## 1. Value Types vs Reference Types

Q: Explain the difference between value types and reference types in C#.

Key Points:
- Value types store data directly and are usually allocated on the stack.
- Reference types store a reference to the data, which is on the heap.
- Assignment of value types copies the data; assignment of reference types copies the reference.

Example:
```csharp
int a = 5; int b = a; b = 10; // a still 5
var arr1 = new int[] {1, 2}; var arr2 = arr1; arr2[0] = 99; // arr1[0] is now 99
```

## 2. Boxing and Unboxing

Q: What is boxing and unboxing?

Key Points:
- Boxing: converting a value type to an object.
- Unboxing: extracting the value type from the object.
- Boxing involves heap allocation and can be costly.

Example:
```csharp
int x = 42; object obj = x; // boxing
int y = (int)obj; // unboxing
```

## 3. Structs vs Classes

Q: Differences between structs and classes?

Key Points:
- Structs are value types, classes are reference types.
- Structs cannot inherit from other structs or classes.
- Structs are more lightweight.

Example:
```csharp
struct Point { public int X; public int Y; }
class Person { public string Name; }
```

## 4. Equals() vs ==

Q: Difference between Equals() and ==?

Key Points:
- `==` can be overloaded, default is reference comparison for reference types.
- `Equals()` can be overridden for custom equality.

Example:
```csharp
string a = new string("hi"); string b = new string("hi");
Console.WriteLine(a == b); // True
Console.WriteLine(a.Equals(b)); // True
```

```

## 5. Virtual, Override, New keywords

Q: Explain virtual, override, and new keywords.

Key Points:
- `virtual`: method can be overridden in derived class.
- `override`: overrides a base class virtual method.
- `new`: hides a member from the base class.

Example:
```csharp
class Base { public virtual void M(){} }
class Derived : Base { public override void M(){} }
```

## 6. Abstract Classes vs Interfaces

Q: Differences between abstract classes and interfaces.

Key Points:
- Abstract class: can have implementation and abstract members.
- Interface: all members abstract by default (before C# 8).
- Use abstract class when sharing code.

## 7. Generics

Q: What are generics and why use them?

Key Points:
- Provide type safety without casting.
- Avoid boxing for value types.
- Can use constraints: `where T : ...`.

Example:
```csharp
class Repo { public void Add(T item){} }
```

## 8. Delegates

Q: What are delegates?

Key Points:
- Type-safe function pointers.
- Can reference methods.

Example:
```csharp
delegate void MyDelegate(string msg);
MyDelegate d = Console.WriteLine;
d("Hello");
```

## 9. Events
```

Q: How do events work in C#?

Key Points:
- Special delegates with `event` keyword.
- Only the declaring class can raise the event.

Example:
```csharp
public event EventHandler MyEvent;
```

## 10. Extension Methods

Q: What are extension methods?

Key Points:
- Static methods in static classes with `this` before first parameter.

Example:
```csharp
public static class StringExt { public static bool IsEmpty(this string s) => s.Length == 0; }
```

## 11. LINQ basics

Q: How does LINQ work?

Key Points:
- Query syntax and method syntax.
- Deferred execution.

Example:
```csharp
var nums = new[] {1,2,3}; var evens = nums.Where(n => n % 2 == 0);
```

## 12. Garbage Collection

Q: How does GC work in .NET?

Key Points:
- Automatic memory management.
- Works in generations (0, 1, 2).

## 13. IDisposable and using statement

Q: When and how to implement IDisposable?

Key Points:
- Used to release unmanaged resources.
- `using` ensures Dispose() is called.

Example:
```csharp
using(var fs = new FileStream("file.txt", FileMode.Open)) { }
```

## 14. Exception Handling

Q: Best practices in exception handling.

Key Points:
- Catch only specific exceptions.
- Use finally for cleanup.

Example:
```csharp
try { } catch(IOException) { } finally { }
```

## 15. Threading basics

Q: Explain threads in .NET.

Key Points:
- Thread: separate path of execution.
- ThreadPool: reuse threads.

## 16. Task Parallel Library

Q: Benefits of TPL.

Key Points:
- Simplifies multithreading.
- Uses thread pool.

Example:
```csharp
await Task.Run(() => {});
```

## 17. lock keyword

Q: How does `lock` work?

Key Points:
- Prevents multiple threads from executing a block at same time.
Example:
```csharp
lock(obj) { }
```

## 18. Immutable Types

Q: Why use immutable types?

Key Points:
- Thread safety.
- Predictable state.

## 19. Reflection

Q: What is reflection?

Key Points:
- Inspect and modify metadata at runtime.

Example:

```csharp
var t = typeof(string);
```

## 20. Attributes

Q: What are attributes in .NET?

Key Points:
- Metadata for code.
- Can be read via reflection.

Example:
```csharp
[Obsolete] void OldMethod(){}
```