1. What is exception handling in C++?
   1. **A mechanism to handle errors and abnormal situations during program execution.**
   2. A way to optimize program performance.
   3. A method to print debug messages.
   4. A feature to avoid using pointers.
2. Which standard C++ header file should be included for using exception handling features?
   1. **<exception>**
   2. <error>
   3. <trycatch>
   4. <stdexcept>
3. What does the std::exception class in C++ provide?
   1. **It is a base class for all C++ exceptions.**
   2. It is used to define custom exception classes.
   3. It is a built-in exception that cannot be derived from.
   4. It is used to terminate the program.
4. How do you handle an exception without catching it?
   1. by using the catch block with no arguments
   2. **by using the catch block with an ellipsis (...)**
   3. by using the catch block with a numeric value
   4. by using the catch block with a character value
5. Which keyword is used to re-throw an exception inside a catch block?
   1. rethrow
   2. **throw**
   3. continue
   4. catch
6. What is the purpose of the "std::terminate" function in C++?
   1. It throws an exception when an error occurs.
   2. It handles all uncaught exceptions.
   3. **It is called when the program encounters an unhandled exception and terminates the program.**
   4. It is used to forcibly terminate the program at any point.
7. Can a catch block catch multiple types of exceptions?
   1. No, each catch block can only catch one specific exception type.
   2. Yes, but only if the exception types are related through inheritance.
   3. **Yes, a catch block can catch any number of exceptions, regardless of their types.**
   4. Yes, but only if the exception types have the same name.
8. What happens if a function throws an exception, but it is not caught by any catch block in the calling function?
   1. The program will continue to execute normally.
   2. The function will be called again with the same arguments.
   3. **The program will terminate abruptly.**
   4. The function will be ignored.
9. When should you use exception handling in C++?
   1. **Whenever you encounter an error or exceptional situation.**
   2. As a replacement for standard if-else statements.
   3. Only when working with user-defined data types.
   4. Only when working with pointers.
10. Which of the following is an example of a standard C++ exception class?
    1. Exception
    2. Error
    3. StringException
    4. **std::runtime\_error**
11. Can you nest multiple try-catch blocks?
    1. Yes, but only up to three levels deep.
    2. **Yes, there is no limit to the nesting of try-catch blocks.**
    3. No, nesting try-catch blocks is not allowed in C++.
    4. Yes, but only if each block catches different types of exceptions.
12. In exception handling, what is the purpose of the "std::bad\_alloc" exception?
    1. It is thrown when a function is called with invalid arguments.
    2. **It is thrown when a dynamic memory allocation fails.**
    3. It is thrown when a function is overloaded incorrectly.
    4. It is thrown when there is an error in the try block.
13. What happens if a catch block throws another exception?
    1. **The program terminates abruptly.**
    2. The catch block will be executed again.
    3. The catch block will be skipped, and the next catch block will be executed.
    4. The program will ignore the new exception.
14. How can you create a custom exception class in C++?
    1. **By inheriting from the "std::exception" class.**
    2. By using the "throw" keyword with a new class name.
    3. By using the "catch" keyword with a new class name.
    4. By using the "new" keyword with a class name.
15. What is the use of the "std::nested\_exception" class in C++?
    1. It is used to create nested try-catch blocks.
    2. It is used to store multiple exceptions in a single object.
    3. It is used to store the exception's error code.
    4. **It is used to propagate exceptions across threads.**
16. What is the purpose of the "std::current\_exception" function in C++?
    1. **It returns the currently thrown exception**.
    2. It returns the most recent exception caught by a catch block.
    3. It returns a null pointer when there is no exception.
    4. It returns the exception's error message.
17. How can you catch all exceptions except "std::bad\_alloc" in C++?
    1. catch (std::bad\_alloc)
    2. catch (!std::bad\_alloc)
    3. **catch (...)**
    4. catch (!bad\_alloc)
18. Which standard C++ header file provides the "std::exception\_ptr" class?
    1. **<exception>**
    2. <stdexcept>
    3. <exception\_ptr>
    4. <exception\_handler>
19. In C++, can you have a catch block without a corresponding try block?
    1. Yes, but only if the catch block throws an exception.
    2. **No, a catch block must always be paired with a try block.**
    3. Yes, but only if the catch block contains no code.
    4. Yes, but only if the catch block has an ellipsis (...) as its argument.
20. What is the purpose of the "std::rethrow\_exception" function in C++?
    1. **It rethrows the last caught exception.**
    2. It throws a new exception.
    3. It catches all exceptions.
    4. It terminates the program.
21. What is the output of the following code?

try {

throw 42;

} catch (int num) {

std::cout << "Caught an integer: " << num << std::endl;

} catch (...) {

std::cout << "Caught an unknown exception." << std::endl;

}

* 1. **Caught an integer: 42**
  2. Caught an unknown exception.
  3. Error: Unhandled exception of type int.
  4. The program will not compile.

1. Which of the following is an example of a user-defined exception class?
   1. std::runtime\_error
   2. std::invalid\_argument
   3. **MyException**
   4. std::bad\_cast
2. Can you use multiple catch blocks for the same exception type in C++?
   1. Yes, but the compiler will issue a warning.
   2. Yes, but the catch blocks must be nested.
   3. No, only one catch block is allowed per exception type.
   4. **Yes, multiple catch blocks are allowed for the same exception type.**
3. What is the purpose of the "std::exception\_ptr::rethrow" function in C++?
   1. It throws a new exception.
   2. **It rethrows the last caught exception.**
   3. It catches all exceptions.
   4. It terminates the program.
4. Which type of exception cannot be caught using a catch block in C++?
   1. User-defined exceptions
   2. Standard exceptions
   3. **Hardware exceptions**
   4. All types of exceptions can be caught.
5. What is the output of the following code?

try {

throw std::runtime\_error("An error occurred.");

} catch (std::exception& ex) {

std::cout << "Caught an exception: " << ex.what() << std::endl;

} catch (...) {

std::cout << "Caught an unknown exception." << std::endl;

}

* 1. **Caught an exception: An error occurred.**
  2. Caught an unknown exception.
  3. Error: Unhandled exception of type std::runtime\_error.
  4. The program will not compile.

1. What happens if a function throws an exception and there is no try-catch block in the entire program?
   1. The program will continue to execute normally.
   2. **The program will terminate abruptly.**
   3. The function will be called again with the same arguments.
   4. The function will be ignored.
2. What is the role of the "std::exception::virtual const char\* what() const noexcept" function?
   1. It throws an exception.
   2. It catches an exception.
   3. **It returns the error message associated with the exception.**
   4. It handles uncaught exceptions.
3. What is an exception in C++ program?
   1. **A problem that arises during the execution of a program**
   2. A problem that arises during compilation
   3. Also known as the syntax error
   4. Also known as semantic error
4. By default, what a program does when it detects an exception?
   1. Continue running
   2. **Results in the termination of the program**
   3. Calls other functions of the program
   4. Removes the exception and tells the programmer about an exception
5. Why do we need to handle exceptions?
   1. **To avoid unexpected behaviour of a program during run-time**
   2. To let compiler remove all exceptions by itself
   3. To successfully compile the program
   4. To get correct output
6. How Exception handling is implemented in the C++ program?
   1. Using Exception keyword
   2. **Using try-catch block**
   3. Using Exception block
   4. Using Error handling schedules
7. Which of the following is an exception in C++?
   1. **Divide by zero**
   2. Semicolon not written
   3. Variable not declared
   4. An expression is wrongly written