1. Records & Immutability

Q: Explain the difference between a `record`, `record struct`, and a regular `class` in C#. In what scenarios would you prefer one over the other?

Key Points:
- `record` → reference type, value-based equality, built for immutability.
- `record struct` → value type, value-based equality, stored on stack.
- `class` → reference type, reference-based equality unless overridden.

Extra Depth:
- Records are ideal for DTOs, immutables, and DDD value objects.
- Record structs reduce allocations.

Example:
```csharp
public record Person(string FirstName, string LastName);
public record struct Point(int X, int Y);
public class PersonClass { public string FirstName { get; set; } }
```

2. Init-only Setters

Q: What are `init`-only setters in C#? Can they be bypassed?

Key Points:
- Allow property initialization only during object construction.
- Cannot change after creation, enforcing immutability-like behavior.

Extra Depth:
- Can be bypassed via reflection or `with` expressions.

Example:
```csharp
public class User { public string Name { get; init; } }
var u = new User { Name = "John" };
```

3. Top-level Statements

Q: What are top-level statements in C#? How do they affect application structure?

Key Points:
- Removes boilerplate for simple programs.
- Implicit `Main` generated.

Example:
```csharp
Console.WriteLine("Hello World!");
```

4. Pattern Matching Enhancements

Q: Demonstrate how relational, logical, and type patterns can be combined in a switch expression.

Example:
```csharp
int x = 42;
```

```
string category = x switch
{
< 0 => "Negative",
>= 0 and < 50 => "Small",
_ => "Large"
};
```

5. Span and Memory Safety

Q: Explain what `Span` is and why it can only be used in certain contexts.

Key Points:
- Stack-only type, avoids allocations.
- Works on contiguous memory.
- Cannot be a field in heap objects.

Example:
```csharp
Span numbers = stackalloc int[3] { 1, 2, 3 };
numbers[0] = 99;
```

6. Nullable Reference Types

Q: How does enabling nullable reference types change compilation behavior?

Key Points:
- Compiler warns when dereferencing null.
- Use `?` for nullable, `!` to suppress warnings.

Example:
```csharp
string? name = null;
// Console.WriteLine(name.Length); // Warning
```

7. Primary Constructors

Q: What are primary constructors in C# 12?

Example:
```csharp
public class Order(string id, decimal total) {
public void Print() => Console.WriteLine($"{id}: {total}");
}
```

8. Async Streams

Q: How do `IAsyncEnumerable` and `await foreach` work?

Example:
```csharp
async IAsyncEnumerable GetNumbers() {
for (int i = 0; i < 3; i++) {
await Task.Delay(500);
```

yield return i;
}
}
await foreach (var n in GetNumbers())
Console.WriteLine(n);
```

### 9. Deconstruction

Q: Show how to implement custom `Deconstruct` methods.

Example:
```csharp
public class Point {
public int X { get; }
public int Y { get; }
public Point(int x, int y) => (X, Y) = (x, y);
public void Deconstruct(out int x, out int y) => (x, y) = (X, Y);
}
var p = new Point(1, 2);
var (x, y) = p;
```

10. File-scoped Namespaces

Q: What are file-scoped namespaces?

Example:
```csharp
namespace MyApp;
public class User {}
```

11. Default Interface Methods

Q: Explain what default interface methods are.

Example:
```csharp
public interface ILogger {
void Log(string message);
void LogInfo(string message) => Log("INFO: " + message);
}
```

12. Discards

Q: What is a discard variable (`_`)?

Example:
```csharp
if (int.TryParse("123", out _)) { }
```

13. Static Abstract Interface Members

Q: What problem do static abstract interface members solve?
```

Example:
```csharp
public interface IAddable where T : IAddable {
static abstract T Add(T a, T b);
}
```

14. Target-typed new Expressions

Q: Explain what target-typed `new()` is.

Example:
```csharp
List nums = new() { 1, 2, 3 };
```

15. Ref Struct & Readonly Struct

Q: What is a `ref struct` and why can it not be boxed?

Example:
```csharp
public readonly struct Point(int x, int y);
```

16. Source Generators

Q: How do source generators work?

Key Points:
- Compile-time code generation, avoids runtime reflection.

17. Interpolated String Handlers

Q: What are interpolated string handlers?

Example:
```csharp
public void Log(LogLevel level, [InterpolatedStringHandlerArgument("level")] LogHandler handler) {}
```

18. With-expressions for Structs

Q: Can you use `with` expressions with structs?

Example:
```csharp
var p1 = new Point(1, 2);
var p2 = p1 with { X = 5 };
```

19. Required Members

Q: What are `required` members?

Example:
```csharp
public class User {
public required string Name { get; init; }
```

}
```

## 20. Collection Expressions

Q: What is the new collection expression syntax `[]` in C# 12?

Example:
```csharp
List numbers = [1, 2, 3];
```
```