1. \*\*What is the purpose of the `try` block in C# exception handling?\*\*

a) To define a method that will execute if an exception occurs

b) To specify a block of code that may throw an exception

c) To handle exceptions thrown by the `catch` block

d) To declare an exception type to be caught

\*\*Answer:\*\* b) To specify a block of code that may throw an exception

2. \*\*Which keyword is used to handle exceptions in C#?\*\*

a) `catch`

b) `throw`

c) `finally`

d) `try`

\*\*Answer:\*\* a) `catch`

3. \*\*What is the purpose of the `finally` block?\*\*

a) To execute code whether an exception is thrown or not

b) To rethrow an exception

c) To handle exceptions that occur in the `try` block

d) To define a block of code that can throw an exception

\*\*Answer:\*\* a) To execute code whether an exception is thrown or not

4. \*\*What happens if an exception is not caught in a `catch` block?\*\*

a) The application terminates

b) The `finally` block is executed

c) The `try` block is re-executed

d) The `catch` block is executed again

\*\*Answer:\*\* a) The application terminates

5. \*\*How do you rethrow an exception in a `catch` block?\*\*

a) `throw`

b) `throw ex`

c) `throw ex.Message`

d) `rethrow`

\*\*Answer:\*\* a) `throw`

6. \*\*Which of the following is true about exception handling in C#?\*\*

a) You must use a `finally` block after every `try` block

b) You can have multiple `catch` blocks for a single `try` block

c) A `catch` block can only catch exceptions of type `Exception`

d) The `catch` block must be followed by a `finally` block

\*\*Answer:\*\* b) You can have multiple `catch` blocks for a single `try` block

7. \*\*What is the base class for all exceptions in C#?\*\*

a) `Error`

b) `Exception`

c) `Throwable`

d) `Fault`

\*\*Answer:\*\* b) `Exception`

8. \*\*Which statement about custom exceptions is true?\*\*

a) Custom exceptions must always derive from `System.Exception`

b) Custom exceptions can derive from `System.Exception` or `System.ApplicationException`

c) Custom exceptions can only be used in specific applications

d) Custom exceptions cannot include additional data

\*\*Answer:\*\* b) Custom exceptions can derive from `System.Exception` or `System.ApplicationException`

9. \*\*What does the `Exception.Message` property provide?\*\*

a) A stack trace of the exception

b) The type of the exception

c) A description of the error

d) The source of the exception

\*\*Answer:\*\* c) A description of the error

10. \*\*What is the purpose of the `Exception.StackTrace` property?\*\*

a) To get a message describing the exception

b) To get the type of the exception

c) To get the line numbers and call stack where the exception occurred

d) To get the source of the exception

\*\*Answer:\*\* c) To get the line numbers and call stack where the exception occurred

11. \*\*Which of the following is not a standard exception type in C#?\*\*

a) `ArgumentException`

b) `IndexOutOfRangeException`

c) `FileNotFoundException`

d) `CustomException`

\*\*Answer:\*\* d) `CustomException`

12. \*\*What happens if an exception occurs inside a `finally` block?\*\*

a) The exception is ignored

b) The exception is propagated and the application terminates

c) The exception is caught by the `catch` block

d) The `finally` block is skipped

\*\*Answer:\*\* b) The exception is propagated and the application terminates

13. \*\*Which keyword is used to create a new exception object?\*\*

a) `new`

b) `create`

c) `exception`

d) `throw`

\*\*Answer:\*\* a) `new`

14. \*\*What should you do if you need to perform some cleanup code whether an exception is thrown or not?\*\*

a) Put the cleanup code in the `catch` block

b) Put the cleanup code in the `try` block

c) Put the cleanup code in the `finally` block

d) Handle cleanup in a separate method

\*\*Answer:\*\* c) Put the cleanup code in the `finally` block

15. \*\*Which exception class would you use to handle invalid operations in your application?\*\*

a) `NullReferenceException`

b) `InvalidOperationException`

c) `ArgumentOutOfRangeException`

d) `IOException`

\*\*Answer:\*\* b) `InvalidOperationException`

16. \*\*How can you catch exceptions of multiple types in a single `catch` block?\*\*

a) Use multiple `catch` blocks for each exception type

b) Use a `catch` block with multiple `catch` clauses

c) Use a `catch` block with a single exception type and check the type in the block

d) Use a `catch` block with a base exception type that all exceptions derive from

\*\*Answer:\*\* d) Use a `catch` block with a base exception type that all exceptions derive from

17. \*\*What does the `Exception.InnerException` property represent?\*\*

a) The exception that was caught by the `catch` block

b) A related exception that caused the current exception

c) A list of all exceptions that have occurred

d) The source of the exception

\*\*Answer:\*\* b) A related exception that caused the current exception

18. \*\*Which of the following is a common best practice for exception handling in C#?\*\*

a) Catch general exceptions whenever possible

b) Use exceptions for control flow

c) Catch specific exceptions rather than general exceptions

d) Avoid using `try-catch` blocks altogether

\*\*Answer:\*\* c) Catch specific exceptions rather than general exceptions

19. \*\*In which scenarios is it appropriate to use exception handling?\*\*

a) For control flow management

b) For validating user input

c) For handling unexpected runtime errors

d) For handling compile-time errors

\*\*Answer:\*\* c) For handling unexpected runtime errors

20. \*\*How can you ensure that a resource is properly disposed of even if an exception occurs?\*\*

a) Use a `try-catch` block

b) Use a `try-finally` block

c) Use a `try-catch-finally` block

d) Use a `finally` block only

\*\*Answer:\*\* c) Use a `try-catch-finally` block