Mention what is unchecked keyword?

The unchecked keyword is used to suppress overflow-checking for integral-type arithmetic operations and conversions. If the unchecked environment is removed, a compilation error occurs. Checking for overflow takes time, the use of unchecked code in situations where there is no danger of overflow might improve performance.

The entry point in the program, where and when it exists?

Static Void Main() is the necessary entry point for any "Executable" (.EXE) to be created in C#. A library (or .DLL) could have other entry points. The reason that Main must be static is that non-static objects must be constructed before you call any methods on them.

String and StringBuilder their main differences?
A string instance is immutable. Immutable means once we create a string object we cannot modify the value of the string Object in the memory. The System.Text.StringBuilder is mutable, that means once we create StringBuilder object we can perform any operation that appears to change the value without creating a new instance for every time.

Mention what are out and ref keywords?
The out is a keyword in C# which is used for the passing the arguments to methods as a reference type. It is necessary to initialize the value of a parameter before returning to the calling method. The ref is a keyword in C# which is used for the passing the arguments by a reference. It is necessary the parameters should initialize before it passes to ref.

ValueType and ReferenceType their main differences?

A Value Type holds the data within its own memory allocation and a Reference Type contains a pointer to another memory location that holds the real data. Generally, Reference Type variables are stored in the heap while Value Type variables are stored in the stack. For example, class stands for Reference Type and a structure is a Value Type.

OOP basic principles?

Encapsulation, Inheritance, Polymorphism.
Encapsulation is achieved when each object keeps its state private, inside a class. Other objects don’t have direct access to this state. Instead, they can only call a list of public functions — called methods.

Inheritance

Objects are often very similar. They share common logic. But they're not entirely the same. It means that you create a (child) class by deriving from another (parent) class. The child class reuses all fields and methods of the parent class (common part) and can implement its own (unique part).

Polymorphism

Polymorphism gives a way to use a class exactly like its parent so there's no confusion with mixing types. But each child class keeps its own methods as they are. This typically happens by defining a (parent) interface to be reused. It outlines a bunch of common methods. Then, each child class implements its own version of these methods.

Abstract class and Interface their main differences between?

An Abstract class is never intended to be instantiated directly. This class must contain at least one abstract method, which is marked by the keyword or modifier abstract in the class definition. The Abstract classes are typically used to define a base class in the class hierarchy.

Like a class, Interface can have methods, properties, events, and indexers as its members. But interfaces will contain only the declaration of the members. The implementation of interface's members will be given by the class who implements the interface implicitly or explicitly.

ABSTRACT CLASS	INTERFACE
It contains both declaration and definition part.	It contains only a declaration part.
Multiple inheritance is not achieved by abstract class.	Multiple inheritance is achieved by interface.
It contain constructor.	It does not contain constructor.
It can contain static members.	It does not contain static members.
It can contain different types of access modifiers like public, private, protected etc.	It only contains public access modifier because everything in the interface is public.
The performance of an abstract class is fast.	The performance of interface is slow because it requires time to search actual method in the corresponding class.
It is used to implement the core identity of class.	It is used to implement peripheral abilities of class.
A class can only use one abstract class.	A class can use multiple interface.
If many implementations are of the same kind and use common behavior, then it is superior to use abstract class.	If many implementations only share methods, then it is superior to use Interface.
Abstract class can contain methods, fields, constants, etc.	Interface can only contain methods .
It can be fully, partially or not implemented.	It should be fully implemented.

Note: Junior .Net devs may answer this question partially but it will be enough for them since this question is very voluminous.

Mention what is .Net Namespaces?

Namespaces in .NET is nothing but a way to organize .NET Framework Class Library into a logical grouping according to their usability, functionality as well as a category they belong to.

Mention what is MSIL in .NET ?

- MSIL stands for Microsoft Intermediate Language
- During the compile time, the source code is converted into Microsoft Intermediate Language (MSIL) by a compiler

- MSIL is a CPU-independent set of instructions that can be efficiently converted to the native code

Mention what are the functions .NET Assembly performs?

Assembly is the main unit of deployment in a .NET Framework application executed as .exe or .dll.
An assembly performs the following functions

- It consists of an IL code that gets executed by the common language runtime
- It forms a security boundary
- By establishing name scope for types at the runtime, it ensures safety
- It carries version information
- It enables side-by-side execution of multiple versions of the same assembly
- Assembly is where permission is requested and granted.

Mention what is .Net Assembly Manifest?

.Net Assembly Manifest is a file which contains metadata about .NET Assemblies. It describes how the elements in the assembly relate to each other. In other words, it describes the relationship and dependencies of the components in the Assembly, scope information, versioning information, etc.