**Q1.**

**Assertion (A):** In C#, a `struct` can inherit from another `struct`.

**Reason (R):** In C#, a `struct` is a value type, and value types can inherit from other value types.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** In C#, a `struct` cannot inherit from another `struct`, but `structs` are indeed value types.

---

**Q2.**

**Assertion (A):** In C#, an `interface` can define static methods.

**Reason (R):** In C#, static methods are not bound to object instances and thus can be defined within an `interface`.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Since C# 8.0, interfaces can indeed define static methods.

---

**Q3.**

**Assertion (A):** In C#, the `StringBuilder` class is more efficient for modifying strings compared to the `String` class.

**Reason (R):** The `StringBuilder` class is mutable, whereas the `String` class is immutable.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** `StringBuilder` is mutable and thus more efficient for operations that modify the string multiple times.

---

**Q4.**

**Assertion (A):** In C#, the `ArrayList` class can store different types of elements.

**Reason (R):** The `ArrayList` class stores elements as objects, which means any type can be added.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** `ArrayList` stores elements as objects, which allows it to store different types.

---

**Q5.**

**Assertion (A):** In C#, the `null` keyword can be used with value types.

**Reason (R):** Value types in C# cannot hold `null` unless they are nullable.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** Value types cannot be assigned `null` unless they are nullable types (`Nullable<T>` or `T?`).

---

**Q6.**

**Assertion (A):** In C#, the `finally` block always executes, regardless of whether an exception is thrown or not.

**Reason (R):** The `finally` block is designed to execute code that must run whether an exception occurs or not.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `finally` block is meant to execute after the try-catch blocks, regardless of an exception.

---

**Q7.**

**Assertion (A):** In C#, the `var` keyword defines a variable whose type is inferred at compile time.

**Reason (R):** The `var` keyword is similar to dynamic typing, allowing variables to change type at runtime.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** `var` infers the type at compile time and cannot change type at runtime.

---

**Q8.**

**Assertion (A):** In C#, a class can implement multiple interfaces.

**Reason (R):** C# allows a class to inherit from multiple classes, which makes it flexible in design.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** C# allows a class to implement multiple interfaces, but a class cannot inherit from multiple classes (no multiple inheritance in C#).

---

**Q9.**

**Assertion (A):** In C#, the `delegate` keyword is used to define callback methods.

**Reason (R):** Delegates are type-safe function pointers in C#.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Delegates are used for defining callback methods and are type-safe.

---

**Q10.**

**Assertion (A):** In C#, the `readonly` keyword can be applied to a method.

**Reason (R):** The `readonly` keyword ensures that the field cannot be modified after initialization in the constructor.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `readonly` keyword can only be applied to fields, not methods.

**Q11.**

**Assertion (A):** In C#, a `class` marked as `sealed` cannot be inherited.

**Reason (R):** The `sealed` keyword restricts the inheritance of the class.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** A `sealed` class cannot be inherited, and the `sealed` keyword is used for this purpose.

---

**Q12.**

**Assertion (A):** In C#, an abstract class can have a constructor.

**Reason (R):** Abstract classes cannot be instantiated, so constructors are not needed.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** Abstract classes can have constructors, which are called when an instance of a derived class is created.

---

**Q13.**

**Assertion (A):** In C#, the `foreach` loop can be used to iterate through a collection of any type.

**Reason (R):** The `foreach` loop requires that the collection implements the `IEnumerable` interface.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `foreach` loop can be used with any collection that implements `IEnumerable`.

---

**Q14.**

**Assertion (A):** In C#, the `switch` statement can be used with any data type.

**Reason (R):** The `switch` statement in C# only supports primitive data types.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** The `switch` statement in C# can be used with integral types, `enum`, `char`, string, and since C# 7.0, with some custom types.

---

**Q15.**

**Assertion (A):** In C#, the `private` access modifier restricts access to the containing class only.

**Reason (R):** A member marked as `private` can be accessed by any code within the same namespace.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** The `private` access modifier restricts access to the containing class, not the entire namespace.

---

**Q16.**

**Assertion (A):** In C#, a `static` class can contain instance methods.

**Reason (R):** A `static` class is not instantiated, so it should not contain instance members.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** A `static` class cannot contain instance methods; it can only contain static members.

---

**Q17.**

**Assertion (A):** In C#, methods can have optional parameters.

**Reason (R):** Optional parameters allow the method to be called with fewer arguments than the number of parameters.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Optional parameters allow methods to be called with fewer arguments, as default values are used for omitted parameters.

---

**Q18.**

**Assertion (A):** In C#, `LINQ` queries can be executed on arrays.

**Reason (R):** `LINQ` queries are only applicable to collections that implement `IQueryable`.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** `LINQ` can be used with any collection that implements `IEnumerable`, including arrays.

---

**Q19.**

**Assertion (A):** In C#, a method marked as `virtual` must be overridden in derived classes.

**Reason (R):** The `virtual` keyword indicates that a method is intended to be overridden.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `virtual` keyword allows a method to be overridden, but it is not mandatory to override it in derived classes.

---

**Q20.**

**Assertion (A):** In C#, the `is` keyword checks for type compatibility at runtime.

**Reason (R):** The `is` keyword is used to determine if an object is an instance of a specific type or implements a specific interface.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `is` keyword is used to check type compatibility at runtime and determine if an object is of a specific type.

---

**Q21.**

**Assertion (A):** In C#, `properties` can only have a getter and no setter.

**Reason (R):** Properties in C# are used to encapsulate private fields and can be read-only.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Properties can be read-only by having only a getter, and this is a common practice to encapsulate fields.

---

**Q22.**

**Assertion (A):** In C#, the `abstract` keyword can be applied to fields.

**Reason (R):** Abstract members do not have an implementation and must be implemented by derived classes.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `abstract` keyword cannot be applied to fields; it can only be applied to methods, properties, events, and indexers.

---

**Q23.**

**Assertion (A):** In C#, the `ref` keyword is used to pass arguments by reference.

**Reason (R):** Passing by reference allows the called method to modify the argument value.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `ref` keyword allows a method to modify the value of an argument passed by reference.

---

**Q24.**

**Assertion (A):** In C#, an enum can inherit from another enum.

**Reason (R):** Enums in C# are value types and cannot participate in inheritance.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** Enums are value types and cannot inherit from another enum or any other type.

---

**Q25.**

**Assertion (A):** In C#, the `out` keyword requires that the argument be initialized before being passed to a method.

**Reason (R):** The `out` keyword is used to indicate that a method will initialize the argument.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `out` keyword allows a method to return multiple values and requires that the method initialize the argument. The argument does not need to be initialized before being passed.

---

**Q26.**

**Assertion (A):** In C#, the `delegate` keyword can be used to define an anonymous method.

**Reason (R):** Anonymous methods are a shorthand way of writing methods that are used only once.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `delegate` keyword can define anonymous methods, which are useful when the method is only needed once.

---

**Q27.**

**Assertion (A):** In C#, the `using` statement is used to manage the scope of objects.

**Reason (R):** The `using` statement ensures that `IDisposable` objects are disposed of as soon as they go out of scope.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `using` statement is designed to ensure proper resource management by automatically disposing of `IDisposable` objects.

---

**Q28.**

**Assertion (A):** In C#, the `params` keyword allows a method to accept a variable number of arguments.

**Reason (R):** The `params` keyword allows the method to be called with an array of arguments.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `params` keyword allows a method to accept a variable number of arguments, which are treated as an array.

---

**Q29.**

**Assertion (A):** In C#, the `typeof` operator can be used to obtain the `Type` object for a type.

**Reason (R):** The `typeof` operator is used to check if an object is of a particular type.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** The `typeof` operator obtains the `Type` object for a type. It does not check if an object is of a particular type.

---

**Q30.**

**Assertion (A):** In C#, a `static` method can access non-static members of the class.

**Reason (R):** Static methods belong to the class rather than to any object instance.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** A `static` method cannot access non-static members of the class because static methods are not associated with any instance.

---

**Q31.**

**Assertion (A):** In C#, the `default` keyword can be used to assign a default value to a variable.

**Reason (R):** The `default` keyword is used primarily in switch statements.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** B) Both A and R are true, but R is not the correct explanation of A.

**Explanation:** The `default` keyword can be used to assign default values and in switch statements, but these are different usages of the same keyword.

---

**Q32.**

**Assertion (A):** In C#, extension methods must be defined within the same class they are extending.

**Reason (R):** Extension methods provide a way to add methods to existing types without modifying them.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** Extension methods are defined in static classes, not in the class they extend.

---

**Q33.**

**Assertion (A):** In C#, an interface can contain fields.

**Reason (R):** An interface defines a contract for methods, properties, events, and indexers but does not define fields.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** Interfaces cannot contain fields; they only define methods, properties, events, and indexers.

---

**Q34.**

**Assertion (A):** In C#, the `try-catch` block is used to handle exceptions.

**Reason (R):** The `try` block contains the code that might throw an exception, while the `catch` block contains the code to handle it.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `try-catch` block is indeed used to handle exceptions in C#.

---

**Q35.**

**Assertion (A):** In C#, the `lock` statement is used to synchronize access to a shared resource.

**Reason (R):** The `lock` statement prevents multiple threads from accessing the same resource simultaneously.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `lock` statement is used to ensure that only one thread can access a resource at a time, providing thread safety.

---

**Q36.**

**Assertion (A):** In C#, the `String` class is a reference type

.

**Reason (R):** The `String` class is immutable, meaning its value cannot be changed after it is created.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** B) Both A and R are true, but R is not the correct explanation of A.

**Explanation:** While the `String` class is a reference type and is immutable, its immutability is not the reason it's a reference type.

---

**Q37.**

**Assertion (A):** In C#, the `throw` keyword is used to signal the occurrence of an exception.

**Reason (R):** Exceptions can be re-thrown using the `throw` keyword inside a `catch` block.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `throw` keyword is used to signal exceptions, and it can be used in a `catch` block to re-throw exceptions.

---

**Q38.**

**Assertion (A):** In C#, the `==` operator can be overloaded.

**Reason (R):** Operator overloading allows user-defined types to behave like built-in types.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `==` operator can be overloaded to provide custom equality checks for user-defined types.

---

**Q39.**

**Assertion (A):** In C#, the `continue` statement skips the current iteration of a loop and proceeds with the next iteration.

**Reason (R):** The `continue` statement is used to exit the loop immediately.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** The `continue` statement skips the remaining code in the current iteration and starts the next iteration, but it does not exit the loop.

---

**Q40.**

**Assertion (A):** In C#, a constructor can be static.

**Reason (R):** Static constructors are used to initialize static fields or perform actions that only need to be done once.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Static constructors are special constructors used to initialize static members of the class and are called only once.

---

**Q41.**

**Assertion (A):** In C#, all exceptions are derived from the `System.Exception` class.

**Reason (R):** The `System.Exception` class provides the base functionality for handling exceptions in C#.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** All exceptions in C# inherit from the `System.Exception` class, which provides the basic functionality for exception handling.

---

**Q42.**

**Assertion (A):** In C#, an `enum` can have methods.

**Reason (R):** `Enum` types in C# are treated as value types and cannot contain methods.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** In C#, `enum` can indeed have methods, although `enum` is a value type.

---

**Q43.**

**Assertion (A):** In C#, the `readonly` keyword can be applied to methods.

**Reason (R):** The `readonly` keyword ensures that a member cannot be modified after its initialization.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `readonly` keyword is used for fields, not methods, and it ensures that the field cannot be modified after its initialization.

---

**Q44.**

**Assertion (A):** In C#, a `static` constructor is called automatically before the first instance of a class is created.

**Reason (R):** Static constructors are used to initialize static members and are invoked only once.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Static constructors are automatically called once, before any instance of the class is created, to initialize static members.

---

**Q45.**

**Assertion (A):** In C#, the `volatile` keyword ensures that a field is not cached by the processor.

**Reason (R):** The `volatile` keyword ensures that the value of the field is always read from memory.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `volatile` keyword prevents the field from being cached, ensuring that the value is always read from memory.

---

**Q46.**

**Assertion (A):** In C#, the `out` keyword is used to indicate that a method will not return a value.

**Reason (R):** The `out` keyword is used for parameters that are passed by reference and are meant to be initialized within the method.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `out` keyword is used for parameters that are passed by reference and are initialized by the called method.

---

**Q47.**

**Assertion (A):** In C#, the `dynamic` keyword bypasses compile-time type checking.

**Reason (R):** The `dynamic` keyword allows operations that are resolved at runtime rather than compile-time.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `dynamic` keyword allows for runtime type checking, bypassing compile-time checks.

---

**Q48.**

**Assertion (A):** In C#, `partial` classes can be spread across multiple files.

**Reason (R):** The `partial` keyword allows a class or method to be defined in multiple files, but they are combined into a single unit at compile time.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct

explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `partial` keyword allows a class or method to be divided into multiple files, which are combined into one at compile time.

---

**Q49.**

**Assertion (A):** In C#, the `params` keyword allows an array of parameters to be passed to a method.

**Reason (R):** The `params` keyword can only be used with one parameter in a method signature, and it must be the last parameter.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `params` keyword allows passing a variable number of arguments as an array, and it must be the last parameter in the method signature.

---

**Q50.**

**Assertion (A):** In C#, a `finally` block is always executed after the try and catch blocks, whether or not an exception is thrown.

**Reason (R):** The `finally` block is used to release resources that are no longer needed after the exception handling process.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `finally` block is used to ensure that resources are released, regardless of whether an exception occurred.

---

**Q51.**

**Assertion (A):** In C#, the `this` keyword can be used in static methods.

**Reason (R):** The `this` keyword refers to the current instance of the class.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `this` keyword refers to the current instance and cannot be used in static methods because they are not associated with any instance.

---

**Q52.**

**Assertion (A):** In C#, `indexers` allow objects to be indexed like arrays.

**Reason (R):** Indexers are used to provide array-like access to the elements of a class or struct.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Indexers enable objects to be indexed like arrays, making them very useful for collections.

---

**Q53.**

**Assertion (A):** In C#, the `stackalloc` keyword is used to allocate memory on the heap.

**Reason (R):** The `stackalloc` keyword is used for creating an array on the stack instead of the heap.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `stackalloc` keyword is used to allocate memory on the stack, not the heap.

---

**Q54.**

**Assertion (A):** In C#, anonymous types allow creating objects without explicitly defining a class.

**Reason (R):** Anonymous types are mainly used for temporary data storage.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Anonymous types are used for temporary data structures, often when the structure is only needed for a short time.

---

**Q55.**

**Assertion (A):** In C#, events are used to notify clients when something of interest occurs.

**Reason (R):** Events are based on the `delegate` model and follow the publish-subscribe pattern.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Events in C# are based on the delegate model and follow the publish-subscribe pattern to notify clients when something occurs.

---

**Q56.**

**Assertion (A):** In C#, a method marked as `override` cannot override a method that is not marked as `virtual` or `abstract`.

**Reason (R):** The `override` keyword is used to extend or modify the abstract or virtual implementation of an inherited method, property, indexer, or event.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `override` keyword is specifically used to override methods that are marked as `virtual`, `abstract`, or already `override` in a base class.

---

**Q57.**

**Assertion (A):** In C#, the `readonly` keyword can be applied to properties.

**Reason (R):** The `readonly` keyword ensures that the property value cannot be modified after initialization.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `readonly` keyword cannot be applied directly to properties; however, properties can be made read-only by providing only a getter and no setter.

---

**Q58.**

**Assertion (A):** In C#, the `String` type is an alias for the `System.String` class.

**Reason (R):** The `System.String` class provides methods and properties for manipulating strings.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** `String` in C# is an alias for the `System.String` class, which provides extensive functionality for working with strings.

---

**Q59.**

**Assertion (A):** In C#, a `try` block must be followed by either a `catch` or `finally` block.

**Reason (R):** The `try` block is used to wrap code that may throw an exception, and the `catch` or `finally` blocks are used to handle or clean up after the exception.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** A `try` block in C# must be followed by either a `catch` block to handle exceptions or a `finally` block to clean up resources.

---

**Q60.**

**Assertion (A):** In C#, `Nullable<T>` can be used to represent all types, including value types and reference types.

**Reason (R):** Nullable types are used to represent value types that can be assigned a `null` value.

**Options:**

A) Both A and

R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** `Nullable<T>` is used to represent value types that can be `null`, but reference types can naturally be `null` and do not need `Nullable<T>`.

---

**Q61.**

**Assertion (A):** In C#, a method marked as `sealed` cannot be overridden by derived classes.

**Reason (R):** The `sealed` keyword is used to prevent further derivation of a class or overriding of a method.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `sealed` keyword prevents further overriding of a method in derived classes.

---

**Q62.**

**Assertion (A):** In C#, the `abstract` keyword can be used with methods and classes.

**Reason (R):** An abstract method or class cannot have an implementation and must be inherited to be used.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `abstract` keyword is used to define methods and classes that cannot be instantiated directly and must be implemented by derived classes.

---

**Q63.**

**Assertion (A):** In C#, the `var` keyword can be used to declare variables with an explicitly defined type.

**Reason (R):** The `var` keyword allows the compiler to infer the type of the variable based on the assigned value.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `var` keyword does not allow explicit type definition; the type is inferred by the compiler from the assigned value.

---

**Q64.**

**Assertion (A):** In C#, the `as` keyword is used to perform a cast that can fail without throwing an exception.

**Reason (R):** The `as` keyword returns `null` if the cast fails, instead of throwing an exception.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `as` keyword attempts to cast an object to a specific type, returning `null` if the cast fails, instead of throwing an exception.

---

**Q65.**

**Assertion (A):** In C#, the `protected internal` access modifier allows access within the same assembly or from derived classes.

**Reason (R):** The `protected internal` access modifier is a combination of `protected` and `internal` access levels.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** `protected internal` members can be accessed from within the same assembly or from derived classes, combining both access levels.

---

**Q66.**

**Assertion (A):** In C#, the `private protected` access modifier allows access from the same class and derived classes within the same assembly.

**Reason (R):** The `private protected` access modifier is more restrictive than `protected internal`.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** `private protected` allows access within the same class and derived classes in the same assembly, making it more restrictive than `protected internal`.

---

**Q67.**

**Assertion (A):** In C#, the `internal` access modifier restricts access to members within the same assembly.

**Reason (R):** The `internal` access modifier is useful for hiding members from code outside the assembly.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `internal` access modifier restricts access to the same assembly, hiding members from external code.

---

**Q68.**

**Assertion (A):** In C#, the `volatile` keyword ensures that a field's value is always updated immediately across all threads.

**Reason (R):** The `volatile` keyword prevents a field from being cached, ensuring it is always read from memory.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `volatile` keyword ensures that a field's value is always accessed directly from memory, preventing caching across threads.

---

**Q69.**

**Assertion (A):** In C#, methods with the `params` keyword can accept a variable number of arguments.

**Reason (R):** The `params` keyword allows the method to accept an array of arguments, treating them as a single collection.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `params` keyword allows a method to accept a variable number of arguments, treating them as an array.

---

**Q70.**

**Assertion (A):** In C#, a `finally` block is executed even if a `return` statement is encountered in the `try` block.

**Reason (R):** The `finally` block is guaranteed to execute regardless of how the `try` block is exited.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `finally` block is always executed, even if a `return` statement is encountered in the `try` block.

---

**Q71.**

**Assertion (A):** In C#, a `delegate` can reference multiple methods at the same time.

**Reason (R):** Delegates are type-safe function pointers that can be used for callback methods.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Delegates can reference multiple methods (multicast delegates), making them suitable for callbacks and event handling.

---

**Q72.**

**Assertion (A):** In C#, an abstract class can implement methods with a body.

**Reason (R):** Abstract classes can provide a base implementation for

methods that can be overridden by derived classes.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Abstract classes can implement methods with a body, providing a base implementation that derived classes can override.

---

**Q73.**

**Assertion (A):** In C#, a method marked as `virtual` cannot be overridden in a derived class.

**Reason (R):** The `virtual` keyword indicates that a method can be overridden in derived classes.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** The `virtual` keyword allows a method to be overridden in a derived class, providing a base method that can be extended or modified.

---

**Q74.**

**Assertion (A):** In C#, a `ref` parameter must be initialized before being passed to a method.

**Reason (R):** The `ref` keyword indicates that the parameter is passed by reference and may be modified by the called method.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** A `ref` parameter must be initialized before being passed to a method because it is passed by reference and can be modified by the method.

---

**Q75.**

**Assertion (A):** In C#, the `in` keyword is used to pass arguments by reference but ensures that the argument is not modified.

**Reason (R):** The `in` keyword allows for efficient passing of large data structures without allowing modifications to the data.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `in` keyword is used to pass arguments by reference without allowing

modifications, making it useful for passing large structures efficiently.

---

**Q76.**

**Assertion (A):** In C#, a `static` constructor is called before any instance constructors.

**Reason (R):** Static constructors are used to initialize static members and are executed only once when the class is first used.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Static constructors are called once before any instance constructors to initialize static members of the class.

---

**Q77.**

**Assertion (A):** In C#, the `dynamic` keyword allows operations to be resolved at runtime instead of compile-time.

**Reason (R):** The `dynamic` keyword bypasses compile-time type checking, enabling flexible coding practices.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `dynamic` keyword in C# defers type checking to runtime, providing flexibility but with potential runtime risks.

---

**Q78.**

**Assertion (A):** In C#, `LINQ` queries can only be used with collections that implement `IEnumerable`.

**Reason (R):** `LINQ` provides a unified syntax for querying various types of data collections in C#.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** `LINQ` queries require collections to implement `IEnumerable` because LINQ operates on sequences of elements.

---

**Q79.**

**Assertion (A):** In C#, the `extern` keyword is used to declare a method that is implemented outside of C#.

**Reason (R):** The `extern` keyword is used in conjunction with `DllImport` to call functions from unmanaged code.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `extern` keyword, often used with `DllImport`, allows C# to call functions implemented in unmanaged code, such as those in DLLs.

---

**Q80.**

**Assertion (A):** In C#, the `volatile` keyword is used to ensure that a variable's value is always read from the memory location, not from a CPU cache.

**Reason (R):** The `volatile` keyword ensures proper synchronization across multiple threads when accessing shared variables.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `volatile` keyword prevents the variable from being cached, ensuring that it is always read from memory, which is important for multi-threaded synchronization.

---

**Q81.**

**Assertion (A):** In C#, the `params` keyword allows for a variable number of parameters in a method.

**Reason (R):** The `params` keyword must be the last parameter in the method signature.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `params` keyword allows for a variable number of parameters and must be the last parameter in the method signature.

---

**Q82.**

**Assertion (A):** In C#, a `static` constructor is called automatically before any static members are accessed.

**Reason (R):** Static constructors are used to initialize static members of a class and are executed only once.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** Static constructors are called automatically before any static members are accessed to initialize them.

---

**Q83.**

**Assertion (A):** In C#, the `ref` keyword allows a method to return multiple values by modifying arguments passed to it.

**Reason (R):** The `ref` keyword passes arguments by reference, allowing the method to modify their values.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `ref` keyword allows a method to modify the value of arguments passed by reference, effectively returning multiple values.

---

**Q84.**

**Assertion (A):** In C#, the `yield` keyword is used to

return each element of a collection one at a time.

**Reason (R):** The `yield` keyword is used in iterator methods to create an enumerable sequence of values.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `yield` keyword simplifies the creation of iterators, returning elements one at a time.

---

**Q85.**

**Assertion (A):** In C#, a `switch` statement can only evaluate integral and enum types.

**Reason (R):** The `switch` statement is limited to evaluating discrete values, such as integers and enumerations.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** D) A is false, but R is true.

**Explanation:** While the `switch` statement can evaluate discrete values, such as integral and enum types, it can also evaluate strings and since C# 7.0, even some custom types.

---

**Q86.**

**Assertion (A):** In C#, the `lock` statement is used to ensure that a block of code is executed by only one thread at a time.

**Reason (R):** The `lock` statement is used to prevent race conditions by providing thread synchronization.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `lock` statement ensures that a block of code is executed by only one thread at a time, preventing race conditions.

---

**Q87.**

**Assertion (A):** In C#, a method marked with `async` can be executed synchronously.

**Reason (R):** The `async` keyword is used to indicate that a method may contain asynchronous operations.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** C) A is true, but R is false.

**Explanation:** While an `async` method can be executed synchronously (if awaited tasks are already completed), the `async` keyword does not indicate that the method will always run asynchronously.

---

**Q88.**

**Assertion (A):** In C#, the `Nullable<T>` type allows value types to be assigned `null`.

**Reason (R):** The `Nullable<T>` type is used to represent undefined or absent values for value types.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** `Nullable<T>` allows value types to represent `null` values, which is useful for indicating undefined or absent data.

---

**Q89.**

**Assertion (A):** In C#, the `where` keyword in generic constraints specifies that a generic type must be a reference type.

**Reason (R):** The `where` keyword is used to apply constraints to generic type parameters.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** B) Both A and R are true, but R is not the correct explanation of A.

**Explanation:** The `where` keyword is indeed used for generic constraints, but it can be used to specify

various constraints, not just that a type must be a reference type.

---

**Q90.**

**Assertion (A):** In C#, a `finally` block will not execute if the process terminates abruptly.

**Reason (R):** The `finally` block is intended to execute regardless of whether an exception was thrown, but it will not run if the process terminates unexpectedly.

**Options:**

A) Both A and R are true, and R is the correct explanation of A.

B) Both A and R are true, but R is not the correct explanation of A.

C) A is true, but R is false.

D) A is false, but R is true.

**Answer:** A) Both A and R are true, and R is the correct explanation of A.

**Explanation:** The `finally` block is meant to run regardless of an exception, but it will not execute if the process terminates abruptly (e.g., if the application crashes or the environment is forcibly terminated).