1. Which of the SOLID principles is related to the idea that a class should have only one reason to change?

a) Open/Closed Principle
b) Single Responsibility Principle
c) Liskov Substitution Principle
d) Interface Segregation Principle

2. In C#, which keyword is used to indicate that a method can be overridden in a derived class?

a) virtual
b) override
c) sealed
d) abstract

3. Which design pattern is often used to implement Inversion of Control (IoC) and Dependency Injection (DI) in C# projects?

a) Singleton
b) Factory Method
c) Dependency Injection
d) Observer

4. In which design pattern is it common to use an interface to create families of related or dependent objects without specifying their concrete classes?

a) Factory Method
b) Abstract Factory
c) Prototype
d) Builder

5. What is the main difference between INNER JOIN and LEFT JOIN in SQL queries?

a) INNER JOIN and LEFT JOIN are identical and can be used interchangeably in SQL queries
b) INNER JOIN returns only records that have matches in both tables, while LEFT JOIN returns all records from the left table and the matching records from the right table
c) INNER JOIN returns all records from the left table and the matching records from the right table, while LEFT JOIN returns only records that have matches in both tables
d) INNER JOIN returns all records from both tables, while LEFT JOIN returns only the records from the left table that do not have matches in the right table.

6. Which design pattern is most appropriate when you need to ensure that a class has only one instance and provide a global point of access to that instance?

a) Singleton
b) Prototype
c) Factory Method
d) Builder

7. Which of the following is a design principle that promotes creating systems where low-level parts can be easily replaced by other implementations?

a) Open/Closed Principle
b) Dependency Inversion Principle
c) Liskov Substitution Principle
d) Interface Segregation Principle

8. In C#, which design pattern is often used to encapsulate the creation of complex objects, providing a simplified interface to the client?

a) Singleton
b) Builder
c) Adapter
d) Observer

9. Which design pattern is most appropriate when you need to ensure that a derived class can be used in place of its base class without affecting the program's behavior?

a) Singleton

b) Factory Method
c) Liskov Substitution Principle
d) Adapter

10. What does the acronym SOLID stand for?
    a) Five principles of object-oriented design
    b) A functional programming technique
    c) A set of programming languages
    d) A software architecture pattern

11. Which design pattern is most appropriate when you need to control the creation of a complex object step by step?
    a) Singleton
    b) Builder
    c) Factory Method
    d) Abstract Factory

12. Which of the following is a benefit of the Observer pattern?
    a) Reduction of code complexity
    b) Improved code readability
    c) Increased code cohesion
    d) Promotion of code reusability

13. Which design pattern is most appropriate when you need to ensure that a class has only one instance and provide a global point of access to that instance, but with a more efficient performance synchronization mechanism?
    a) Singleton
    b) Monostate
    c) Multiton
    d) Lazy Initialization

14. Which design pattern is most appropriate when you need to ensure that a class can be extended without modifying existing code?

a) Adapter
b) Observer
c) Decorator
d) Facade

15. Which of the following SOLID principles suggests that a class should be open for extension but closed for modification?

a) Single Responsibility Principle
b) Open/Closed Principle
c) Liskov Substitution Principle
d) Interface Segregation Principle

16. Which design pattern is most appropriate when you need to convert the interface of a class into another interface that a client expects?

a) Adapter
b) Facade
c) Decorator
d) Bridge

17. Which design pattern is most appropriate when you need to represent an operation to be performed on a set of objects in a tree structure?

a) Strategy
b) Visitor
c) Composite
d) Chain of Responsibility

18. In C#, which keyword is used to define a class that cannot be instantiated directly and that may contain abstract methods?

a) virtual
b) override
c) sealed
d) abstract

19. Which design pattern is most appropriate when you need to decouple objects so that changes in one object do not require changes in others?

a) Mediator
b) Memento
c) Chain of Responsibility
d) Observer

20. Which design pattern is most appropriate when you need to define an interface to create an object, but allow subclasses to decide which class to instantiate?

a) Factory Method
b) Abstract Factory
c) Prototype
d) Builder

SQL:

1. In SQL, which keyword is used to remove rows from a table?
   a) DELETE
   b) REMOVE
   c) DROP
   d) ERASE

2. Which SQL command is used to add a new column to an existing table?
   a) ADD COLUMN
   b) ALTER TABLE
   c) INSERT INTO
   d) UPDATE TABLE

3. Which SQL clause is used to return only distinct values?
   a) DISTINCT
   b) UNIQUE
   c) DIFFERENT
   d) DISTINCTIVE

4. In SQL, which aggregate function returns the average value of a numeric column?
   a) AVG()
   b) SUM()
   c) COUNT()
   d) MAX()

5. Which SQL statement is used to sort the result set in descending order?
   a) ORDER BY DESC
   b) SORT DESC
   c) DESCENDING
   d) DESC

Service-Oriented Architecture (SOA):

1. What is a key characteristic of Service-Oriented Architecture (SOA)?

a) Tight coupling between services
b) Heavy reliance on monolithic applications
c) Loosely coupled services
d) Limited scalability

2. In SOA, what is a service contract?

a) A legally binding document between service providers
b) A document describing how services communicate with each other
c) A set of predefined rules governing service interactions
d) An agreement between service consumers and providers

3. Which of the following is a primary advantage of using SOA?

a) Reduced complexity
b) Tight integration between systems
c) Limited reusability of services
d) High coupling between services

4. In SOA, what is the role of a service consumer?

a) Provides services to other components
b) Consumes and interacts with services
c) Defines service contracts
d) Manages service deployments

5. Which technology is commonly used for communication between services in a Service-Oriented Architecture?

a) SOAP (Simple Object Access Protocol)
b) JSON (JavaScript Object Notation)
c) XML (eXtensible Markup Language)
d) REST (Representational State Transfer)

Hexagonal

1. **What is hexagonal architecture and what are its main objectives?**

a) It is a programming paradigm based on six sides to facilitate the implementation of graphical interfaces.
b) It is an architectural pattern that aims to separate business logic from infrastructure code and facilitate the substitution of external components.
c) It is a coding method for computer systems that use hexadecimal calculations instead of binaries.
d) It is an algorithm for converting decimal numbers to hexadecimal efficiently.

2. **What are the main components of a typical hexagonal architecture and how do they interact with each other?**

a) Core components include adapters, ports, and drivers; they interact using RESTful API calls.
b) Core components include domain entities, use cases, and adapters; they interact using asynchronous communication.
c) Core components include controllers, services, and repositories; they interact using dependency injection.
d) Core components include ports, adapters, and applications; they interact through domain events and asynchronous messages.

3. **How does hexagonal programming facilitate automated testing in a system?**

a) By integrating all components into a single module, making it easier to test the system as a whole.
b) By decoupling business logic from implementation details and infrastructure, allowing more effective unit and integration testing.
c) By restricting access to external components, making it more difficult to test the system's integration with other systems.
d) By using only manual testing instead of automated tests, ensuring broader coverage.

4. **What is the difference between a port and an adapter in hexagonal architecture?**

a) A port defines the interface that allows the system to communicate with the outside world, while an adapter implements business logic. #
b) A port implements business logic, while an adapter is responsible for communicating with external databases.
c) A port represents a specific implementation of a use case, while an adapter deals with data transformation into different formats.
d) A port and an adapter are interchangeable terms and refer to the same thing in hexagonal programming.

5. **What are the possible challenges when implementing hexagonal programming in a software development project?**

a) Challenges may include increased initial complexity due to the need to structure code around well-defined layers. #
b) Challenges may include lack of support from popular tools and frameworks that are not aligned with the principles of hexagonal programming.
c) Challenges may include difficulties in integration with legacy systems and external dependencies that do not follow the hexagonal programming paradigm.
d) All of the above options are correct, as implementing hexagonal programming can pose various challenges depending on the project context.

Microservices

1. **What are microservices, and how do they differ from traditional monolithic architectures?**

a) Microservices are small software components that encapsulate specific functionalities and communicate through synchronous protocols, while monolithic architectures consist of a single, tightly integrated application.
b) Microservices are large-scale enterprise applications designed for high availability and fault tolerance, while monolithic architectures are small, single-purpose applications.
c) Microservices are tightly coupled components that share a single codebase, while monolithic architectures consist of loosely coupled, independently deployable services.
d) Microservices and monolithic architectures are terms used interchangeably to refer to the same architectural style.

2. **What are the advantages and disadvantages of using microservices compared to monolithic architectures?**

a) Advantages of microservices include improved scalability and fault isolation, while disadvantages include increased operational complexity and potential for distributed system failures.
b) Advantages of monolithic architectures include simplicity in development and deployment, while disadvantages include limited scalability and difficulty in maintaining large codebases.
c) Both microservices and monolithic architectures offer the same advantages and disadvantages, but with different trade-offs depending on the specific use case.
d) Advantages of microservices include reduced development time and easier debugging, while disadvantages include decreased flexibility and increased coupling between components.

3. **How do microservices handle data management and storage compared to monolithic architectures?**

a) Microservices typically use a single, centralized database for all services, while monolithic architectures rely on distributed databases to ensure fault tolerance.
b) Microservices implement data isolation by each service having its database, while monolithic architectures use shared databases across all components.
c) Microservices and monolithic architectures use the same approach to data management and storage, with no significant differences between them.
d) Microservices depend on external data stores managed by third-party providers, while monolithic architectures manage all data storage internally.

4. **What challenges arise when implementing microservices in terms of inter-service communication and orchestration?**

a) Challenges may include dealing with network latency, ensuring message reliability, and implementing robust service discovery mechanisms.
b) Challenges are minimal as microservices inherently simplify communication through lightweight protocols and centralized orchestration.
c) Challenges primarily revolve around managing service dependencies and versioning, as microservices require strict adherence to API contracts.
d) Challenges arise from the need for extensive configuration management and resource provisioning, which can complicate deployment processes.

5. **How do microservices impact organizational structure and development processes compared to monolithic architectures?**

a) Microservices encourage smaller, cross-functional teams that take ownership of individual services, while monolithic architectures promote large, specialized teams focused on specific layers of the application.
b) Microservices have no significant impact on organizational structure, as they are purely technical implementations with no influence on team dynamics.
c) Microservices require centralized control and coordination across teams to ensure consistency and compatibility, whereas monolithic architectures allow for more autonomous development efforts.
d) Microservices and monolithic architectures both require similar organizational structures and development processes, with no significant differences between them.