**C# Basics and Object-Oriented Programming**

1. **What are the key principles of OOP (Object-Oriented Programming)?**
   * **Explain the four principles: Encapsulation, Inheritance, Polymorphism, and Abstraction, with examples.**
2. **What is the difference between an abstract class and an interface?**
   * **Discuss when to use abstract classes vs. interfaces and their practical applications.**
3. **What is the difference between override, overload, and new in C#?**
   * **Explain method overriding, method overloading, and hiding methods using the new keyword.**
4. **What are access modifiers in C#?**
   * **Explain public, private, protected, internal, and protected internal.**
5. **What is the difference between a value type and a reference type in C#?**
   * **Cover stack vs. heap allocation and examples like structs (value type) and classes (reference type).**

**Memory Management**

1. **What is garbage collection, and how does it work in C#?**
   * **Explain the concept of managed memory, the GC process, and terms like generations (Gen 0, 1, 2).**
2. **What is the Dispose pattern? How is it related to the IDisposable interface?**
   * **Explain how to implement Dispose and its relation to releasing unmanaged resources.**
3. **What is the difference between finalize and dispose?**
   * **Describe when each is called and their purpose in memory management.**

**Advanced Concepts**

1. **What are delegates in C#? How are they different from events?**
   * **Discuss delegates as function pointers and how events provide encapsulation.**
2. **What is async and await in C#?**
   * **Explain the concept of asynchronous programming, the Task-based Asynchronous Pattern (TAP), and how async/await simplifies writing asynchronous code.**
3. **What are extension methods?**
   * **Discuss how extension methods work and how to use them to add functionality to existing types without modifying them.**
4. **What is LINQ?**
   * **Explain LINQ (Language Integrated Query) with examples of querying collections and databases.**
5. **What is the difference between IEnumerable, IQueryable, and List in C#?**
   * **Cover where each is used and their execution models (e.g., deferred execution in LINQ).**

**Multithreading and Parallel Programming**

1. **What is the difference between Task, Thread, and ThreadPool in C#?**
   * **Discuss when to use each and their pros/cons.**
2. **What is lock in C#? How does it prevent race conditions?**
   * **Explain the use of the lock keyword and how it synchronizes access to shared resources.**
3. **What are async and await, and how do they differ from traditional multithreading?**
   * **Highlight the differences between asynchronous programming and multithreading.**
4. **What is volatile in C#?**
   * **Explain how the volatile keyword ensures visibility of changes to a variable across threads.**
5. **What is the difference between Parallel.ForEach and Task.WhenAll?**
   * **Cover their use cases and performance implications.**

**Design Patterns**

1. **What are the most common design patterns used in C#?**
   * **Discuss patterns like Singleton, Factory, Repository, Dependency Injection, and Observer.**
2. **How is the Singleton pattern implemented in C#?**
   * **Provide examples of thread-safe Singleton using Lazy<T>.**
3. **What is Dependency Injection (DI)?**
   * **Explain the principle and demonstrate how it is implemented in .NET Core using built-in DI container.**
4. **What is the difference between a factory and an abstract factory pattern?**
   * **Provide scenarios where each is used.**

**.NET Core and ASP.NET Core**

1. **What is the difference between .NET Framework and .NET Core?**
   * **Discuss platform independence, performance improvements, and microservices compatibility.**
2. **What is middleware in ASP.NET Core?**
   * **Explain the pipeline concept in ASP.NET Core and give examples of custom middleware.**
3. **What are the different lifetimes of services in dependency injection (DI) in ASP.NET Core?**
   * **Explain Singleton, Scoped, and Transient lifetimes with examples.**
4. **How do you configure and consume logging in .NET Core?**
   * **Cover the built-in ILogger interface and its configuration.**
5. **What is the difference between IActionResult and IEnumerable in ASP.NET Core?**
   * **Explain their roles in web APIs and MVC.**
6. **What is Kestrel?**
   * **Discuss Kestrel as a cross-platform, high-performance web server used in .NET Core.**

**Best Practices and Performance**

1. **How do you improve the performance of a C# application?**
   * **Discuss caching, asynchronous programming, efficient data structures, and lazy loading.**
2. **What is a memory leak in .NET, and how can you prevent it?**
   * **Explain scenarios like unreferenced event handlers and using proper Dispose patterns.**
3. **How do you optimize LINQ queries for performance?**
   * **Discuss tips like avoiding ToList() prematurely and using AsEnumerable() wisely.**

**Unit Testing and Debugging**

1. **How do you implement unit tests in C#?**
   * **Discuss using xUnit, NUnit, or MSTest.**
2. **What is mocking in unit testing?**
   * **Explain mocking frameworks like Moq and how to test dependencies.**
3. **How do you debug performance issues in a C# application?**
   * **Cover tools like Visual Studio Profiler, dotTrace, and Application Insights.**

**Entity Framework and Data Access**

1. **What is the difference between DbContext and DbSet in Entity Framework?**
   * **Discuss their roles in database operations.**
2. **What are migrations in Entity Framework?**
   * **Explain the use of migrations for schema management.**
3. **What is the difference between lazy loading and eager loading in Entity Framework?**
   * **Discuss their pros and cons and use cases.**
4. **What is the difference between First(), FirstOrDefault(), and Single() in LINQ?**
   * **Provide examples of when to use each method.**

**Interview Scenario-Based Questions**

1. **Explain a challenging bug you encountered and how you resolved it.**
   * **Share a real-world scenario showcasing problem-solving and debugging skills.**
2. **Design a thread-safe cache mechanism in C#.**
   * **Provide a basic implementation using ConcurrentDictionary or other thread-safe collections.**
3. **How would you handle a high-volume message queue using Kafka and C#?**
   * **Discuss strategies for partitioning, consuming, and processing messages efficiently.**

**C# Language Deep Dive**

1. **What are covariance and contravariance in C#?**
   * Explain how covariance and contravariance work in delegates, generics, and interfaces with examples.
2. **Explain the difference between dynamic, var, and object in C#.**
   * Discuss their usage, compile-time vs. runtime behavior, and scenarios where they are most applicable.
3. **What is the readonly keyword? How does it differ from const?**
   * Compare immutability in readonly fields vs. compile-time constants.
4. **What are record types in C#? How are they different from classes?**
   * Explain immutability, value equality, and use cases of **record types** introduced in C# 9.
5. **What is a tuple, and how is it different from a struct?**
   * Discuss the use of tuples in modern C# and when to use structs instead.
6. **What are span types (Span<T> and Memory<T>), and how do they improve performance?**
   * Cover memory-efficient operations, slicing, and use in high-performance scenarios.

**Advanced Object-Oriented Programming**

1. **What is the difference between a shallow copy and a deep copy in C#?**
   * Discuss how to implement both and their implications on memory.
2. **Explain the SOLID principles with real-world examples.**
   * Focus on how these principles are implemented in large-scale C# projects.
3. **What is polymorphism, and how does it differ in compile-time and runtime?**
   * Include examples of operator overloading (compile-time) and method overriding (runtime).
4. **What is dependency inversion, and how is it different from dependency injection?**
   * Explain their relationship and how they fit into modern application architectures.

**Multithreading and Asynchronous Programming**

1. **What is the difference between async/await and traditional threading in C#?**
   * Discuss how Task-based asynchronous programming differs from manual thread management.
2. **What is the role of SynchronizationContext in C#?**
   * Explain how it affects asynchronous operations and UI thread synchronization.
3. **What are Monitor, Mutex, Semaphore, and AutoResetEvent in C#?**
   * Discuss their differences, use cases, and impact on thread synchronization.
4. **What is IAsyncEnumerable<T>, and how does it improve asynchronous data streaming?**
   * Demonstrate its usage for processing data streams in a non-blocking way.
5. **What is the ConcurrentDictionary in C#, and how does it differ from Dictionary?**
   * Explain thread safety and performance improvements with concurrent collections.

**Memory Management and Performance**

1. **How does the garbage collector work in .NET?**
   * Discuss concepts like **generations**, **finalization**, and **large object heap (LOH)**.
2. **What are weak references in C#?**
   * Explain their purpose and when to use them to avoid memory leaks.
3. **What is the difference between Span<T> and ArraySegment<T>?**
   * Discuss their roles in memory slicing and performance optimization.
4. **How would you diagnose and fix a memory leak in a .NET application?**
   * Talk about tools like **dotMemory**, **CLR Profiler**, or **Visual Studio Diagnostic Tools**.
5. **What are the differences between value types and reference types in terms of memory management?**
   * Discuss boxing, unboxing, and implications on performance.

**Design Patterns**

1. **What is the difference between the Singleton and Dependency Injection patterns?**
   * Compare their purposes and how they’re implemented in C#.
2. **What is the CQRS pattern? How is it implemented in a C# application?**
   * Explain **Command Query Responsibility Segregation** and how it fits into distributed systems.
3. **Explain the Repository pattern and its advantages.**
   * Provide an example of implementing the Repository pattern with Entity Framework.
4. **What is the strategy pattern? How is it different from the state pattern?**
   * Discuss the differences with examples in real-world scenarios.
5. **What is the observer pattern? How does C# implement it using events and delegates?**
   * Demonstrate how C# simplifies the observer pattern with events.

**Entity Framework and Database**

1. **What is the difference between AsNoTracking() and regular tracking in Entity Framework?**
   * Discuss performance benefits and scenarios where AsNoTracking() is preferred.
2. **What is the difference between Include and Select in LINQ queries?**
   * Explain lazy loading, eager loading, and projection with examples.
3. **What is a DbContext? How is it managed in a multi-threaded application?**
   * Discuss issues like connection pooling and how to use DbContext safely.
4. **What are database migrations in Entity Framework, and how do you manage them in a team environment?**
   * Cover scenarios for handling conflicts in migrations.
5. **What are the performance implications of using LINQ to SQL vs. stored procedures?**
   * Compare flexibility vs. performance considerations.

**ASP.NET Core and Microservices**

1. **What is middleware in ASP.NET Core? How do you implement custom middleware?**
   * Provide an example of writing and integrating middleware in the request pipeline.
2. **What are filters in ASP.NET Core? How are they different from middleware?**
   * Explain the different types of filters (e.g., authorization, action, result).
3. **What is the difference between IActionResult and ActionResult<T> in Web APIs?**
   * Discuss their role in HTTP response generation.
4. **What are the key components of a microservices architecture?**
   * Discuss concepts like **API Gateway**, **service discovery**, and **event-driven communication**.
5. **How does ASP.NET Core handle dependency injection?**
   * Explain service lifetimes (Transient, Scoped, and Singleton) with examples.

**LINQ and Functional Programming**

1. **How does deferred execution work in LINQ?**
   * Discuss when queries are executed and how it affects performance.
2. **What is the difference between Select and SelectMany in LINQ?**
   * Explain with examples to clarify their behavior in flattening collections.
3. **What is the difference between First(), FirstOrDefault(), Single(), and SingleOrDefault()?**
   * Provide examples and when to use each.
4. **What is expression trees in C#?**
   * Explain how expression trees are used in LINQ and dynamic queries.
5. **What is PLINQ (Parallel LINQ)? How does it improve performance?**
   * Discuss scenarios where PLINQ is advantageous.

**Code Design and Architecture**

1. **How do you design a thread-safe Singleton in C#?**
   * Provide an example using Lazy<T>.
2. **How do you handle circular dependencies in a .NET application?**
   * Discuss strategies like refactoring, service locator pattern, or breaking dependencies.
3. **What is the difference between tightly coupled and loosely coupled systems?**
   * Provide examples of achieving loose coupling in C# applications.
4. **How do you implement logging and error handling in a distributed application?**
   * Discuss tools like **Serilog**, **NLog**, or **Application Insights**.
5. **What is the Clean Architecture, and how would you implement it in a .NET project?**
   * Explain its principles and layers: **Domain**, **Application**, **Infrastructure**, and **Presentation**.

**Scenario-Based Questions**

1. **How would you design a high-performance real-time chat application in C#?**
   * Discuss concepts like **SignalR**, WebSocket, and asynchronous operations.
2. **How would you optimize a slow-performing LINQ query in C#?**
   * Mention deferred execution, projection, and caching techniques.
3. **How would you handle rate-limiting in an ASP.NET Core API?**
   * Discuss using middleware or external tools like **Polly**.
4. **What would you do if you detect a memory leak in your application?**
   * Discuss debugging tools and fixing strategies.
5. **How do you ensure high availability and fault tolerance in a C# microservices-based application?**
   * Mention techniques like **circuit breakers**, retries, and load balancing.