Matching Type Questions

Set A: Terms (Match with **Set B: Descriptions**)

Set A (Terms)	**Set B (Descriptions)**
1. **Nullable types**	A. A type that provides a mechanism to encapsulate methods and properties to function like primitive types.
2. **Polymorphism**	B. A feature allowing a derived class to provide a specific implementation of a method defined in the base class.
3. **Indexer**	C. A type used to assign `null` to value types or use Nullable operators to test the value of value types.
4. **Delegates**	D. A special method that allows objects to be indexed in a similar way as arrays.
5. **LINQ**	E. A data query language integrated into C# to filter, order, and manipulate data collections.
6. **Extension Methods**	F. A feature that allows methods to be added to existing types without modifying the type.
7. **Covariance**	G. A mechanism that allows derived classes to be substituted for base classes.
8. **Sealed Class**	H. A class that cannot be inherited.
9. **Anonymous Types**	I. A way to define a class in the method body without explicitly declaring it.
10. **Auto-Implemented Properties**	J. A feature that simplifies property declarations when no additional logic is needed in the property accessors.

Answers:

1. **C** (Nullable types - A type used to assign `null` to value types or use Nullable operators to test the value of value types.)

2. **B** (Polymorphism - A feature allowing a derived class to provide a specific implementation of a method defined in the base class.)

3. **D** (Indexer - A special method that allows objects to be indexed in a similar way as arrays.)

4. **A** (Delegates - A type that provides a mechanism to encapsulate methods and properties to function like primitive types.)

5. **E** (LINQ - A data query language integrated into C# to filter, order, and manipulate data collections.)

6. **F** (Extension Methods - A feature that allows methods to be added to existing types without modifying the type.)

7. **G** (Covariance - A mechanism that allows derived classes to be substituted for base classes.)

8. **H** (Sealed Class - A class that cannot be inherited.)

9. **I** (Anonymous Types - A way to define a class in the method body without explicitly declaring it.)

10. **J** (Auto-Implemented Properties - A feature that simplifies property declarations when no additional logic is needed in the property accessors.)

Matching Type Questions

Set A: Terms (Match with **Set B: Descriptions**)

Set A (Terms)	**Set B (Descriptions)**
11. **Boxing**	K. The process of converting a value type to an object type.
12. **Unboxing**	L. The process of converting an object type back to a value type.
13. **Abstract Class**	M. A class that cannot be instantiated and is meant to be a base class for other classes.
14. **Interface**	N. A contract that defines a set of methods and properties that a class must implement.
15. **Partial Class**	O. A class whose definition is split across multiple files.
16. **Generics**	P. A feature that allows classes, methods, and delegates to be defined with a placeholder for data types.
17. **Destructor**	Q. A special method used to clean up unmanaged resources before an object is garbage-collected.
18. **Static Constructor**	R. A constructor used to initialize static members of a class.

| 19. **Enum** | S. A distinct type that consists of a set of named constants called the enumerator list. |

| 20. **Struct** | T. A value type that is typically used to encapsulate small groups of related variables. |

| 21. **Namespace** | U. A logical grouping of classes, interfaces, structs, and other types to avoid naming conflicts. |

| 22. **Reflection** | V. A feature that allows the inspection of metadata about types at runtime. |

| 23. **Attributes** | W. A way to add declarative information to code elements, such as classes, methods, or properties. |

| 24. **Events** | X. A messaging system that allows a class or object to notify other classes or objects when something of interest occurs. |

| 25. **Thread** | Y. A basic unit of execution in a program, allowing multiple operations to run concurrently. |

| 26. **Exception Handling** | Z. A mechanism to handle runtime errors and ensure the program continues running. |

| 27. **Finalizer** | AA. A method called by the garbage collector when an object is no longer in use, just before memory is reclaimed. |

| 28. **IndexOutOfRangeException** | BB. An exception that occurs when trying to access an element in an array or collection that does not exist. |

| 29. **Delegate** | CC. A type that references a method and can be used to pass methods as arguments or define callback methods. |

| 30. **Type Inference** | DD. The ability of the compiler to determine the type of a variable based on the value assigned to it. |

| 31. **Nullable Contexts** | EE. A feature that allows developers to control the nullability of reference types to help avoid null reference errors. |

| 32. **Tuples** | FF. A data structure that allows you to store a fixed-size collection of items, possibly of different types. |

| 33. **Garbage Collection** | GG. The process of automatically reclaiming memory that is no longer in use by the program. |

| 34. **Yield Keyword** | HH. A keyword used to create an iterator method that returns each element of a collection one at a time. |

| 35. **Unsafe Code** | II. A context in which pointers can be used directly in C#. |

| 36. **Event Handling** | JJ. The process of managing and responding to events, typically using delegates in C#. |

| 37. **Anonymous Methods** | KK. Methods that are defined without a name, usually for the purpose of being passed as arguments to delegates or events. |

| 38. **Func Delegate** | LL. A delegate that points to a method returning a value and can take up to 16 parameters. |

| 39. **Action Delegate** | MM. A delegate that points to a method that does not return a value and can take up to 16 parameters. |

| 40. **Predicate Delegate** | NN. A delegate that points to a method that returns a boolean value and takes a single parameter. |

| 41. **Task Parallel Library (TPL)**| OO. A set of public types and APIs to simplify parallelism and concurrency in C#. |

| 42. **Async and Await** | PP. Keywords used to simplify asynchronous programming by allowing the code to run asynchronously. |

| 43. **Memory Leak** | QQ. A situation where memory that is no longer needed is not released, causing an application to consume more and more memory over time. |

| 44. **Named Parameters** | RR. A feature that allows arguments to be passed to a method by specifying the parameter name along with its value. |

| 45. **Optional Parameters** | SS. Parameters that have default values and do not need to be supplied by the caller if the default value is acceptable. |

| 46. **Object Initializer** | TT. A syntax feature that allows you to initialize an object's properties at the time of creation without calling a constructor. |

| 47. **Method Overloading** | UU. The process of defining multiple methods with the same name but different parameters in the same class. |

| 48. **Property** | VV. A member that provides a flexible mechanism to read, write, or compute the value of a private field. |

| 49. **Lambda Expressions** | WW. An anonymous function that can contain expressions and statements, used to create delegates or expression tree types. |

| 50. **Thread Pool** | XX. A pool of worker threads managed by the runtime, which can be used to execute tasks asynchronously. |

Answers:

11. **K** (Boxing - The process of converting a value type to an object type.)

12. **L** (Unboxing - The process of converting an object type back to a value type.)

13. **M** (Abstract Class - A class that cannot be instantiated and is meant to be a base class for other classes.)

14. **N** (Interface - A contract that defines a set of methods and properties that a class must implement.)

15. **O** (Partial Class - A class whose definition is split across multiple files.)

16. **P** (Generics - A feature that allows classes, methods, and delegates to be defined with a placeholder for data types.)

17. **Q** (Destructor - A special method used to clean up unmanaged resources before an object is garbage-collected.)

18. **R** (Static Constructor - A constructor used to initialize static members of a class.)

19. **S** (Enum - A distinct type that consists of a set of named constants called the enumerator list.)

20. **T** (Struct - A value type that is typically used to encapsulate small groups of related variables.)

21. **U** (Namespace - A logical grouping of classes, interfaces, structs, and other types to avoid naming conflicts.)

22. **V** (Reflection - A feature that allows the inspection of metadata about types at runtime.)

23. **W** (Attributes - A way to add declarative information to code elements, such as classes, methods, or properties.)

24. **X** (Events - A messaging system that allows a class or object to notify other classes or objects when something of interest occurs.)

25. **Y** (Thread - A basic unit of execution in a program, allowing multiple operations to run concurrently.)

26. **Z** (Exception Handling - A mechanism to handle runtime errors and ensure the program continues running.)

27. **AA** (Finalizer - A method called by the garbage collector when an object is no longer in use, just before memory is reclaimed.)

28. **BB** (IndexOutOfRangeException - An exception that occurs when trying to access an element in an array or collection that does not exist.)

29. **CC** (Delegate - A type that references a method and can be used to pass methods as arguments or define callback methods.)

30. **DD** (Type Inference - The ability of the compiler to determine the type of a variable based on the value assigned to it.)

31. **EE** (Nullable Contexts - A feature that allows developers to control the nullability of reference types to help avoid null reference errors.)

32. **FF** (Tuples - A data structure that allows you to store a fixed-size collection of items, possibly of different types.)

33. **GG** (Garbage Collection - The process of automatically reclaiming memory that is no longer in use by the program.)

34. **HH** (Yield Keyword - A keyword used to create an iterator method that returns each element of a collection one at a time.)

35. **II** (Unsafe Code - A context in which pointers can be used directly in C#.)

36. **JJ** (Event Handling - The process of managing and responding to events, typically using delegates in C#.)

37. **KK** (Anonymous Methods - Methods that are defined without a name, usually for the purpose of being passed as arguments to delegates or events.)

38. **LL** (Func Delegate - A delegate that points to a method returning a value and can take up to 16 parameters.)

39. **MM** (Action Delegate - A delegate that points to a method that does not return a value and can take up to 16 parameters.)

40. **NN** (Predicate Delegate - A delegate that points to a method that returns a boolean value and takes a single parameter.)

41. **OO** (Task Parallel Library (TPL) - A set of public types and APIs to simplify parallelism and concurrency in C#.)

42. **PP** (Async and Await - Keywords used to simplify asynchronous programming by allowing the code to run asynchronously.)

43. **QQ** (Memory Leak - A situation where memory that is no longer needed is not released, causing an application to consume more and more memory over time.)

44. **RR** (Named Parameters - A feature that allows arguments to be passed to a method by specifying the parameter name along with its value.)

45. **SS** (Optional Parameters - Parameters that have default values and do not need to be supplied by the caller if the default value is acceptable.)

46. **TT** (Object Initializer - A syntax feature that allows you to initialize an object's properties at the time of creation without calling a constructor.)

47. **UU** (Method Overloading - The process of defining multiple methods with the same name but different parameters in the same class.)

48. **VV** (Property - A member that provides a flexible mechanism to read, write, or compute the value of a private field.)

49. **WW** (Lambda Expressions - An anonymous function that can contain expressions and statements, used to create delegates or expression tree types.)

50. **XX** (Thread Pool - A pool of worker threads managed by the runtime, which can be used to execute tasks asynchronously.)