1. \*\*What is the purpose of the `ref` locals and returns feature introduced in C# 7.0?\*\*

- a) To pass arguments by value.

- b) To create nullable value types.

- c) To allow reference to the original data.

- d) To pass methods as arguments.

- \*\*Answer:\*\* c) To allow reference to the original data.

2. \*\*Which of the following statements correctly defines a local function in C# 7.0?\*\*

- a) `void LocalFunc() { }`

- b) `public void LocalFunc() { }`

- c) `private void LocalFunc() { }`

- d) `protected void LocalFunc() { }`

- \*\*Answer:\*\* a) `void LocalFunc() { }`

3. \*\*What is the default value of a tuple element when using C# 7.0's value tuples?\*\*

- a) Null

- b) 0

- c) Undefined

- d) It must be explicitly initialized.

- \*\*Answer:\*\* b) 0

4. \*\*Which of the following is NOT a new feature introduced in C# 7.0?\*\*

- a) Tuples

- b) Pattern Matching

- c) Async Main

- d) Switch expressions

- \*\*Answer:\*\* d) Switch expressions

5. \*\*In C# 7.0, what is the correct way to deconstruct a tuple into individual variables?\*\*

- a) `(var x, var y) = Tuple.Create(1, 2);`

- b) `Tuple.Create(var x, var y) = (1, 2);`

- c) `var (x, y) = Tuple.Create(1, 2);`

- d) `var x, y = Tuple.Create(1, 2);`

- \*\*Answer:\*\* c) `var (x, y) = Tuple.Create(1, 2);`

6. \*\*Which keyword is used to return a reference to a value in C# 7.0?\*\*

- a) `ref`

- b) `out`

- c) `in`

- d) `ref readonly`

- \*\*Answer:\*\* a) `ref`

7. \*\*What is the purpose of the `is` pattern matching in C# 7.0?\*\*

- a) To match strings.

- b) To check if an object is of a specific type.

- c) To perform mathematical operations.

- d) To iterate through collections.

- \*\*Answer:\*\* b) To check if an object is of a specific type.

8. \*\*Which of the following statements is correct about C# 7.0's `out` variable declaration?\*\*

- a) `out` variables must be declared before the method call.

- b) `out` variables can be declared inline within the method call.

- c) `out` variables cannot be used in C# 7.0.

- d) `out` variables must be initialized before passing.

- \*\*Answer:\*\* b) `out` variables can be declared inline within the method call.

9. \*\*What is the correct way to use a literal pattern in a `switch` statement in C# 7.0?\*\*

- a) `case 5:`

- b) `case int x when x == 5:`

- c) `case 5 is int:`

- d) `case var x when x == 5:`

- \*\*Answer:\*\* a) `case 5:`

10. \*\*Which feature of C# 7.0 allows multiple returns from a method?\*\*

- a) Value tuples

- b) Local functions

- c) Ref locals

- d) Expression-bodied members

- \*\*Answer:\*\* a) Value tuples

11. \*\*Which of the following correctly declares a `readonly` reference in C# 7.0?\*\*

- a) `ref readonly int x = ref GetValue();`

- b) `readonly ref int x = GetValue();`

- c) `int readonly ref x = GetValue();`

- d) `readonly int x = GetValue();`

- \*\*Answer:\*\* a) `ref readonly int x = ref GetValue();`

12. \*\*In C# 7.0, what is the correct way to write a binary literal for the number 13?\*\*

- a) `0b1101`

- b) `0b1011`

- c) `0x1101`

- d) `0x0D`

- \*\*Answer:\*\* a) `0b1101`

13. \*\*What is the output of the following C# 7.0 code snippet?\*\*

```csharp

var tuple = (x: 3, y: 4);

Console.WriteLine(tuple.x);

```

- a) 3

- b) 4

- c) (3, 4)

- d) Compile-time error

- \*\*Answer:\*\* a) 3

14. \*\*Which of the following allows pattern matching on the type of an object in C# 7.0?\*\*

- a) `is`

- b) `as`

- c) `switch`

- d) `typeof`

- \*\*Answer:\*\* a) `is`

15. \*\*How do you create a named tuple with two elements in C# 7.0?\*\*

- a) `(int x, int y) tuple = (1, 2);`

- b) `var tuple = (int x, int y) = (1, 2);`

- c) `var tuple = (1, 2);`

- d) `var tuple = (x: 1, y: 2);`

- \*\*Answer:\*\* d) `var tuple = (x: 1, y: 2);`

16. \*\*Which of the following correctly describes the `default` literal in C# 7.0?\*\*

- a) It sets a variable to zero.

- b) It sets a variable to its default value.

- c) It assigns null to reference types.

- d) It is not a valid literal in C#.

- \*\*Answer:\*\* b) It sets a variable to its default value.

17. \*\*In C# 7.0, how can you discard a value using pattern matching?\*\*

- a) Use the `discard` keyword.

- b) Use an underscore `\_`.

- c) Use `null`.

- d) Use `default`.

- \*\*Answer:\*\* b) Use an underscore `\_`.

18. \*\*What is the correct way to declare a binary literal for the number 10 in C# 7.0?\*\*

- a) `0b1010`

- b) `0b0101`

- c) `0b1001`

- d) `0b0011`

- \*\*Answer:\*\* a) `0b1010`

19. \*\*Which of the following is a feature of C# 7.0 allowing more concise code with lambda expressions?\*\*

- a) Expression-bodied constructors

- b) Inline functions

- c) Multi-line lambdas

- d) Statement-bodied lambdas

- \*\*Answer:\*\* a) Expression-bodied constructors

20. \*\*Which of the following is a valid example of inline `out` variable declaration in C# 7.0?\*\*

- a) `int result; if (int.TryParse("123", out result)) { }`

- b) `if (int.TryParse("123", out int result)) { }`

- c) `if (int.TryParse("123", out var result)) { }`

- d) `int result = 0; if (int.TryParse("123", result)) { }`

- \*\*Answer:\*\* b) `if (int.TryParse("123", out int result)) { }`

21. \*\*Which of the following correctly demonstrates a use of tuples in C# 7.0?\*\*

- a) `var tuple = (1, "string");`

- b) `Tuple<int, string> tuple = new Tuple<int, string>(1, "string");`

- c) `Tuple tuple = (1, "string");`

- d) `tuple = (1, "string");`

- \*\*Answer:\*\* a) `var tuple = (1, "string");`

22. \*\*What will the following code output in C# 7.0?\*\*

```csharp

(int x, int y) = (10, 20);

Console.WriteLine(x);

```

- a) 10

- b) 20

- c) (10, 20)

- d) Compile-time error

- \*\*Answer:\*\* a) 10

23. \*\*Which of the following is true about `out` variables in C# 7.0?\*\*

- a) They must be declared before use.

- b) They can be declared inline within method calls.

- c) They cannot be initialized before passing.

- d) They must be declared as `ref`.

- \*\*Answer:\*\* b) They can be declared inline within method calls.

24. \*\*In C# 7.0, how would you indicate that a method returns by reference?\*\*

- a) `ref int MethodName() { }`

- b) `int ref MethodName() { }`

- c) `ref int MethodName(int x) { return ref x; }`

- d) `int MethodName(ref x) { return x; }`

- \*\*Answer:\*\* c) `ref int MethodName(int x) { return ref x; }`

25. \*\*Which of the following correctly defines a tuple with named elements in C# 7.0?\*\*

- a) `var person = (name: "John", age: 30);`

- b) `var person = (string name = "John", int age = 30);`

- c) `var person = ("John", 30);`

- d) `var person = Tuple.Create("John", 30);`

- \*\*Answer:\*\* a) `var person = (name: "John", age: 30);`

26. \*\*In C# 7.0, which feature is used to match a value based on its type?\*\*

- a) Type pattern matching

- b) Type inference

- c) Type safety

- d) Type casting

- \*\*Answer:\*\* a) Type pattern matching

27. \*\*What is the new syntax introduced in C# 7.0 for out variables?\*\*

- a) `out var x`

- b) `out int x`

- c) `out readonly x`

- d) `out ref x`

- \*\*Answer:\*\* b) `out int x`

28. \*\*Which of the following C# 7.0 features helps in writing safer code by enabling pattern matching?\*\*

- a) `switch` expressions

- b) Type patterns

- c) LINQ expressions

- d) Indexers

- \*\*Answer:\*\* b) Type patterns

29. \*\*Which of the following is the correct syntax for a deconstructing assignment in C# 7.0?\*\*

- a) `var (x, y) = (10, 20);`

- b) `(var x, var y) = (10, 20);`

- c) `(int x, int y) = Tuple.Create(10, 20);`

- d) `(int x, y) = (10, 20);`

- \*\*Answer:\*\* a) `var (x, y) = (10, 20);`

30. \*\*Which of the following is a feature that allows in-line declaration of out variables in C# 7.0?\*\*

- a) Inline `out`

- b) Local `out`

- c) `out` variable declaration

- d) `out` in-line variable

- \*\*Answer:\*\* c) `out` variable declaration

31. \*\*In C# 7.0, how would you write a switch statement to match an object type?\*\*

- a) `switch(obj) { case int i: ... }`

- b) `switch(obj) { case is int: ... }`

- c) `switch(obj) { case var i: ... }`

- d) `switch(obj) { case int when obj: ... }`

- \*\*Answer:\*\* a) `switch(obj) { case int i: ... }`

32. \*\*What is the output of the following C# 7.0 code?\*\*

```csharp

int? x = null;

int y = x ?? 0;

Console.WriteLine(y);

```

- a) 0

- b) null

- c) Compile-time error

- d) Exception at runtime

- \*\*Answer:\*\* a) 0

33. \*\*What does the `ref` modifier in C# 7.0 indicate when used with method return types?\*\*

- a) The method returns a reference to the original data.

- b) The method passes arguments by reference.

- c) The method returns a copy of the data.

- d) The method is a reference method.

- \*\*Answer:\*\* a) The method returns a reference to the original data.

34. \*\*Which of the following is true about expression-bodied members in C# 7.0?\*\*

- a) They are restricted to single-line statements.

- b) They can only be used with methods.

- c) They are used to shorten method bodies.

- d) They are a feature of C# 6.0, not C# 7.0.

- \*\*Answer:\*\* c) They are used to shorten method bodies.

35. \*\*In C# 7.0, which of the following keywords can be used with local functions?\*\*

- a) `static`

- b) `async`

- c) `private`

- d) `protected`

- \*\*Answer:\*\* b) `async`

36. \*\*Which of the following correctly describes a C# 7.0 tuple with named fields?\*\*

- a) `(int a, int b) = (1, 2);`

- b) `(a: 1, b: 2)`

- c) `var tuple = (1, 2);`

- d) `var tuple = (a: 1, b: 2);`

- \*\*Answer:\*\* d) `var tuple = (a: 1, b: 2);`

37. \*\*In C# 7.0, how do you represent a hexadecimal literal?\*\*

- a) `0x10`

- b) `0b10`

- c) `0h10`

- d) `0x010`

- \*\*Answer:\*\* a) `0x10`

38. \*\*Which of the following is a correct use of the discard `\_` in C# 7.0?\*\*

- a) `var (x, \_) = (10, 20);`

- b) `var \_ = 10;`

- c) `var (\_, x) = (10, 20);`

- d) `var x = 10, \_ = 20;`

- \*\*Answer:\*\* a) `var (x, \_) = (10, 20);`

39. \*\*What does the `default` literal represent in C# 7.0?\*\*

- a) The default value of a type

- b) A new instance of a class

- c) A null value

- d) A constant value

- \*\*Answer:\*\* a) The default value of a type

40. \*\*Which of the following allows multiple return values from a method in C# 7.0?\*\*

- a) Value tuples

- b) Local functions

- c) Out parameters

- d) Anonymous types

- \*\*Answer:\*\* a) Value tuples

41. \*\*Which of the following correctly demonstrates a pattern matching with `is` in C# 7.0?\*\*

- a) `if (obj is int x) { Console.WriteLine(x); }`

- b) `if (obj is int) { Console.WriteLine(obj); }`

- c) `if (obj == int x) { Console.WriteLine(x); }`

- d) `if (obj == int) { Console.WriteLine(obj); }`

- \*\*Answer:\*\* a) `if (obj is int x) { Console.WriteLine(x); }`

42. \*\*Which of the following can be used as a `case` label in a `switch` statement in C# 7.0?\*\*

- a) An integer value

- b) A string value

- c) A type pattern

- d) All of the above

- \*\*Answer:\*\* d) All of the above

43. \*\*What is the correct way to write an out variable in C# 7.0?\*\*

- a) `out var x`

- b) `out int x`

- c) `out readonly x`

- d) `out ref x`

- \*\*Answer:\*\* b) `out int x`

44. \*\*Which of the following C# 7.0 features allows returning multiple values from a method?\*\*

- a) Value tuples

- b) Local functions

- c) Out parameters

- d) Anonymous types

- \*\*Answer:\*\* a) Value tuples

45. \*\*Which of the following allows for more expressive switch statements in C# 7.0?\*\*

- a) Type pattern matching

- b) Value tuples

- c) Expression-bodied members

- d) Local functions

- \*\*Answer:\*\* a) Type pattern matching

46. \*\*Which of the following is an example of using an inline `out` variable in C# 7.0?\*\*

- a) `if (int.TryParse("123", out int result)) { }`

- b) `if (int.TryParse("123", out var result)) { }`

- c) `if (int.TryParse("123", out result)) { }`

- d) `if (int.TryParse("123", out string result)) { }`

- \*\*Answer:\*\* a) `if (int.TryParse("123",

out int result)) { }`

47. \*\*Which feature of C# 7.0 simplifies code by reducing the need for temporary variables?\*\*

- a) Out variable declaration

- b) Local functions

- c) Expression-bodied members

- d) Value tuples

- \*\*Answer:\*\* a) Out variable declaration

48. \*\*Which of the following C# 7.0 features helps in writing cleaner code by allowing the use of inline `out` variables?\*\*

- a) Expression-bodied members

- b) Local functions

- c) Out variable declaration

- d) Value tuples

- \*\*Answer:\*\* c) Out variable declaration

49. \*\*Which of the following is a correct example of using a `ref` return in C# 7.0?\*\*

- a) `ref int GetRef() { return ref myInt; }`

- b) `int ref GetRef() { return myInt; }`

- c) `ref GetRef() { return myInt; }`

- d) `ref int GetRef() => myInt;`

- \*\*Answer:\*\* a) `ref int GetRef() { return ref myInt; }`

50. \*\*Which of the following C# 7.0 features allows you to discard values in a tuple deconstruction?\*\*

- a) Discards

- b) Local functions

- c) Expression-bodied members

- d) Inline `out` variables

- \*\*Answer:\*\* a) Discards

51. \*\*Which of the following C# 7.0 features allows you to define a method inside another method?\*\*

- a) Local functions

- b) Inline functions

- c) Nested methods

- d) Lambda expressions

- \*\*Answer:\*\* a) Local functions

52. \*\*Which of the following is true about `is` pattern matching in C# 7.0?\*\*

- a) It can match both type and value.

- b) It can only match types.

- c) It is used only in switch statements.

- d) It cannot be used in if statements.

- \*\*Answer:\*\* a) It can match both type and value.

53. \*\*Which of the following C# 7.0 features allows you to return multiple values from a method?\*\*

- a) Value tuples

- b) Anonymous types

- c) Tuple types

- d) Local functions

- \*\*Answer:\*\* a) Value tuples

54. \*\*Which of the following is a correct way to use the `default` literal in C# 7.0?\*\*

- a) `int x = default;`

- b) `int x = 0;`

- c) `int x = null;`

- d) `int x = new int();`

- \*\*Answer:\*\* a) `int x = default;`

55. \*\*Which of the following correctly uses pattern matching in a switch statement in C# 7.0?\*\*

- a) `switch (x) { case int n: ... }`

- b) `switch (x) { case var n: ... }`

- c) `switch (x) { case n when n == 1: ... }`

- d) `switch (x) { case default: ... }`

- \*\*Answer:\*\* a) `switch (x) { case int n: ... }`

56. \*\*Which of the following C# 7.0 features helps in writing cleaner code by allowing the use of inline `out` variables?\*\*

- a) Expression-bodied members

- b) Local functions

- c) Out variable declaration

- d) Value tuples

- \*\*Answer:\*\* c) Out variable declaration

57. \*\*Which of the following is a correct example of using an inline `out` variable in C# 7.0?\*\*

- a) `if (int.TryParse("123", out int result)) { }`

- b) `if (int.TryParse("123", out var result)) { }`

- c) `if (int.TryParse("123", out result)) { }`

- d) `if (int.TryParse("123", out string result)) { }`

- \*\*Answer:\*\* a) `if (int.TryParse("123", out int result)) { }`

58. \*\*Which of the following C# 7.0 features helps in writing more concise code with lambdas?\*\*

- a) Expression-bodied members

- b) Local functions

- c) Value tuples

- d) Inline `out` variables

- \*\*Answer:\*\* a) Expression-bodied members

59. \*\*Which of the following C# 7.0 features allows for multiple return values from a method?\*\*

- a) Value tuples

- b) Anonymous types

- c) Ref locals

- d) Expression-bodied members

- \*\*Answer:\*\* a) Value tuples

60. \*\*Which of the following is a correct example of using a `ref` local in C# 7.0?\*\*

- a) `ref int x = ref array[0];`

- b) `int ref x = ref array[0];`

- c) `ref int x = array[0];`

- d) `int x = ref array[0];`

- \*\*Answer:\*\* a) `ref int x = ref array[0];`

61. \*\*Which of the following C# 7.0 features allows you to discard values in a tuple deconstruction?\*\*

- a) Discards

- b) Local functions

- c) Expression-bodied members

- d) Inline `out` variables

- \*\*Answer:\*\* a) Discards

62. \*\*Which of the following C# 7.0 features helps in writing more expressive code with pattern matching?\*\*

- a) Type pattern matching

- b) Value tuples

- c) Expression-bodied members

- d) Local functions

- \*\*Answer:\*\* a) Type pattern matching

63. \*\*Which of the following C# 7.0 features allows you to define a method inside another method?\*\*

- a) Local functions

- b) Inline functions

- c) Nested methods

- d) Lambda expressions

- \*\*Answer:\*\* a) Local functions

64. \*\*Which of the following is true about `is` pattern matching in C# 7.0?\*\*

- a) It can match both type and value.

- b) It can only match types.

- c) It is used only in switch statements.

- d) It cannot be used in if statements.

- \*\*Answer:\*\* a) It can match both type and value.

65. \*\*Which of the following C# 7.0 features allows you to return multiple values from a method?\*\*

- a) Value tuples

- b) Anonymous types

- c) Tuple types

- d) Local functions

- \*\*Answer:\*\* a) Value tuples

66. \*\*Which of the following is a correct way to use the `default` literal in C# 7.0?\*\*

- a) `int x = default;`

- b) `int x = 0;`

- c) `int x = null;`

- d) `int x = new int();`

- \*\*Answer:\*\* a) `int x = default;`

67. \*\*Which of the following correctly uses pattern matching in a switch statement in C# 7.0?\*\*

- a) `switch (x) { case int n: ... }`

- b) `switch (x) { case var n: ... }`

- c) `switch (x) { case n when n == 1: ... }`

- d) `switch (x) { case default: ... }`

- \*\*Answer:\*\* a) `switch (x) { case int n: ... }`

68. \*\*Which of the following C# 7.0 features allows for inline declaration of out variables?\*\*

- a) Out variable declaration

- b) Local functions

- c) Expression-bodied members

- d) Value tuples

- \*\*Answer:\*\* a) Out variable declaration

69. \*\*Which of the following is a correct example of using an inline `out` variable in C# 7.0?\*\*

- a) `if (int.TryParse("123", out int result)) { }`

- b) `if (int.TryParse("123", out var result)) { }`

- c) `if (int.TryParse("123", out result)) { }`

- d) `if (int.TryParse("123", out string result)) { }`

- \*\*Answer:\*\* a) `if (int.TryParse("123", out int result)) { }`

70. \*\*Which of the following C# 7.0 features helps in writing more concise code with lambdas?\*\*

- a) Expression-bodied members

- b) Local functions

- c) Value tuples

- d) Inline `out` variables

- \*\*Answer:\*\* a) Expression-bodied members

71. \*\*Which of the following C# 7.0 features allows for multiple return values from a method?\*\*

- a) Value tuples

- b) Anonymous types

- c) Ref locals

- d) Expression-bodied members

- \*\*Answer:\*\* a) Value tuples

72.

\*\*Which of the following is a correct example of using a `ref` local in C# 7.0?\*\*

- a) `ref int x = ref array[0];`

- b) `int ref x = ref array[0];`

- c) `ref int x = array[0];`

- d) `int x = ref array[0];`

- \*\*Answer:\*\* a) `ref int x = ref array[0];`

73. \*\*Which of the following C# 7.0 features allows you to discard values in a tuple deconstruction?\*\*

- a) Discards

- b) Local functions

- c) Expression-bodied members

- d) Inline `out` variables

- \*\*Answer:\*\* a) Discards

74. \*\*Which of the following C# 7.0 features helps in writing more expressive code with pattern matching?\*\*

- a) Type pattern matching

- b) Value tuples

- c) Expression-bodied members

- d) Local functions

- \*\*Answer:\*\* a) Type pattern matching

75. \*\*Which of the following C# 7.0 features allows you to define a method inside another method?\*\*

- a) Local functions

- b) Inline functions

- c) Nested methods

- d) Lambda expressions

- \*\*Answer:\*\* a) Local functions

76. \*\*Which of the following is true about `is` pattern matching in C# 7.0?\*\*

- a) It can match both type and value.

- b) It can only match types.

- c) It is used only in switch statements.

- d) It cannot be used in if statements.

- \*\*Answer:\*\* a) It can match both type and value.

77. \*\*Which of the following C# 7.0 features allows you to return multiple values from a method?\*\*

- a) Value tuples

- b) Anonymous types

- c) Tuple types

- d) Local functions

- \*\*Answer:\*\* a) Value tuples

78. \*\*Which of the following is a correct way to use the `default` literal in C# 7.0?\*\*

- a) `int x = default;`

- b) `int x = 0;`

- c) `int x = null;`

- d) `int x = new int();`

- \*\*Answer:\*\* a) `int x = default;`

79. \*\*Which of the following correctly uses pattern matching in a switch statement in C# 7.0?\*\*

- a) `switch (x) { case int n: ... }`

- b) `switch (x) { case var n: ... }`

- c) `switch (x) { case n when n == 1: ... }`

- d) `switch (x) { case default: ... }`

- \*\*Answer:\*\* a) `switch (x) { case int n: ... }`

80. \*\*Which of the following C# 7.0 features allows for inline declaration of out variables?\*\*

- a) Out variable declaration

- b) Local functions

- c) Expression-bodied members

- d) Value tuples

- \*\*Answer:\*\* a) Out variable declaration

81. \*\*Which of the following is a correct example of using an inline `out` variable in C# 7.0?\*\*

- a) `if (int.TryParse("123", out int result)) { }`

- b) `if (int.TryParse("123", out var result)) { }`

- c) `if (int.TryParse("123", out result)) { }`

- d) `if (int.TryParse("123", out string result)) { }`

- \*\*Answer:\*\* a) `if (int.TryParse("123", out int result)) { }`

82. \*\*Which of the following C# 7.0 features helps in writing more concise code with lambdas?\*\*

- a) Expression-bodied members

- b) Local functions

- c) Value tuples

- d) Inline `out` variables

- \*\*Answer:\*\* a) Expression-bodied members

83. \*\*Which of the following C# 7.0 features allows for multiple return values from a method?\*\*

- a) Value tuples

- b) Anonymous types

- c) Ref locals

- d) Expression-bodied members

- \*\*Answer:\*\* a) Value tuples

84. \*\*Which of the following is a correct example of using a `ref` local in C# 7.0?\*\*

- a) `ref int x = ref array[0];`

- b) `int ref x = ref array[0];`

- c) `ref int x = array[0];`

- d) `int x = ref array[0];`

- \*\*Answer:\*\* a) `ref int x = ref array[0];`

85. \*\*Which of the following C# 7.0 features allows you to discard values in a tuple deconstruction?\*\*

- a) Discards

- b) Local functions

- c) Expression-bodied members

- d) Inline `out` variables

- \*\*Answer:\*\* a) Discards

86. \*\*Which of the following C# 7.0 features helps in writing more expressive code with pattern matching?\*\*

- a) Type pattern matching

- b) Value tuples

- c) Expression-bodied members

- d) Local functions

- \*\*Answer:\*\* a) Type pattern matching

87. \*\*Which of the following C# 7.0 features allows you to define a method inside another method?\*\*

- a) Local functions

- b) Inline functions

- c) Nested methods

- d) Lambda expressions

- \*\*Answer:\*\* a) Local functions

88. \*\*Which of the following is true about `is` pattern matching in C# 7.0?\*\*

- a) It can match both type and value.

- b) It can only match types.

- c) It is used only in switch statements.

- d) It cannot be used in if statements.

- \*\*Answer:\*\* a) It can match both type and value.

89. \*\*Which of the following C# 7.0 features allows you to return multiple values from a method?\*\*

- a) Value tuples

- b) Anonymous types

- c) Tuple types

- d) Local functions

- \*\*Answer:\*\* a) Value tuples

90. \*\*Which of the following is a correct way to use the `default` literal in C# 7.0?\*\*

- a) `int x = default;`

- b) `int x = 0;`

- c) `int x = null;`

- d) `int x = new int();`

- \*\*Answer:\*\* a) `int x = default;`

91. \*\*Which of the following correctly uses pattern matching in a switch statement in C# 7.0?\*\*

- a) `switch (x) { case int n: ... }`

- b) `switch (x) { case var n: ... }`

- c) `switch (x) { case n when n == 1: ... }`

- d) `switch (x) { case default: ... }`

- \*\*Answer:\*\* a) `switch (x) { case int n: ... }`

92. \*\*Which of the following C# 7.0 features allows for inline declaration of out variables?\*\*

- a) Out variable declaration

- b) Local functions

- c) Expression-bodied members

- d) Value tuples

- \*\*Answer:\*\* a) Out variable declaration

93. \*\*Which of the following is a correct example of using an inline `out` variable in C# 7.0?\*\*

- a) `if (int.TryParse("123", out int result)) { }`

- b) `if (int.TryParse("123", out var result)) { }`

- c) `if (int.TryParse("123", out result)) { }`

- d) `if (int.TryParse("123", out string result)) { }`

- \*\*Answer:\*\* a) `if (int.TryParse("123", out int result)) { }`

94. \*\*Which of the following C# 7.0 features helps in writing more concise code with lambdas?\*\*

- a) Expression-bodied members

- b) Local functions

- c) Value tuples

- d) Inline `out` variables

- \*\*Answer:\*\* a) Expression-bodied members

95. \*\*Which of the following C# 7.0 features allows for multiple return values from a method?\*\*

- a) Value tuples

- b) Anonymous types

- c) Ref locals

- d) Expression-bodied members

- \*\*Answer:\*\* a) Value tuples

96. \*\*Which of the following is a correct example of using a `ref` local in C# 7.0?\*\*

- a) `ref int x = ref array[0];`

- b) `int ref x = ref array[0];`

- c) `ref int x = array[0];`

- d) `int x = ref array[0];`

- \*\*Answer:\*\* a) `ref int x = ref array[

0];`

97. \*\*Which of the following C# 7.0 features allows you to discard values in a tuple deconstruction?\*\*

- a) Discards

- b) Local functions

- c) Expression-bodied members

- d) Inline `out` variables

- \*\*Answer:\*\* a) Discards

98. \*\*Which of the following C# 7.0 features helps in writing more expressive code with pattern matching?\*\*

- a) Type pattern matching

- b) Value tuples

- c) Expression-bodied members

- d) Local functions

- \*\*Answer:\*\* a) Type pattern matching

99. \*\*Which of the following C# 7.0 features allows you to define a method inside another method?\*\*

- a) Local functions

- b) Inline functions

- c) Nested methods

- d) Lambda expressions

- \*\*Answer:\*\* a) Local functions

100. \*\*Which of the following is true about `is` pattern matching in C# 7.0?\*\*

- a) It can match both type and value.

- b) It can only match types.

- c) It is used only in switch statements.

- d) It cannot be used in if statements.

- \*\*Answer:\*\* a) It can match both type and value.