C# (Basic to Advanced)

1. **Question**: What is the difference between == and Equals() in C#?
   Answer: == compares reference equality for reference types and value equality for value types, while Equals() can be overridden to provide custom value equality logic.
   Difficulty: Easy

2. **Question**: What is a nullable type in C#?
   Answer: A nullable type is a value type that can also represent null. It is defined using ?, e.g., int? can hold an integer or **null**.
   Difficulty: Easy

3. **Question**: What is a delegate in C#?

   Answer: A type that represents references to methods with a specific parameter list and return type. When you instantiate a delegate, you can associate its instance with any method that matches its signature and return type. You can invoke (or call) the method through the delegate instance.

4. **Question**: Explain the async and await keywords in C#.
   Answer: async modifies a method to indicate it contains asynchronous operations, while await is used to pause execution until the awaited task completes.
   Difficulty: Medium

5. **Question**: What is the difference between an abstract class and an interface in C#?
   Answer: An abstract class can provide method implementations and state, while an interface only defines method signatures without implementations.
   Difficulty: Medium

6. **Question**: How does garbage collection work in C#?
   Answer: Garbage collection automatically frees memory occupied by objects that are no longer referenced. It uses a generational approach to optimize performance.
   Difficulty: Medium

7. **Question**: What are extension methods in C#?
   Answer: Extension methods allow you to add new methods to existing types without modifying their source code. They are defined in static classes and use the this keyword in the first parameter.
   Difficulty: Medium

8. **Question**: Explain the using statement in C#.
   Answer: The using statement ensures that IDisposable objects are disposed of once they are no longer needed, even if an exception occurs.
   Difficulty: Medium

9. **Question**: What is dependency injection in C#?
   Answer: Dependency injection is a design pattern that allows a class to receive its dependencies from an external source rather than creating them internally, promoting loose coupling and easier testing.
   Difficulty: Hard

10. **Question**: How do you implement asynchronous programming in C# using the Task Parallel Library (TPL)?
    Answer: You can use Task.Run() to run code asynchronously, Task.WhenAll() to wait for multiple tasks to complete, and async/await for easier asynchronous programming.
    Difficulty: Hard

11. **Question**: Explain the concept of covariance and contravariance in C#.
    Answer: Covariance allows a method to return a more derived type than specified, while contravariance allows a method to accept parameters of less derived types. This is typically used with delegates and generics.
    Difficulty: Very Hard

Object-Oriented Programming in C#

1. **Question**: What are the four pillars of OOP?
   Answer: The four pillars are encapsulation, inheritance, polymorphism, and abstraction.
   Difficulty: Easy

2. **Question**: Explain encapsulation with an example.
   Answer: Encapsulation is the bundling of data and methods that operate on that data within a single unit (class). Example: Using private fields with public properties for access control.
   Difficulty: Easy

3. **Question**: What is inheritance in C#?
   Answer: Inheritance allows a class to inherit properties and methods from another class, promoting code reuse and a hierarchical relationship.
   Difficulty: Easy

4. **Question**: Define polymorphism.
   Answer: Polymorphism allows methods to do different things based on the object it is acting upon, typically through method overriding or interfaces.
   Difficulty: Medium

5. **Question**: What is the difference between method overloading and method overriding?
   Answer: Method overloading allows multiple methods with the same name but different parameters in the same class, while method overriding replaces a base class method in a derived class.
   Difficulty: Medium

6. **Question**: How do you implement an interface in C#?
   Answer: A class implements an interface by providing implementations for all its members, using the : interfaceName syntax after the class name.
   Difficulty: Medium

7. **Question**: Explain the concept of an abstract class with an example.
   Answer: An abstract class cannot be instantiated and can contain abstract methods (without implementation). Derived classes must implement these methods.
   Difficulty: Medium

8. **Question**: What is a sealed class?
   Answer: A sealed class cannot be inherited, which prevents further derivation and can be used for performance optimization.
   Difficulty: Medium

9. **Question**: How does C# handle multiple inheritance?
   Answer: C# does not support multiple inheritance for classes, but it allows a class to implement multiple interfaces.
   Difficulty: Hard

10. **Question**: What are the differences between an abstract class and an interface in C#?
    Answer: An abstract class can contain implementation, fields, and constructors, while an interface can only define method signatures and properties.
    Difficulty: Hard

Design Patterns in C#

1. **Question**: What is a design pattern?
   Answer: A design pattern is a reusable solution to a common software design problem that occurs in a given context.
   Difficulty: Easy

2. **Question**: Explain the Singleton pattern.
   Answer: The Singleton pattern ensures a class has only one instance and provides a global access point to that instance.
   Difficulty: Easy

3. **Question**: What is the Factory Method pattern?
   Answer: The Factory Method pattern defines an interface for creating objects but lets subclasses alter the type of objects that will be created.
   Difficulty: Medium

4. **Question**: Describe the Observer pattern.
   Answer: Observer pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
   Difficulty: Medium

5. **Question**: Explain the Strategy pattern.
   Answer: The Strategy pattern defines a family of algorithms, encapsulates each one, and makes them interchangeable. This allows the algorithm to vary independently from the clients that use it.
   Difficulty: Medium

6. **Question**: What is the difference between the Adapter and the Facade pattern?
   Answer: The Adapter pattern allows incompatible interfaces to work together, while the Facade pattern provides a simplified interface to a complex subsystem.
   Difficulty: Medium

7. **Question**: Explain the Decorator pattern.
   Answer: The Decorator pattern allows behavior to be added to individual objects, either

statically or dynamically, without affecting the behavior of other objects from the same class.
Difficulty: Hard

SOLID Principles

1. **Question**: What does SOLID stand for?
   Answer: SOLID stands for Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion principles.
   Difficulty: Easy
2. **Question**: Explain the Single Responsibility Principle (SRP).
   Answer: SRP states that a class should have only one reason to change, meaning it should only have one job or responsibility.
   Difficulty: Easy
3. **Question**: What is the Open/Closed Principle (OCP)?
   Answer: OCP states that software entities (classes, modules, functions) should be open for extension but closed for modification, allowing new functionality without altering existing code.
   Difficulty: Medium
4. **Question**: Describe the Liskov Substitution Principle (LSP).
   Answer: LSP states that objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.
   Difficulty: Medium
5. **Question**: What is the Interface Segregation Principle (ISP)?
   Answer: ISP states that no client should be forced to depend on methods it does not use, advocating for smaller and more specific interfaces.
   Difficulty: Medium
6. **Question**: Explain the Dependency Inversion Principle (DIP).
   Answer: DIP states that high-level modules should not depend on low-level modules but should depend on abstractions. Both should depend on abstractions, promoting loose coupling.
   Difficulty: Medium
7. **Question**: How do SOLID principles improve code quality?
   Answer: SOLID principles enhance maintainability, scalability, and readability, reduce coupling, and help prevent issues that arise from modifying existing code.
   Difficulty: Hard
8. **Question**: Give an example of applying the Open/Closed Principle in C#.
   Answer: You can use interfaces or abstract classes to define behavior and create concrete implementations for new functionality without changing existing code.
   Difficulty: Hard
9. **Question**: How can the Dependency Inversion Principle be implemented in C#?
   Answer: You can use dependency injection frameworks or manual dependency injection to pass dependencies into classes rather than creating them internally.
   Difficulty: Hard

10. **Question**: What are some common violations of SOLID principles?
    Answer: Common violations include having classes with multiple responsibilities, tightly coupled code, large interfaces, and methods that handle multiple concerns.
    Difficulty: Very Hard

Data Structures in C#

1. **What is an array in C# and how do you declare it?**

   Difficulty: Easy
   Answer: An array in C# is a collection of variables of the same type stored in contiguous memory locations. It is declared using square brackets.

2. **How does a List<T> differ from an array in C#?**

   Difficulty: Easy
   Answer: A List<T> is a dynamic array that automatically resizes itself as elements are added or removed. In contrast, a standard array has a fixed size once it is initialized.

3. **What is a Dictionary<TKey, TValue> in C# and when would you use it?**

   Difficulty: Easy
   Answer: A Dictionary<TKey, TValue> is a collection of key-value pairs. It is used when you need to quickly retrieve data using a unique key. The keys must be unique.

4. **What is a stack and how can you implement it in C#?**
   Difficulty: Easy
   Answer: A stack is a Last-In-First-Out (LIFO) data structure. C# provides a Stack<T> class to implement a stack.

5. **What is a queue and how can you implement it in C#?**
   Difficulty: Medium
   Answer: A queue is a First-In-First-Out (FIFO) data structure. C# provides a Queue<T> class for this.

6. **How do you implement a linked list in C#?**
   Difficulty: Medium
   Answer: C# provides a LinkedList<T> class for a doubly linked list, where each node has pointers to both the next and previous node.

7. **What is the difference between a HashSet<T> and a List<T> in C#?**

   Difficulty: Medium
   Answer: A HashSet<T> stores unique elements and is optimized for fast lookups (constant time O(1)), while a List<T> can store duplicates and generally has slower lookup times (linear O(n)).

8. How would you reverse a linked list in C#?

Difficulty: Medium
Answer: To reverse a singly linked list, you need to iterate through the list and reverse the pointers of each node. Here's a simple implementation:

9. What is the time complexity of searching in a balanced binary search tree (BST)?

Difficulty: Hard
Answer: The time complexity of searching in a balanced BST is O(log n) because the tree is structured in such a way that each comparison reduces the search space by half.

10. How do you detect a cycle in a linked list in C#?

Difficulty: Hard
Answer: One common approach to detect a cycle in a linked list is to use Floyd’s Tortoise and Hare algorithm. You maintain two pointers (slow and fast). The slow pointer moves one step, and the fast pointer moves two steps. If they meet, there is a cycle.

Unit Testing in C#

1. **Question**: What is unit testing?
   Answer: Unit testing is the practice of testing individual units of code (usually functions or methods) to ensure they work as expected.
   Difficulty: Easy
2. **Question**: What are some common unit testing frameworks in C#?
   Answer: Common frameworks include NUnit, xUnit, and MSTest.
   Difficulty: Easy
3. **Question**: How do you create a unit test in NUnit?
   Answer: You create a test class, annotate methods with [Test], and use assertions to check expected outcomes.
   Difficulty: Easy
4. **Question**: What is the purpose of mocking in unit testing?
   Answer: Mocking is used to create dummy implementations of dependencies to isolate the unit under test and verify interactions without invoking actual behavior.
   Difficulty: Medium
5. **Question**: Explain the Arrange-Act-Assert (AAA) pattern.
   Answer: The AAA pattern structures unit tests by first arranging the necessary setup, then acting on the object under test, and finally asserting the expected outcomes.
   Difficulty: Medium
6. **Question**: What is Test-Driven Development (TDD)?
   Answer: TDD is a software development approach where tests are written before the code itself, promoting better design and ensuring that code meets requirements.
   Difficulty: Medium

7. **Question**: How do you test asynchronous methods in C#?
   Answer: You can use async Task in your test method and await the asynchronous calls to test their outcomes.
   Difficulty: Medium
8. **Question**: What is the difference between integration tests and unit tests?
   Answer: Unit tests focus on individual components in isolation, while integration tests check how multiple components work together.
   Difficulty: Medium
9. **Question**: What are some common practices for writing good unit tests?
   Answer: Practices include writing tests that are independent, repeatable, descriptive, covering edge cases, and maintaining a consistent naming convention.
   Difficulty: Hard
10. **Question**: How do you handle exceptions in unit tests?
    Answer: You can use assertions to check for expected exceptions, using constructs like Assert.Throws<ExceptionType>(() => MethodCall()) in NUnit.
    Difficulty: Hard

Web API with .NET

1. **Question:** What is a Web API?

   Answer: A Web API is an interface that allows different applications to communicate with each other over the internet using HTTP. It enables interaction between clients and servers.

   Difficulty: Easy

2. **Question: What is REST?**

   Answer: REST (Representational State Transfer) is an architectural style for designing networked applications. It relies on stateless, client-server communication and uses standard HTTP methods like GET, POST, PUT, and DELETE.

   Difficulty: Easy

3. **Question: What is JSON and why is it used in APIs?**

   Answer: JSON (JavaScript Object Notation) is a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate. It is commonly used in APIs because of its simplicity and compatibility with JavaScript.

   Difficulty: Easy

4. **Question: What are the differences between SOAP and REST APIs?**

   Answer: SOAP (Simple Object Access Protocol) is a protocol that relies on XML and has strict standards, while REST is an architectural style that can use various formats (JSON, XML, etc.) and is more flexible and easier to use. REST is generally considered lightweight compared to SOAP.

   Difficulty: Medium

5. **Question: How do you handle versioning in a Web API?**

   Answer: Versioning can be handled using several strategies, such as URL versioning (e.g., /v1/resource), query parameter versioning (e.g., /resource?version=1), and header versioning (e.g., using a custom header). The choice depends on the project's needs and the expected frequency of changes.

   Difficulty: Medium

6. **Question: What are status codes in HTTP, and can you name a few?**

   Answer: Status codes are three-digit numbers returned by a server to indicate the result of a client's request. Common ones include 200 (OK), 201 (Created), 400 (Bad Request), 401 (Unauthorized), and 404 (Not Found).

   Difficulty: Medium

7. **Question: How can you ensure idempotency in a Web API?**

   Answer: Idempotency means that multiple identical requests should have the same effect as a single request. This can be achieved by using HTTP methods appropriately (e.g., GET, PUT) and designing resource operations so that they do not cause unintended side effects when repeated.

   Difficulty: Hard

8. **Question: How do you implement pagination in a Web API?**

   Answer: Pagination can be implemented using query parameters like limit and offset or page and size. This helps control the amount of data returned in a single response and improve performance.

   Difficulty: Hard

9. **Question: What is CORS, and why is it important for Web APIs?**

   Answer: CORS (Cross-Origin Resource Sharing) is a security feature that allows or restricts resources requested from a different domain. It is important because it protects APIs from malicious requests coming from unauthorized domains.

   Difficulty: Hard

10. **Question: How can you test a Web API?**

    Answer: Testing can be performed using tools like Postman, Swagger, or automated testing frameworks (e.g., NUnit, Jest). Tests should include unit tests, integration tests, and end-to-end tests to ensure all components function as expected.

    Difficulty: Hard

11. **Question: What is middleware in the context of web applications?**

    Answer: Middleware is software that acts as a bridge between different applications or services, processing requests and responses in a web application, often in a sequential order.

Difficulty: Easy

12. **Question: Can you name a few common uses for middleware?**

    Answer: Common uses include authentication, logging, error handling, request parsing, and serving static files.

    Difficulty: Easy

13. **Question: What is the difference between application-level and router-level middleware in Express?**

    Answer: Application-level middleware is bound to an instance of the app and applies to all routes. Router-level middleware applies only to specific routes defined in a router, allowing more granular control.

    Difficulty: Medium

14. **Question: What are error-handling middleware, and how do they differ from regular middleware?**

    Answer: Error-handling middleware is specifically designed to handle errors that occur during request processing. It takes four arguments: err, req, res, and next, allowing it to manage errors and send appropriate responses.

    Difficulty: Medium

15. **Question: How does middleware affect the request-response lifecycle in a web application?**

    Answer: Middleware functions can intercept and modify requests and responses, perform actions before passing control to the next function, or terminate the request-response cycle. This can include tasks like authentication, validation, and error handling.

    Difficulty: Hard

16. **Question: What are some performance implications of using middleware?**

    Answer: Using too many middleware functions can lead to performance bottlenecks, especially if they perform heavy computations or blocking operations. It's essential to use efficient middleware and optimize the request-response cycle to maintain performance.

    Difficulty: Hard

17. **Question: What is API security?**

    Answer: API security refers to the measures taken to protect APIs from unauthorized access, misuse, and abuse, ensuring the integrity, confidentiality, and availability of data and services.

    Difficulty: Easy

18. **Question: What are common authentication methods for APIs?**

    Answer: Common authentication methods include Basic Authentication, API keys, OAuth, and JWT (JSON Web Tokens). Each has its use cases and security implications.

Difficulty: Easy

19. **Question: What is HTTPS, and why is it important for APIs?**

    Answer: HTTPS (Hypertext Transfer Protocol Secure) is an extension of HTTP that uses SSL/TLS to encrypt data exchanged between clients and servers. It is important for protecting sensitive data and preventing man-in-the-middle attacks.

    Difficulty: Easy

20. **Question: What is routing in web applications?**

    Answer: Routing is the process of defining how an application responds to client requests for specific endpoints, typically by associating URLs with corresponding request handlers.

    Difficulty: Easy

21. **Question**: What is a RESTful API?

    Answer: A RESTful API is an architectural style that uses standard HTTP methods (GET, POST, PUT, DELETE) to operate on resources identified by URLs.
    Difficulty: Easy

22. **Question**: Explain the role of routing in ASP.NET Web API.
    Answer: Routing is responsible for mapping incoming HTTP requests to the appropriate controller and action method based on the URL pattern.
    Difficulty: Easy

23. **Question**: How do you handle errors in a Web API?
    Answer: You can use exception filters or middleware to catch and log exceptions, returning standardized error responses.
    Difficulty: Medium

24. **Question**: What is model binding in ASP.NET Web API?
    Answer: Model binding is the process of converting HTTP request data (query string, form data, JSON) into .NET objects that can be used in action methods.
    Difficulty: Medium

25. **Question**: How do you secure a Web API?
    Answer: You can secure a Web API using authentication (e.g., OAuth, JWT) and authorization strategies, such as role-based access control.
    Difficulty: Medium

26. **Question**: What is Swagger, and how is it used in ASP.NET Web API?
    Answer: Swagger is a tool that generates interactive API documentation. In ASP.NET Web API, you can use it to document endpoints, request/response types, and test the API directly.
    Difficulty: Medium

27. **Question**: How do you implement versioning in a Web API?
    Answer: You can implement versioning by using URL path segments, query strings, or custom

headers to differentiate between versions of the API.
Difficulty: Hard

28. **Question**: Explain the difference between HTTP status codes 200, 201, 400, and 404.
Answer: 200 OK indicates success, 201 Created indicates a successful resource creation, 400 Bad Request indicates client-side error due to invalid request, and 404 Not Found indicates that the requested resource was not found.
Difficulty: Hard

29. **Question**: What is CORS, and how do you enable it in ASP.NET Web API?
Answer: CORS (Cross-Origin Resource Sharing) allows restricted resources on a web page to be requested from another domain. You can enable it by adding the Microsoft.AspNet.WebApi.Cors package and configuring it in WebApiConfig.
Difficulty: Hard

30. **Question**: How do you handle pagination in a Web API?
Answer: Pagination can be handled by implementing query parameters (e.g., page and pageSize) to control the number of records returned in the response.
Difficulty: Very Hard

Entity Framework Core

1. **Question**: What is Entity Framework Core?
Answer: Entity Framework Core is an object-relational mapper (ORM) for .NET that allows developers to work with databases using .NET objects, eliminating the need for most data-access code.
Difficulty: Easy

2. **Question**: How do you define a model in Entity Framework Core?
Answer: A model is defined as a class representing a database table. Properties of the class represent columns in the table, and data annotations can be used for configuration.
Difficulty: Easy

3. **Question**: Explain the concept of migrations in EF Core.
Answer: Migrations are a way to incrementally update the database schema to keep it in sync with the application’s data model, allowing you to apply and revert changes.
Difficulty: Medium

4. **Question**: What are DbSets in EF Core?
Answer: A DbSet represents a collection of entities that can be queried from the database. Each DbSet corresponds to a table in the database.
Difficulty: Medium

5. **Question**: How do you configure relationships between entities in EF Core?
Answer: Relationships can be configured using navigation properties and the Fluent API or data annotations, specifying one-to-one, one-to-many, or many-to-many relationships.
Difficulty: Medium

6. **Question**: Explain lazy loading in EF Core.
Answer: Lazy loading is a feature that loads related data only when it is accessed, which can

help improve performance and reduce initial data load times.
Difficulty: Medium

7. **Question**: How do you perform a query using LINQ in EF Core?
Answer: You can use LINQ to query the DbSet, applying methods like Where(), Select(), and ToList() to filter, transform, and execute the query against the database.
Difficulty: Medium

8. **Question**: What is the difference between Add() and Attach() methods in EF Core?
Answer: Add() is used to add a new entity to the context and marks it as added, while Attach() is used for existing entities to inform the context of their existence without changing their state.
Difficulty: Hard

9. **Question**: Explain the concept of tracking vs. no-tracking queries in EF Core.
Answer: Tracking queries monitor changes to entities, while no-tracking queries do not track changes, which can improve performance for read-only operations.
Difficulty: Hard

10. **Question**: How do you implement global filters in EF Core?
Answer: Global filters can be implemented using the OnModelCreating method in the DbContext, allowing you to apply filtering logic globally to specific entity types.
Difficulty: Very Hard

Entity Framework Lazy Loading

1. What is Lazy Loading in Entity Framework?

Answer: Lazy loading is a feature in Entity Framework that delays the loading of related data until it is explicitly accessed. This can improve performance by reducing the amount of data loaded initially.

Difficulty: Easy

2. How do you enable Lazy Loading in Entity Framework?

Answer: Lazy loading is enabled by default in Entity Framework when using virtual navigation properties in your entities. You can also configure it in the context using Configuration.LazyLoadingEnabled = true.

Difficulty: Easy

3. What are the potential downsides of using Lazy Loading?

Answer: Lazy loading can lead to performance issues due to multiple database queries being executed for each navigation property accessed. This can result in the "N+1 query problem."

Difficulty: Medium

4. How can you disable Lazy Loading in Entity Framework?

Answer: You can disable lazy loading by setting Configuration.LazyLoadingEnabled = false in your DbContext class.

Difficulty: Medium

5. What is the difference between Eager Loading and Lazy Loading?

Answer: Eager loading retrieves related entities as part of the initial query using the Include method, while lazy loading retrieves related entities on-demand when they are accessed for the first time.

Difficulty: Medium

6. When would you prefer Eager Loading over Lazy Loading?

Answer: You would prefer eager loading when you know you will need the related data immediately, as it can reduce the number of queries and improve performance in such scenarios.

Difficulty: Medium

7. How can you implement Lazy Loading with a specific condition?

Answer: You can use proxy classes for entities and configure your navigation properties with virtual. You can also manage conditions in your queries to control when to load data.

Difficulty: Medium

8. What are Proxy Objects in Entity Framework?

Answer: Proxy objects are dynamically generated subclasses that override virtual navigation properties to implement lazy loading. They allow Entity Framework to automatically load related entities when accessed.

Difficulty: Hard

9. How do you handle Lazy Loading in a disconnected scenario?

Answer: In disconnected scenarios, you can use DTOs (Data Transfer Objects) to explicitly load related data before detaching from the context or use explicit loading instead of lazy loading.

Difficulty: Hard

10. How do you troubleshoot performance issues related to Lazy Loading?

Answer: You can analyze the generated SQL queries, use profiling tools, and optimize your data access patterns. Consider using eager loading or explicit loading where appropriate to avoid excessive queries.

Difficulty: Hard

Microservices in C#

1. **What are microservices?**

Answer: Microservices are an architectural style that structures an application as a collection of small, independent services, each running in its own process and communicating with lightweight mechanisms, often HTTP APIs. They are designed to be loosely coupled, enabling teams to develop, deploy, and scale each service independently.

Difficulty: Easy

2. **What are the advantages of using microservices?**

Answer: Advantages include improved scalability, better fault isolation, technology diversity, faster time to market, easier maintenance, and the ability to deploy independently.

Difficulty: Easy

3. **How do you manage data consistency in microservices?**

Answer: Data consistency can be managed through eventual consistency, using distributed transactions (though they are complex), or implementing a saga pattern that coordinates transactions across multiple services.

Difficulty: Medium

4. **What is a service discovery in microservices?**

Answer: Service discovery is the process of automatically detecting devices and services on a network. In microservices, it allows services to find and communicate with each other without hardcoding the network locations.

Difficulty: Medium

5. **How do you handle inter-service communication in microservices?**

Answer: Inter-service communication can be handled using synchronous protocols like REST or gRPC, or asynchronous messaging queues such as RabbitMQ, Apache Kafka, or Azure Service Bus.

Difficulty: Medium

6. **What is the role of API Gateway in microservices architecture?**

Answer: An API Gateway acts as a single entry point for all client requests. It routes requests to the appropriate microservices, handles cross-cutting concerns like authentication, logging, and rate limiting, and can aggregate responses from multiple services.

Difficulty: Medium

7. **How do you implement security in microservices?**

Answer: Security can be implemented through OAuth2, JWT for authentication, and SSL/TLS for secure communication. Each service should also validate tokens and have role-based access control.

Difficulty: Medium

8. **What is the Circuit Breaker pattern in microservices?**

Answer: The Circuit Breaker pattern is used to prevent cascading failures in microservices by detecting failures and temporarily blocking requests to the failing service. It allows for fallback strategies and recovery when the service is healthy again.

Difficulty: Medium

9. **How do you test microservices?**

Answer: Microservices can be tested using unit tests, integration tests, and contract tests. Tools like Postman or Swagger can be used for API testing, and frameworks like JUnit or NUnit for unit tests.

Difficulty: Hard

10. **What is the role of Docker in microservices?**

Answer: Docker allows developers to package applications and their dependencies into containers, ensuring that services run consistently across different environments. It simplifies deployment and scaling of microservices.

Difficulty: Hard

Monolithic Applications in C#

1. **What is a monolithic application?**

Answer: A monolithic application is a single, unified software application where all components are interconnected and interdependent, typically running as a single service.

Difficulty: Easy

2. What are the advantages of a monolithic architecture?

Answer: Advantages include simplicity in development and deployment, easier debugging, and better performance due to fewer inter-process communications.

Difficulty: Easy

3. What are the disadvantages of monolithic applications?

Answer: Disadvantages include limited scalability, difficulty in adopting new technologies, increased risk of failures affecting the entire system, and challenges in team collaboration as the codebase grows.

Difficulty: Medium

4. How do you improve the performance of a monolithic application?

Answer: Performance can be improved through code optimization, caching strategies, database indexing, and using asynchronous programming to prevent blocking calls.

Difficulty: Medium

5. What is the process of breaking down a monolith into microservices?

Answer: The process involves identifying bounded contexts, refactoring functionalities into services, establishing inter-service communication, and gradually migrating users to the new services while maintaining the existing monolith until complete.

Difficulty: Medium

6. How do you handle dependencies in a monolithic application?

Answer: Dependencies can be managed using dependency injection, keeping the application modular, and ensuring that components are loosely coupled, enabling easier updates and maintenance.

Difficulty: Medium

7. What strategies can you use to deploy a monolithic application?

Answer: Strategies include rolling deployments, blue-green deployments, and canary releases, which help in minimizing downtime and risks during deployment.

Difficulty: Medium

8. How do you ensure maintainability in a monolithic application?

Answer: Maintainability can be ensured through proper code organization, consistent coding standards, comprehensive documentation, and regular code reviews.

Difficulty: Hard

9. What role does logging play in monolithic applications?

Answer: Logging is crucial for monitoring application behavior, diagnosing issues, and tracking performance metrics. Implementing a centralized logging solution can simplify the management of logs from various parts of the application.

Difficulty: Hard

10. How do you handle scaling in a monolithic application?

Answer: Scaling can be achieved through vertical scaling (increasing resources on the server) and horizontal scaling (replicating the entire application across multiple servers), although the latter can be more challenging due to the tightly coupled nature of monoliths.

Difficulty: Hard

Docker and Kubernetes

1. **Question**: What is Docker?
   Answer: Docker is a platform for developing, shipping, and running applications in containers, enabling isolation and consistency across different environments.
   Difficulty: Easy
2. **Question**: What is a Docker container?
   Answer: A Docker container is a lightweight, standalone executable package that includes everything needed to run a piece of software, including code, runtime, libraries, and system tools.
   Difficulty: Easy
3. **Question**: How do you create a Docker image?
   Answer: A Docker image can be created using a Dockerfile, which contains instructions on how to build the image, and the docker build command is used to build it.
   Difficulty: Medium
4. **Question**: What is a Docker volume?
   Answer: A Docker volume is a persistent storage mechanism that allows data to be stored outside the container, ensuring data persistence even when containers are stopped or removed.
   Difficulty: Medium
5. **Question**: Explain the concept of microservices in the context of Docker.
   Answer: Microservices architecture involves developing applications as a suite of small, independent services, each running in its own container, allowing for scalability and isolation of concerns.
   Difficulty: Medium
6. **Question**: What is Kubernetes?
   Answer: Kubernetes is an open-source container orchestration platform that automates deploying, scaling, and managing containerized applications across clusters of hosts.
   Difficulty: Medium
7. **Question**: How do you deploy a containerized application using Kubernetes?
   Answer: You can deploy a containerized application using a Kubernetes deployment manifest that describes the desired state, and then applying it using kubectl apply.
   Difficulty: Medium
8. **Question**: What are Kubernetes pods?
   Answer: Pods are the smallest deployable units in Kubernetes, which can contain one or more containers that share the same network namespace and storage.
   Difficulty: Medium
9. **Question**: Explain the role of services in Kubernetes.
   Answer: Services provide a stable endpoint for accessing a set of pods, enabling load balancing and service discovery within the cluster.
   Difficulty: Hard

10. **Question**: What is the difference between a StatefulSet and a Deployment in Kubernetes?
Answer: A StatefulSet is used for managing stateful applications that require persistent storage and stable network identities, while a Deployment is used for stateless applications.
Difficulty: Very Hard

HTML

1. **Question**: What does HTML stand for?
Answer: HTML stands for Hypertext Markup Language.
Difficulty: Easy

2. **Question**: What is the purpose of the <head> tag in HTML?
Answer: The <head> tag contains meta-information about the document, such as title, links to stylesheets, and scripts, but does not display content directly.
Difficulty: Easy

3. **Question**: How do you create a hyperlink in HTML?
Answer: A hyperlink is created using the <a> tag, with the href attribute specifying the URL, e.g., <a href="http://example.com">Link</a>.
Difficulty: Easy

4. **Question**: What is the difference between <div> and <span> tags?
Answer: <div> is a block-level element used for grouping content, while <span> is an inline element used for styling parts of text.
Difficulty: Easy

5. **Question**: How do you create an ordered list in HTML?
Answer: An ordered list can be created using the <ol> tag, with each item wrapped in an <li> tag, e.g., <ol><li>Item 1</li></ol>.
Difficulty: Medium

6. **Question**: What are semantic elements in HTML?
Answer: Semantic elements clearly describe their meaning to both the browser and the developer, such as <article>, <section>, <header>, and <footer>.
Difficulty: Medium

7. **Question**: What is the purpose of the <meta> tag in HTML?
Answer: The <meta> tag provides metadata about the HTML document, such as character set, author, and viewport settings for responsive design.
Difficulty: Medium

8. **Question**: How can you embed an image in HTML?
Answer: An image can be embedded using the <img> tag with the src attribute pointing to the image URL, e.g., <img src="image.jpg" alt="Description">.
Difficulty: Medium

9. **Question**: What is the difference between id and class attributes in HTML?
Answer: The id attribute is unique within a document and can be used to target a single element, while the class attribute can be reused for multiple elements for styling purposes.
Difficulty: Hard

10. **Question**: How do you create a form in HTML?
    Answer: A form can be created using the <form> tag, which can contain various input elements like <input>, <select>, and <textarea> to gather user input.
    Difficulty: Hard

CSS

1. **Question**: What does CSS stand for?
   Answer: CSS stands for Cascading Style Sheets.
   Difficulty: Easy
2. **Question**: What is the purpose of CSS?
   Answer: CSS is used to style and layout web pages, including colors, fonts, spacing, and positioning of elements.
   Difficulty: Easy
3. **Question**: How do you link a CSS file to an HTML document?
   Answer: A CSS file can be linked using the <link> tag in the <head> section of the HTML document, e.g., <link rel="stylesheet" href="styles.css">.
   Difficulty: Easy
4. **Question**: What is the difference between class selectors and ID selectors in CSS?
   Answer: Class selectors are denoted with a dot (.) and can be applied to multiple elements, while ID selectors are denoted with a hash (#) and must be unique within the document.
   Difficulty: Medium
5. **Question**: Explain the box model in CSS.
   Answer: The box model describes the layout of elements, including margins, borders, padding, and the actual content area.
   Difficulty: Medium
6. **Question**: What are CSS preprocessors?
   Answer: CSS preprocessors, like SASS or LESS, extend CSS with features like variables, nested rules, and mixins, improving maintainability and organization.
   Difficulty: Medium
7. **Question**: How can you center a block element horizontally in CSS?
   Answer: You can center a block element by setting its margin to auto and specifying a width, e.g., width: 50%; margin: 0 auto;.
   Difficulty: Medium
8. **Question**: What are CSS Flexbox and Grid?
   Answer: Flexbox is a layout model that allows for responsive design using one-dimensional layouts, while Grid is a two-dimensional layout system that enables complex designs.
   Difficulty: Medium
9. **Question**: Explain the concept of specificity in CSS.
   Answer: Specificity determines which CSS rule is applied when multiple rules match the same element, calculated based on the types of selectors used (inline, ID, class, etc.).
   Difficulty: Hard

10. **Question**: What is responsive design, and how can it be achieved with CSS?
    Answer: Responsive design ensures web pages adapt to different screen sizes. It can be achieved using fluid grids, flexible images, and media queries to apply different styles based on the viewport size.
    Difficulty: Very Hard

JavaScript

1. **Question**: What is JavaScript?
   Answer: JavaScript is a high-level, dynamic programming language commonly used for adding interactivity and functionality to web pages.
   Difficulty: Easy
2. **Question**: What are variables in JavaScript?
   Answer: Variables are used to store data values and can be declared using var, let, or const, each with different scoping rules.
   Difficulty: Easy
3. **Question**: What is the difference between == and === in JavaScript?
   Answer: == compares values for equality with type coercion, while === compares values for both equality and type, ensuring strict equality.
   Difficulty: Medium
4. **Question**: What are JavaScript functions?
   Answer: Functions are reusable blocks of code that perform specific tasks, defined using the function keyword or as arrow functions.
   Difficulty: Medium
5. **Question**: What is an object in JavaScript?
   Answer: An object is a collection of key-value pairs, where keys are strings (or Symbols) and values can be any data type, including other objects.
   Difficulty: Medium
6. **Question**: Explain the concept of closures in JavaScript.
   Answer: Closures are functions that retain access to their outer scope variables even after the outer function has finished executing, enabling data privacy.
   Difficulty: Hard
7. **Question**: What is the Document Object Model (DOM)?
   Answer: The DOM is a programming interface that represents the structure of an HTML document, allowing JavaScript to manipulate the content, structure, and styles of web pages.
   Difficulty: Medium
8. **Question**: How can you create a promise in JavaScript?
   Answer: A promise can be created using the Promise constructor, which takes a function with resolve and reject callbacks to handle asynchronous operations.
   Difficulty: Medium
9. **Question**: What are ES6 modules?
   Answer: ES6 modules allow developers to encapsulate code in separate files and use import

and export statements to share functionalities between files.
Difficulty: Hard

10. **Question**: Explain the concept of asynchronous programming in JavaScript.
Answer: Asynchronous programming allows tasks to run in the background without blocking the main thread, enabling non-blocking operations through callbacks, promises, or async/await syntax.
Difficulty: Very Hard

TypeScript

1. **Question**: What is TypeScript?
Answer: TypeScript is a superset of JavaScript that adds static typing and other features, enabling developers to catch errors at compile-time instead of runtime.
Difficulty: Easy

2. **Question**: How do you declare a variable in TypeScript?
Answer: Variables can be declared using let, const, or var, with optional type annotations, e.g., let name: string = "John";.
Difficulty: Easy

3. **Question**: What are interfaces in TypeScript?
Answer: Interfaces define the shape of objects, allowing for strong type-checking and ensuring that objects conform to a specific structure.
Difficulty: Medium

4. **Question**: What is the purpose of the any type in TypeScript?
Answer: The any type allows variables to hold values of any type, bypassing type-checking and providing flexibility, but should be used sparingly to maintain type safety.
Difficulty: Medium

5. **Question**: How do you create a function with optional parameters in TypeScript?
Answer: Optional parameters can be declared by appending a question mark ? to the parameter name, e.g., function greet(name?: string) {}.
Difficulty: Medium

6. **Question**: What are generics in TypeScript?
Answer: Generics allow functions and classes to operate on types specified at the time of use, enabling code reusability and flexibility while maintaining type safety.
Difficulty: Hard

7. **Question**: How do you define a tuple in TypeScript?
Answer: A tuple is defined using square brackets with specific types for each element, e.g., let point: [number, number] = [10, 20];.
Difficulty: Medium

8. **Question**: What is a union type in TypeScript?
Answer: A union type allows a variable to hold values of multiple types, defined using the pipe (|) symbol, e.g., let value: string | number;.
Difficulty: Medium

9. **Question**: How do you create a class in TypeScript?
   Answer: A class is created using the class keyword, with optional constructors, methods, and access modifiers, e.g., class Person { constructor(public name: string) {} }.
   Difficulty: Medium

10. **Question**: What is the difference between interface and type in TypeScript?
    Answer: Both are used to define types, but interface can be merged and extends other interfaces, while type is more flexible and can represent primitive types, unions, and intersections.
    Difficulty: Hard

Angular

1. **Question:** What is Angular, and how does it differ from AngularJS?
   Answer: Angular is a platform for building mobile and desktop web applications. It is a complete rewrite from the same team that built AngularJS, which was based on JavaScript. Angular uses TypeScript and has a component-based architecture, improved performance, and better dependency injection.
   Difficulty: Easy

2. **Question:** What are components in Angular?
   Answer: Components are the basic building blocks of Angular applications. They control a part of the user interface (UI) and consist of an HTML template, a TypeScript class, and CSS styles.
   Difficulty: Easy

3. **Question:** What is a service in Angular?
   Answer: A service in Angular is a reusable piece of code that can be shared across components. Services typically contain business logic or data access logic and are instantiated only once during the application lifetime.
   Difficulty: Easy

4. **Question:** Explain dependency injection in Angular.
   Answer: Dependency Injection (DI) is a design pattern used to implement IoC (Inversion of Control). It allows a class to receive its dependencies from external sources rather than creating them itself. Angular's DI framework provides services to components, making code modular and easier to test.
   Difficulty: Medium

5. **Question:** What are directives in Angular?
   Answer: Directives are classes that can modify the behavior or appearance of elements in the DOM. Angular has three types of directives: components, structural directives (e.g., *ngIf, *ngFor), and attribute directives.
   Difficulty: Medium

6. **Question:** How do you handle forms in Angular?
   Answer: Angular provides two approaches to handle forms: Template-driven forms and Reactive forms. Template-driven forms use Angular directives in the HTML template, while Reactive forms use reactive programming techniques and require the use of FormGroup and

FormControl.
Difficulty: Medium

7. **Question:** What is RxJS, and how is it used in Angular?
Answer: RxJS (Reactive Extensions for JavaScript) is a library for reactive programming using Observables. In Angular, RxJS is used to handle asynchronous data streams, such as HTTP requests and event handling.
Difficulty: Medium

8. **Question:** What is the Angular CLI, and why is it useful?
Answer: The Angular CLI (Command Line Interface) is a powerful tool that allows developers to create, manage, and maintain Angular applications. It automates tasks like generating components, services, and modules, and it helps with building and deploying applications.
Difficulty: Medium

9. **Question:** How can you optimize the performance of an Angular application?
Answer: Performance optimization strategies include using the OnPush change detection strategy, lazy loading modules, minimizing bundle sizes with tree shaking, and using trackBy with *ngFor to improve rendering performance.
Difficulty: Hard

10. **Question:** Explain the concept of change detection in Angular.
Answer: Change detection is the process of checking the state of the application and updating the UI when changes occur. Angular uses a mechanism called Zones to track changes and a change detection strategy (Default or OnPush) to determine how to check for changes efficiently.
Difficulty: Hard

Angular Lazy Loading

1. **What is Lazy Loading in Angular?**

   Answer: Lazy loading is a technique in Angular that allows you to load feature modules only when they are needed, which can improve the initial load time of the application.

   Difficulty: Easy

2. **How do you implement Lazy Loading in Angular?**

   Answer: You can implement lazy loading in Angular by configuring the route for the module using the loadChildren property in your routing module.

   Difficulty: Easy

3. **What is the purpose of loadChildren in Angular routing?**

   Answer: The loadChildren property in Angular routing is used to specify the module to be loaded lazily. It can be set to a string representing the module path or a dynamic import.

   Difficulty: Easy

4. **How do you create a lazy-loaded module in Angular?**

Answer: You can create a lazy-loaded module by generating a new module using Angular CLI and configuring its routing to be loaded lazily in the main application routing.

Difficulty: Medium

5. What is the difference between eager loading and lazy loading in Angular?

Answer: Eager loading loads all modules upfront when the application starts, while lazy loading loads modules on demand as the user navigates to specific routes.

Difficulty: Medium

6. Can you lazy load a component in Angular? If so, how?

Answer: Yes, you can lazy load components using the ng-container directive or by using Angular's dynamic component loading features. However, typically, lazy loading is used with modules.

Difficulty: Medium

7. What are some best practices for implementing Lazy Loading in Angular?

Answer: Best practices include keeping feature modules small, using shared modules effectively, and ensuring that lazy-loaded routes are properly configured to avoid loading unnecessary code.

Difficulty: Medium

8. How does Angular handle module dependencies when using Lazy Loading?

Answer: Angular resolves module dependencies during the lazy loading process by analyzing the imports and ensuring all necessary modules are loaded when a lazy-loaded module is requested.

Difficulty: Hard

9. What are the implications of Lazy Loading on routing and guards in Angular?

Answer: Lazy loading can affect how guards are applied to routes. Guards need to be aware of lazy-loaded modules to prevent unauthorized access, and it may require a different configuration for route protection.

Difficulty: Hard

10. How do you optimize performance with Lazy Loading in Angular applications?

Answer: You can optimize performance by utilizing preloading strategies, ensuring proper caching, minimizing the size of lazy-loaded modules, and analyzing bundle sizes using tools like Webpack.

Difficulty: Hard

ReactJS

1. **Question:** What is React, and why is it used?
   Answer: React is a JavaScript library for building user interfaces, particularly for single-page applications. It allows developers to create reusable UI components and efficiently manage the state of an application.
   Difficulty: Easy
2. **Question:** What are components in React?
   Answer: Components are the building blocks of a React application. They are reusable pieces of code that return React elements (JSX) and can manage their own state. Components can be functional or class-based.
   Difficulty: Easy
3. **Question:** What is JSX?
   Answer: JSX (JavaScript XML) is a syntax extension for JavaScript that allows writing HTML-like code within JavaScript. It makes the code more readable and is transformed into React elements during compilation.
   Difficulty: Easy
4. **Question:** Explain the concept of state in React.
   Answer: State is an object that represents the dynamic data of a component. It determines how the component renders and behaves. When state changes, the component re-renders to reflect the new state.
   Difficulty: Medium
5. **Question:** What are props in React?
   Answer: Props (short for properties) are a way of passing data from a parent component to a child component. They are immutable and help make components reusable.
   Difficulty: Medium
6. **Question:** What is the difference between state and props?
   Answer: State is managed within a component and can change over time, while props are immutable and are passed to a component from its parent. State affects the rendering of the component, while props are used to customize child components.
   Difficulty: Medium
7. **Question:** How do you manage state in a React application?
   Answer: State can be managed using local component state with the useState hook, or globally with state management libraries like Redux, MobX, or React's Context API.
   Difficulty: Medium
8. **Question:** What is a higher-order component (HOC)?
   Answer: A higher-order component is a function that takes a component and returns a new component. HOCs are used for code reuse, such as adding additional functionality to components without modifying their original code.
   Difficulty: Hard

9. **Question:** What are React hooks?
   Answer: React hooks are functions that let you use state and other React features in functional components. Common hooks include useState, useEffect, and useContext. They allow functional components to manage state and side effects.
   Difficulty: Hard

10. **Question:** Explain the concept of the virtual DOM in React.
    Answer: The virtual DOM is an in-memory representation of the real DOM. React creates a virtual DOM to optimize rendering. When the state of a component changes, React updates the virtual DOM first, compares it with the real DOM, and then updates only the parts of the real DOM that have changed, improving performance.
    Difficulty: Hard

React Lazy Loading

1. **What is Lazy Loading in React?**

   Answer: Lazy loading in React is a technique used to delay the loading of components until they are needed, which can improve the initial loading time of the application.

   Difficulty: Easy

2. **How do you implement Lazy Loading for a component in React?**

   Answer: You can implement lazy loading in React using React.lazy() to dynamically import the component and Suspense to handle loading states while the component is being fetched.

   Difficulty: Easy

3. **What is the purpose of React.lazy() in component lazy loading?**

   Answer: React.lazy() is a function that allows you to define a component that will be loaded lazily. It takes a function that returns a dynamic import.

   Difficulty: Easy

4. **How do you handle loading states when using Lazy Loading in React?**

   Answer: You can handle loading states using the Suspense component, which allows you to specify a fallback UI that will be displayed while the lazy-loaded component is being fetched.

   Difficulty: Medium

5. **Can you lazy load routes in React? If so, how?**

   Answer: Yes, you can lazy load routes in React using a combination of React.lazy() and React Router's Route component, typically within a Suspense component for handling loading states.

   Difficulty: Medium

6. **What are the performance benefits of Lazy Loading in React?**

   Answer: Lazy loading reduces the initial bundle size, leading to faster load times and improved performance by only loading components as needed instead of all at once.

Difficulty: Medium

7. **What is code splitting in the context of Lazy Loading in React?**

   Answer: Code splitting is a technique that involves breaking up the code into smaller chunks that can be loaded on demand, which is often achieved through lazy loading.

   Difficulty: Medium

8. **How do you combine Lazy Loading with React's error boundaries?**

   Answer: You can use error boundaries around lazy-loaded components to catch errors during the loading process and display a fallback UI instead of crashing the entire application.

   Difficulty: Hard

9. **What are some best practices for using Lazy Loading in React applications?**

   Answer: Best practices include keeping components small, using meaningful loading indicators, leveraging code splitting effectively, and ensuring error boundaries are implemented.

   Difficulty: Hard

10. **How can you optimize the loading performance of lazily loaded components in React?**

    Answer: You can optimize loading performance by analyzing bundle sizes with tools like Webpack Bundle Analyzer, using dynamic imports strategically, and implementing preloading or prefetching techniques.

    Difficulty: Hard

React Hooks

1. What are React Hooks?

Difficulty: Easy
Answer: React Hooks are functions that let you use state and other React features in functional components without needing to convert them into class components. They were introduced in React 16.8.

2. What is the purpose of the useState Hook?

Difficulty: Easy
Answer: The useState Hook allows you to add state to functional components. It returns an array with the current state and a function to update it.

3. How does the useEffect Hook work?

Difficulty: Medium
Answer: The useEffect Hook allows you to perform side effects in your functional components. It runs after every render by default, but you can control when it runs by passing a dependency array as the second argument.

4. What is the purpose of the dependency array in useEffect?

Difficulty: Medium
Answer: The dependency array tells React when to re-run the effect. If you provide an empty array, the effect runs only once after the initial render. If you provide specific dependencies, the effect runs when those dependencies change.

5. How do you clean up effects in useEffect?

Difficulty: Medium
Answer: You can clean up effects by returning a function from the useEffect callback. This cleanup function is called before the effect runs again or when the component unmounts.

6. Can you use multiple useEffect Hooks in a single component?

Difficulty: Medium
Answer: Yes, you can use multiple useEffect Hooks in a single component. Each effect can handle different side effects independently.

7. What is the difference between useMemo and useCallback?

Difficulty: Medium
Answer: useMemo is used to memoize expensive calculations to optimize performance, while useCallback is used to memoize callback functions to prevent unnecessary re-renders.

8. How do you share state between components using Hooks?

Difficulty: Medium
Answer: You can share state between components by lifting state up to a common ancestor and passing state and updater functions down as props. Alternatively, you can use context with the useContext Hook to provide shared state.

9. What is useReducer and when would you use it?

Difficulty: Hard
Answer: The useReducer Hook is an alternative to useState that is used for managing complex state logic in functional components. It's beneficial when state transitions are complex or when managing state that depends on previous state.

10. Explain the rules of Hooks.

Difficulty: Hard
Answer: The rules of Hooks state that:

1. You can only call Hooks at the top level of your React function (not inside loops, conditions, or nested functions).
2. You can only call Hooks from React function components or custom Hooks, not from regular JavaScript functions.

React Functional Components Life Cycle

1. What is the life cycle of a React functional component?

Difficulty: Easy
Answer: The life cycle of a functional component involves three main phases: mounting (when the component is created and inserted into the DOM), updating (when the component is re-rendered due to state or prop changes), and unmounting (when the component is removed from the DOM).

2. How do functional components differ from class components regarding lifecycle methods?

Difficulty: Easy
Answer: Functional components do not have lifecycle methods like class components. Instead, they use the useEffect Hook to handle side effects at different stages of the lifecycle, mimicking lifecycle methods like componentDidMount, componentDidUpdate, and componentWillUnmount.

3. What is the purpose of the useEffect Hook in managing the component lifecycle?

Difficulty: Medium
Answer: The useEffect Hook allows you to perform side effects in a functional component. You can simulate lifecycle methods by controlling when the effect runs based on the dependency array, enabling you to execute code at mount, update, and unmount phases.

4. How would you mimic componentDidMount using useEffect?

Difficulty: Medium
Answer: To mimic componentDidMount, you can use useEffect with an empty dependency array, which ensures the effect runs only once after the initial render.

5. How do you mimic componentDidUpdate in functional components?

Difficulty: Medium
Answer: You can mimic componentDidUpdate by using useEffect and specifying the dependencies you want to watch. The effect will run every time one of those dependencies changes.

6. How do you mimic componentWillUnmount using useEffect?

Difficulty: Medium
Answer: To mimic componentWillUnmount, return a cleanup function from the useEffect callback. This cleanup function will be called when the component is unmounted or before the effect runs again.

7. What is the purpose of React.StrictMode in relation to lifecycle methods?

Difficulty: Medium
Answer: React.StrictMode is a tool for highlighting potential problems in an application. It intentionally invokes certain lifecycle methods, including the cleanup functions of effects, twice in development mode to help identify side effects that could lead to bugs.

8. How can you optimize performance in functional components?

Difficulty: Medium
Answer: You can optimize performance by using React.memo to prevent unnecessary re-renders, using useMemo to memoize expensive calculations, and useCallback to memoize functions passed as props.

9. What are the drawbacks of using useEffect?

Difficulty: Hard
Answer: Some drawbacks of useEffect include potential for bugs due to incorrect dependency management, complexity in understanding effects with multiple dependencies, and performance issues if effects are not optimized.

10. How would you handle errors in a component lifecycle?

Difficulty: Hard
Answer: You can handle errors in functional components by using error boundaries in class components or by implementing error handling within effects (e.g., using try-catch blocks in asynchronous operations). In React 18, you can also use the ErrorBoundary component to catch errors in functional components.

SQL in General

Easy Level

1. **Question: What is SQL?**
    - **Answer:** SQL stands for Structured Query Language. It is a standard programming language specifically for managing and manipulating relational databases. SQL is used for tasks such as querying data, updating records, and creating or modifying database structures.
    - **Difficulty: Easy**
2. **Question: What is a primary key?**
    - **Answer:** A primary key is a unique identifier for a record in a database table. It ensures that no two rows can have the same value in the primary key column(s) and cannot contain NULL values.
    - **Difficulty: Easy**
3. **Question: What is the difference between INNER JOIN and LEFT JOIN?**
    - **Answer:** An INNER JOIN returns only the rows where there is a match in both tables. A LEFT JOIN returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values are returned for columns from the right table.
    - **Difficulty: Easy**

4. **Question: What is a foreign key?**
    - **Answer:** A foreign key is a column or a set of columns in one table that uniquely identifies a row in another table. It establishes a relationship between the two tables and enforces referential integrity.
    - **Difficulty: Easy**
5. **Question: What is the purpose of the GROUP BY clause?**
    - **Answer:** The GROUP BY clause is used to arrange identical data into groups. It is often used with aggregate functions like COUNT(), SUM(), AVG(), etc., to perform operations on each group of data.
    - **Difficulty: Easy**

Medium Level

6. **Question: What are the differences between UNION and UNION ALL?**
    - **Answer:** UNION combines the result sets of two or more SELECT statements and removes duplicate rows. UNION ALL combines the results without removing duplicates, which can lead to better performance if duplicates are not a concern.
    - **Difficulty: Medium**
7. **Question: How can you prevent SQL injection?**
    - **Answer:** To prevent SQL injection, use prepared statements or parameterized queries instead of concatenating SQL strings. Always validate and sanitize user inputs, and use ORM (Object-Relational Mapping) tools that automatically handle SQL queries safely.
    - **Difficulty: Medium**
8. **Question: Explain the concept of indexes in SQL.**
    - **Answer:** An index is a database object that improves the speed of data retrieval operations on a table. It is created on one or more columns of a table and can significantly reduce the amount of data that needs to be scanned for queries.
    - **Difficulty: Medium**

Hard Level

9. **Question: What is a subquery, and how is it different from a join?**
    - **Answer:** A subquery is a query nested inside another query. It can return a single value, a set of values, or a table. The key difference from a join is that a join combines rows from two or more tables based on a related column, while a subquery can be used to perform operations on individual sets of data.
    - **Difficulty: Hard**

10. **Question: Explain the ACID properties in a database context.**
    - **Answer:** ACID stands for Atomicity, Consistency, Isolation, and Durability. These properties ensure reliable processing of database transactions:
        - Atomicity: Ensures that all operations within a transaction are completed successfully; if one operation fails, the entire transaction fails.
        - Consistency: Ensures that a transaction brings the database from one valid state to another, maintaining data integrity.
        - Isolation: Ensures that concurrent transactions do not affect each other's execution.
        - Durability: Ensures that once a transaction is committed, it remains so, even in the event of a system failure.
    - **Difficulty: Hard**

SQL Database Normalization

Easy Level

1. **Question: What is normalization in a database?**
    - **Answer:** Normalization is the process of organizing data in a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller ones and defining relationships between them.
    - **Difficulty: Easy**
2. **Question: What are the different normal forms?**
    - **Answer:** The different normal forms are:
        - First Normal Form (1NF): Ensures that all columns contain atomic values and each entry in a column is of the same kind.
        - Second Normal Form (2NF): Achieves 1NF and ensures that all non-key attributes are fully functionally dependent on the primary key.
        - Third Normal Form (3NF): Achieves 2NF and removes transitive dependency, where non-key attributes depend on other non-key attributes.
    - **Difficulty: Easy**

Medium Level

3. **Question: Explain First Normal Form (1NF).**
    - **Answer:** A table is in 1NF if all columns contain atomic (indivisible) values, and each entry in a column is of the same data type. There should be no repeating groups or arrays in any column.
    - **Difficulty: Medium**

4. **Question: What is Second Normal Form (2NF), and how do you achieve it?**
   - **Answer:** A table is in 2NF if it is in 1NF and all non-key attributes are fully functionally dependent on the primary key. To achieve 2NF, identify partial dependencies (where non-key attributes depend only on part of a composite key) and move them to separate tables.
   - **Difficulty: Medium**
5. **Question: What is a transitive dependency?**
   - **Answer:** A transitive dependency occurs when a non-key attribute depends on another non-key attribute rather than depending solely on the primary key. This can lead to redundancy and anomalies in data. To resolve transitive dependencies, you can create new tables and establish relationships.
   - **Difficulty: Medium**

Hard Level

6. **Question: Explain Third Normal Form (3NF).**
   - **Answer:** A table is in 3NF if it is in 2NF and there are no transitive dependencies. This means that all non-key attributes must depend only on the primary key. If a non-key attribute depends on another non-key attribute, it should be moved to a separate table to maintain data integrity.
   - **Difficulty: Hard**
7. **Question: What is denormalization, and when would you use it?**
   - **Answer:** Denormalization is the process of intentionally introducing redundancy into a database by merging tables or adding redundant data. It can improve read performance and reduce the complexity of queries, especially in data warehousing scenarios, but can lead to data anomalies if not managed carefully.
   - **Difficulty: Hard**
8. **Question: What are the advantages and disadvantages of normalization?**
   - **Answer:**
     - Advantages: Reduces data redundancy, ensures data integrity, makes the database easier to maintain, and helps avoid anomalies during data operations (insert, update, delete).
     - Disadvantages: Can lead to complex queries, may require more joins (which can slow down performance), and can complicate the database design process.
   - **Difficulty: Hard**

9. **Question: What is Boyce-Codd Normal Form (BCNF)?**
   - **Answer:** BCNF is a stronger version of 3NF. A table is in BCNF if it is in 3NF and, for every functional dependency X→YX \rightarrow YX→Y, XXX is a superkey. This ensures that all dependencies are on a key and helps eliminate redundancy more effectively than 3NF.
   - **Difficulty: Hard**
10. **Question: How do you identify when to normalize a database?**
    - **Answer:** You should consider normalizing a database when you notice data redundancy, update anomalies, or when the same data appears in multiple places. Additionally, if the database design is complex and leads to difficulties in maintenance or querying, it may be time to normalize.
    - **Difficulty: Hard**

SQL Server

1. **Question:** What is SQL Server?
   Answer: SQL Server is a relational database management system (RDBMS) developed by Microsoft. It is used to store and retrieve data as requested by other software applications, and it supports structured query language (SQL) for database operations.
   Difficulty: Easy
2. **Question:** What is a primary key in SQL Server?
   Answer: A primary key is a unique identifier for each record in a table. It ensures that no two rows can have the same value in that column(s) and cannot contain null values.
   Difficulty: Easy
3. **Question:** What is normalization?
   Answer: Normalization is the process of organizing a database to reduce redundancy and improve data integrity. It involves dividing large tables into smaller, related tables and defining relationships between them.
   Difficulty: Easy
4. **Question:** What are joins in SQL Server?
   Answer: Joins are used to combine rows from two or more tables based on a related column. Common types of joins include INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN.
   Difficulty: Medium
5. **Question:** What is a stored procedure?
   Answer: A stored procedure is a precompiled collection of SQL statements that can be executed as a single unit. They can accept parameters, perform operations, and return results. Stored procedures improve performance and security.
   Difficulty: Medium

6. **Question:** Explain the concept of indexes in SQL Server.
   Answer: Indexes are data structures that improve the speed of data retrieval operations on a database table. They work like a book's index, allowing SQL Server to find data quickly without scanning the entire table.
   Difficulty: Medium
7. **Question:** What is a transaction in SQL Server?
   Answer: A transaction is a sequence of one or more SQL operations that are treated as a single unit of work. Transactions ensure data integrity by following the ACID properties (Atomicity, Consistency, Isolation, Durability).
   Difficulty: Medium
8. **Question:** How do you handle errors in SQL Server?
   Answer: Errors in SQL Server can be handled using TRY...CATCH blocks. When an error occurs, control is transferred to the CATCH block, where error handling can be performed, such as logging the error or rolling back a transaction.
   Difficulty: Hard
9. **Question:** What are triggers in SQL Server?
   Answer: Triggers are special types of stored procedures that automatically execute in response to certain events on a table or view, such as INSERT, UPDATE, or DELETE operations. They are used for enforcing business rules or auditing changes.
   Difficulty: Hard
10. **Question:** Explain the concept of data warehousing in SQL Server.
    Answer: Data warehousing is the process of collecting, storing, and managing large volumes of data from various sources for analysis and reporting. SQL Server provides features like SQL Server Integration Services (SSIS) for ETL (Extract, Transform, Load) processes and SQL Server Analysis Services (SSAS) for data analysis.
    Difficulty: Hard

MongoDB

1. **Question:** What is MongoDB?
   Answer: MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It is designed for scalability and performance, allowing developers to work with large volumes of unstructured data.
   Difficulty: Easy
2. **Question:** What are collections in MongoDB?
   Answer: Collections are equivalent to tables in relational databases. They are groups of MongoDB documents and can store documents of varying structures.
   Difficulty: Easy
3. **Question:** Explain the difference between a document and a record in MongoDB.
   Answer: A document is a basic unit of data in MongoDB, represented in BSON (Binary JSON) format. A record, in relational databases, refers to a row in a table. MongoDB documents can contain nested fields, making them more flexible than traditional records.
   Difficulty: Easy

4. **Question:** What is a replica set in MongoDB?
   Answer: A replica set is a group of MongoDB servers that maintain the same dataset. It provides redundancy and high availability by automatically handling failover and data replication.
   Difficulty: Medium
5. **Question:** How do you perform a query in MongoDB?
   Answer: Queries in MongoDB are performed using the find() method, which allows filtering documents based on specified criteria. Queries can also include operators, projections, and sorting options.
   Difficulty: Medium
6. **Question:** What are indexes in MongoDB?
   Answer: Indexes are special data structures that improve the speed of query operations on a collection. MongoDB supports various types of indexes, including single-field, compound, geospatial, and text indexes.
   Difficulty: Medium
7. **Question:** Explain the aggregation framework in MongoDB.
   Answer: The aggregation framework is a powerful tool for processing and transforming data in MongoDB. It allows performing operations like filtering, grouping, and sorting through a pipeline of stages, returning aggregated results.
   Difficulty: Medium
8. **Question:** What is sharding in MongoDB?
   Answer: Sharding is the process of distributing data across multiple servers or clusters to ensure scalability and performance. It allows MongoDB to handle large datasets and high throughput by dividing the data into smaller, more manageable pieces.
   Difficulty: Hard
9. **Question:** How do you handle transactions in MongoDB?
   Answer: MongoDB supports multi-document transactions that allow multiple operations to be executed atomically. Transactions can be started with session.startTransaction(), and operations can be committed or aborted based on success or failure.
   Difficulty: Hard
10. **Question:** What is the difference between MongoDB and traditional relational databases?
    Answer: The primary difference is that MongoDB is a NoSQL database that uses a document-oriented approach, while traditional relational databases use tables and rows. MongoDB offers schema flexibility, horizontal scaling, and high performance for unstructured data, while relational databases enforce strict schemas and use SQL for querying.
    Difficulty: Hard

Git

1. **Question:** What is Git?
   Answer: Git is a distributed version control system that allows multiple developers to collaborate on a project while tracking changes to files over time.
   Difficulty: Easy

2. **Question:** What is a repository in Git?
   Answer: A repository (repo) is a storage space for a project where all the files and their revision history are kept. It can be local (on a developer's machine) or remote (on a server like GitHub).
   Difficulty: Easy
3. **Question:** What is a branch in Git?
   Answer: A branch is a parallel version of a repository. It allows developers to work on features or fixes without affecting the main codebase. The default branch is usually called main or master.
   Difficulty: Easy
4. **Question:** How do you create a new branch in Git?
   Answer: You can create a new branch in Git using the command git branch <branch-name> and switch to it using git checkout <branch-name>, or you can combine both actions using git checkout -b <branch-name>.
   Difficulty: Medium
5. **Question:** Explain the difference between git merge and git rebase.
   Answer: git merge combines two branches by creating a new commit that includes changes from both branches, preserving the history. git rebase integrates changes by moving the entire branch to a new base commit, resulting in a linear history.
   Difficulty: Medium
6. **Question:** What is a pull request?
   Answer: A pull request (PR) is a way to propose changes to a repository. It allows other developers to review and discuss the changes before merging them into the main codebase.
   Difficulty: Medium
7. **Question:** How can you revert a commit in Git?
   Answer: You can revert a commit using the command git revert <commit-hash>, which creates a new commit that undoes the changes from the specified commit. Alternatively, you can use git reset to move the current branch pointer back to a previous commit (with caution as it affects history).
   Difficulty: Medium
8. **Question:** What are Git tags, and how are they used?
   Answer: Git tags are references to specific points in the repository's history, often used to mark release versions. You can create a tag using git tag <tag-name> and push it to a remote repository with git push origin <tag-name>.
   Difficulty: Hard
9. **Question:** How do you handle merge conflicts in Git?
   Answer: Merge conflicts occur when two branches have changes in the same lines of a file. You can resolve conflicts by manually editing the files, removing conflict markers, and then staging the resolved files using git add <file>, followed by git commit.
   Difficulty: Hard

10. **Question:** Explain the concept of Git workflows.
    Answer: Git workflows define the strategies and practices for using Git in collaborative environments. Common workflows include Git Flow, GitHub Flow, and GitLab Flow, each outlining how branches, merging, and releases should be managed to facilitate collaboration and maintain code quality.
    Difficulty: Hard

Scrum

1. **Question:** What is Scrum?
   Answer: Scrum is an Agile framework for managing complex projects. It emphasizes iterative progress, collaboration, and adaptability to changing requirements through a structured approach with roles, events, and artifacts.
   Difficulty: Easy

2. **Question:** What are the key roles in Scrum?
   Answer: The key roles in Scrum are the Product Owner, who defines the product backlog and prioritizes features; the Scrum Master, who facilitates the Scrum process and removes obstacles; and the Development Team, which delivers the product increment.
   Difficulty: Easy

3. **Question:** What is a sprint in Scrum?
   Answer: A sprint is a time-boxed iteration, typically lasting 1 to 4 weeks, during which a potentially shippable product increment is created. It begins with planning and ends with a review and retrospective.
   Difficulty: Easy

4. **Question:** Explain the purpose of a sprint backlog.
   Answer: The sprint backlog is a list of tasks and user stories that the Development Team commits to completing during a sprint. It is a subset of the product backlog and is updated daily to reflect progress.
   Difficulty: Medium

5. **Question:** What is the difference between a product backlog and a sprint backlog?
   Answer: The product backlog is a prioritized list of features, enhancements, and bug fixes for the entire product, while the sprint backlog is a subset of the product backlog that the team commits to completing during the current sprint.
   Difficulty: Medium

6. **Question:** What are the Scrum ceremonies?
   Answer: The Scrum ceremonies include Sprint Planning, Daily Stand-up (Daily Scrum), Sprint Review, and Sprint Retrospective. Each ceremony serves a specific purpose in facilitating communication and progress tracking within the team.
   Difficulty: Medium

7. **Question:** How do you estimate work in Scrum?
   Answer: Work in Scrum is typically estimated using techniques like Story Points, T-shirt sizes, or Ideal Days. Teams discuss and assign estimates to user stories based on complexity and effort, often using planning poker for consensus.
   Difficulty: Medium

8. **Question:** What is the definition of done in Scrum?
   Answer: The Definition of Done (DoD) is a shared understanding among the team of what it means for a product increment to be considered complete. It includes criteria like passing tests, code reviews, and documentation.
   Difficulty: Hard

9. **Question:** Explain the concept of a Scrum Board.
   Answer: A Scrum Board is a visual representation of the work in a sprint. It displays the status of tasks, typically organized into columns such as "To Do," "In Progress," and "Done," facilitating team collaboration and transparency.
   Difficulty: Hard

10. **Question:** How do you handle changes to requirements during a sprint?
    Answer: Changes to requirements during a sprint are generally discouraged to maintain focus. However, if critical changes arise, the Scrum Team can discuss them and decide if they will be added to the product backlog for consideration in the next sprint rather than altering the current sprint's scope.
    Difficulty: Hard

C# LINQ

1. What is LINQ, and how is it different from traditional SQL?

- **Difficulty:** Easy
- **Answer:** LINQ (Language Integrated Query) is a .NET framework component that adds native data querying capabilities to .NET languages. Unlike traditional SQL, which is executed against a database, LINQ queries can be used with various data sources, such as arrays, lists, XML, and databases, directly within the C# language.

2. What is deferred execution in LINQ?

- **Difficulty:** Easy
- **Answer:** Deferred execution means that the evaluation of a LINQ query is delayed until the query variable is iterated over. This allows for optimization and can improve performance, as the data is only retrieved when needed.

3. How do you perform a join operation in LINQ?

- **Difficulty:** Medium
- **Answer:** You can perform joins in LINQ using the Join method or the query syntax. For example:

csharp

Copiar código

```
var result = from a in collectionA
             join b in collectionB on a.Key equals b.Key
             select new { a, b };
```

4. Explain the difference between IEnumerable and IQueryable.

- **Difficulty:** Medium
- **Answer:** IEnumerable is used for in-memory collections and executes queries against the data in memory. IQueryable is designed for querying remote data sources, such as databases, and constructs an expression tree to be executed on the server.

5. What are lambda expressions, and how are they used in LINQ?

- **Difficulty:** Medium
- **Answer:** Lambda expressions are anonymous functions that can contain expressions and statements. They are used in LINQ to create inline functions for query operations, such as Select, Where, and OrderBy. Example:

csharp

Copiar código

```
var result = collection.Where(x => x.Property == value);
```

6. What is the GroupBy method in LINQ, and how is it used?

- **Difficulty:** Medium
- **Answer:** The GroupBy method is used to group elements that share a common attribute. It returns a collection of groups, where each group contains elements that match the specified key. Example:

csharp

Copiar código

```
var grouped = collection.GroupBy(x => x.KeyProperty);
```

7. How do you handle null values in LINQ queries?

- **Difficulty:** Medium
- **Answer:** You can use the null-coalescing operator ?? or the DefaultIfEmpty method to handle nulls in LINQ queries. Example:

csharp

Copiar código

```
var result = collection.Where(x => x.Property != null);
```

8. Explain the purpose of the SelectMany method in LINQ.

- **Difficulty:** Medium
- **Answer:** SelectMany is used to flatten a collection of collections into a single collection. It projects each element of a sequence to an IEnumerable and flattens the resulting sequences into one sequence. Example:

csharp

Copiar código

```
var result = collection.SelectMany(x => x.SubCollection);
```

9. What is LINQ to XML, and how does it work?

- **Difficulty:** Hard
- **Answer:** LINQ to XML is a set of classes that allow for querying and manipulating XML documents in a more concise way. You can load XML into an XDocument, and then use LINQ queries to access and modify its elements. Example:

csharp

Copiar código

```
var xml = XDocument.Load("file.xml");

var result = from el in xml.Descendants("Element")

             select el.Value;
```

10. How can you improve the performance of LINQ queries?

- **Difficulty:** Hard
- **Answer:** You can improve LINQ query performance by:
  - Using AsQueryable for remote queries.
  - Avoiding unnecessary ToList() calls.
  - Using Select to retrieve only necessary fields.
  - Implementing caching for repeated queries.
  - Ensuring indexes are applied in the underlying database.

C# MVC

1. What is MVC architecture?

- **Difficulty:** Easy
- **Answer:** MVC (Model-View-Controller) is a design pattern used for developing user interfaces that separates the application into three interconnected components: Model (data), View (UI), and Controller (business logic). This separation promotes organized code and scalability.

2. What are the roles of Model, View, and Controller in MVC?

- **Difficulty:** Easy
- **Answer:**
  - **Model:** Represents the data and business logic.
  - **View:** Represents the UI that displays the data.
  - **Controller:** Handles user input, interacts with the model, and selects the view to display.

3. Explain the concept of routing in ASP.NET MVC.

- **Difficulty:** Medium
- **Answer:** Routing in ASP.NET MVC is the mechanism that maps incoming requests to the appropriate controller and action. It uses the URL pattern defined in the route configuration to determine how to handle the request.

4. How do you pass data from a controller to a view?

- **Difficulty:** Medium
- **Answer:** Data can be passed from a controller to a view using:
  - **ViewBag:** A dynamic object that allows you to pass data to the view.
  - **ViewData:** A dictionary that holds data to be passed to the view.
  - **Model:** A strongly typed object that represents the data for the view.

5. What are Action Filters in ASP.NET MVC?

- **Difficulty:** Medium
- **Answer:** Action Filters are attributes that can be applied to controllers or action methods to execute custom logic before or after action method execution. They can be used for logging, authentication, or modifying the result. Examples include Authorize, ActionName, and OutputCache.

6. How do you create a custom Action Filter?

- **Difficulty:** Medium
- **Answer:** To create a custom Action Filter, you implement the IActionFilter interface or derive from ActionFilterAttribute. You then override the OnActionExecuting or OnActionExecuted methods to add your logic. Example:

csharp

Copiar código

```
public class CustomActionFilter : ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        // Custom logic here
    }
}
```

7. What is model binding in ASP.NET MVC?

- **Difficulty:** Medium
- **Answer:** Model binding is the process by which MVC maps data from HTTP requests to action method parameters or model properties. It converts data types, handles complex types, and validates the incoming data automatically.

8. How do you handle validation in ASP.NET MVC?

- **Difficulty:** Medium
- **Answer:** Validation can be handled using data annotations on model properties, which specify rules like Required, StringLength, and Range. MVC automatically validates the model based on these annotations when the form is submitted.

9. What are Partial Views, and how are they used?

- **Difficulty:** Hard
- **Answer:** Partial Views are reusable views that can be rendered within other views. They help break down complex views into smaller components. They can be used with Html.Partial() or Html.RenderPartial() to include them in a parent view.

10. How do you implement dependency injection in ASP.NET MVC?

- **Difficulty:** Hard
- **Answer:** Dependency injection in ASP.NET MVC can be implemented by configuring a dependency injection container (like Autofac or Unity) in the Global.asax file or using a service

provider. You register services in the container and resolve them in controllers via constructor injection.

Azure CI/CD Pipelines

1. What is CI/CD in the context of Azure DevOps?

- **Difficulty:** Easy
- **Answer:** CI/CD stands for Continuous Integration and Continuous Deployment. In Azure DevOps, CI/CD pipelines automate the process of integrating code changes, running tests, and deploying applications to different environments, enabling faster and more reliable software delivery.

2. What is the difference between Continuous Integration and Continuous Deployment?

- **Difficulty:** Easy
- **Answer:** Continuous Integration (CI) is the practice of automatically integrating code changes into a shared repository and running tests to detect issues early. Continuous Deployment (CD) is the practice of automatically deploying code changes to production after passing CI tests, ensuring that new features are available to users quickly.

3. How do you create a CI/CD pipeline in Azure DevOps?

- **Difficulty:** Medium
- **Answer:** You can create a CI/CD pipeline in Azure DevOps by navigating to the Pipelines section, selecting "New Pipeline," and then configuring the source repository, selecting a template, and defining build and deployment steps in the YAML file or using the visual designer.

4. What are the key components of a CI/CD pipeline in Azure?

- **Difficulty:** Medium
- **Answer:** The key components of a CI/CD pipeline in Azure include:
  - **Build Pipeline:** Compiles code, runs tests, and produces artifacts.
  - **Release Pipeline:** Deploys artifacts to target environments.
  - **Triggers:** Define when pipelines are executed (e.g., on code push, pull request).
  - **Tasks:** Individual actions that are performed within the pipeline (e.g., build, test, deploy).

5. How can you implement environment-specific configurations in Azure Pipelines?

- **Difficulty:** Medium
- **Answer:** You can implement environment-specific configurations using variable groups, pipeline variables, or Azure App Configuration. These allow you to define and manage settings for different environments, such as development, staging, and production.

6. What is a YAML pipeline, and how does it differ from a classic pipeline?

- **Difficulty:** Medium
- **Answer:** A YAML pipeline is defined using YAML syntax, which allows for versioning and storing the pipeline as code within the repository. In contrast, a classic pipeline is configured using a visual interface in Azure DevOps. YAML pipelines provide greater flexibility and reusability.

7. How do you handle secrets and sensitive information in Azure Pipelines?

- **Difficulty:** Medium
- **Answer:** You can handle secrets and sensitive information in Azure Pipelines using Azure Key Vault or pipeline variables marked as secrets. Secrets are encrypted and not exposed in logs, ensuring secure handling of sensitive data during builds and deployments.

8. What are deployment strategies you can implement in Azure DevOps?

- **Difficulty:** Hard
- **Answer:** Common deployment strategies in Azure DevOps include:
  - **Blue-Green Deployment:** Two identical environments (blue and green) allow switching traffic with minimal downtime.
  - **Canary Deployment:** Gradual rollout of changes to a small subset of users to minimize risk.
  - **Rolling Deployment:** Sequentially replacing instances of the application with new versions.

9. How can you integrate automated testing into your CI/CD pipeline?

- **Difficulty:** Hard
- **Answer:** Automated testing can be integrated into CI/CD pipelines by adding test tasks in the build pipeline to run unit tests, integration tests, and UI tests using frameworks like NUnit, MSTest, or Selenium. Test results can be published and monitored in Azure DevOps.

10. How do you monitor and troubleshoot Azure Pipelines?

- **Difficulty:** Hard
- **Answer:** Monitoring and troubleshooting Azure Pipelines can be done by:
  - Viewing logs for each pipeline run to identify errors or warnings.
  - Using Azure Monitor to track pipeline performance and success rates.
  - Setting up alerts for failures or issues in the pipeline.
  - Leveraging built-in diagnostic tools to analyze specific tasks.

Big O notation

What is Big O notation?

Answer: Big O notation is a mathematical notation used to describe the upper bound of the time complexity or space complexity of an algorithm in computer science. It provides a high-level understanding of the performance characteristics of an algorithm, particularly in terms of how the time or space requirements grow as the input size increases.