases

- **Identifying Edge Cases**
    - Anticipating and handling atypical or unexpected scenarios
    - Strategies for testing and validation
- **Robustness and Error Handling**
    - Designing for resilience and fault tolerance
    - Comprehensive exception handling and recovery mechanisms

Choice of Technology

- **Technology Stack Selection**
    - Evaluation criteria for selecting frameworks, libraries, and tools
    - Justification of technology choices based on project requirements and constraints
- **Integration and Compatibility**
    - Ensuring seamless integration of chosen technologies
    - Addressing compatibility issues and dependencies

Hiring Manager Round (60 mins)

Soft Skills

- **Communication**
    - Clarity in conveying ideas and technical concepts
    - Active listening and effective interpersonal interactions
- **Collaboration**
    - Ability to work effectively within a team
    - Openness to feedback and willingness to assist others
- **Adaptability**
    - Flexibility in handling changing requirements or environments
    - Resilience and ability to manage stress or unexpected challenges
- **Time Management**
    - Prioritization of tasks and meeting deadlines
    - Efficient management of workload and project timelines

Core Skills

- **Technical Proficiency**
    - Mastery of essential technologies and tools relevant to the role
    - Demonstrated expertise in fundamental concepts and practices
- **Development Practices**
    - Adherence to best coding practices and standards
    - Proficiency in code review and quality assurance processes
- **Domain Knowledge**
    - Deep understanding of the specific domain or industry of the project
    - Application of relevant knowledge to solve domain-specific problems

Problem Solving

- **Analytical Thinking**
    - Ability to decompose complex problems into manageable components
    - Application of logical reasoning to develop effective solutions
- **Innovative Solutions**
    - Creativity in approaching and resolving issues
    - Use of novel or non-standard techniques to address challenges
- **Decision Making**
    - Evaluation of options and making informed decisions
    - Balancing trade-offs and considering long-term implications

Fitment to the Team

- **Cultural Fit**
    - Alignment with the team's values, mission, and work culture
    - Contribution to a positive and collaborative team environment
- **Role Suitability**
    - Match between individual skills and the requirements of the role
    - Contribution to the team's objectives and project goals
- **Growth Potential**
    - Ability to grow and adapt within the team
    - Willingness to take on new challenges and expand expertise.