1. What is the role of the "std::exception::what()" 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.
2. What is the purpose of throwing an exception in C++?
   1. To terminate the program immediately
   2. **To handle unexpected situations or errors**
   3. To create a new variable
   4. To call a function
3. In C++, which keyword is used to throw an exception?
   1. raise
   2. **throw**
   3. catch
   4. try
4. When an exception is thrown, the program flow is transferred to the nearest:
   1. **try block containing catch block(s)**
   2. switch statement
   3. while loop
   4. if-else statement
5. Which section of the code should contain the code that may throw an exception?
   1. try
   2. catch
   3. throw
   4. **both a) and c)**
6. In C++, can we throw an integer as an exception?
   1. **Yes**
   2. No
7. What happens if an exception is thrown and not caught?
   1. The program continues execution normally.
   2. **The program terminates abnormally.**
   3. The compiler automatically catches the exception.
   4. The program waits until the exception is caught.
8. To catch multiple types of exceptions in C++, we can use:
   1. multiple try blocks
   2. if-else statements
   3. **nested catch blocks**
   4. switch statements
9. Which block should be used to handle exceptions in C++?
   1. **catch**
   2. try
   3. throw
   4. except
10. What will happen if an exception is thrown inside a catch block?
    1. The program will ignore the new exception.
    2. The new exception will be caught by the outer try-catch block.
    3. **The program will terminate.**
    4. The catch block will be skipped.
11. If a function throws an exception and it is not caught within that function, what will happen?
    1. The program will terminate.
    2. The exception will be ignored.
    3. The function will re-execute.
    4. **The exception will propagate to the calling function.**
12. What is the purpose of rethrowing an exception in C++?
    1. To handle it at a later stage
    2. To create a new exception
    3. **To propagate it to the calling function**
    4. To display a custom error message
13. In C++, can we catch an exception by value?
    1. **Yes**
    2. No
14. When catching an exception by reference in C++, which keyword is used?
    1. as
    2. is
    3. by
    4. **&**
15. What is the syntax to specify that a C++ function may throw an exception?
    1. void functionName() excepts
    2. void functionName() throw
    3. **void functionName() throws**
    4. void functionName() noexcept
16. If a function doesn't specify any exceptions in C++, what does it mean?
    1. **The function can throw any type of exception.**
    2. The function cannot throw any exceptions.
    3. The function will automatically catch any exceptions thrown.
    4. The function will rethrow any exceptions caught.
17. In C++, can we use a base class catch block to catch derived class exceptions?
    1. **Yes**
    2. No
18. Which C++ keyword is used to define a user-defined exception class?
    1. throw
    2. exception
    3. **class**
    4. catch
19. When catching exceptions in C++, which catch block should come first for a hierarchy of catch blocks?
    1. **The most specific one**
    2. The least specific one
    3. Any order will work fine
    4. It doesn't matter
20. What will be the output of the following code?

#include <iostream>

void throw\_exception() {

throw "An exception occurred!";

}

int main() {

try {

throw\_exception();

}

catch (const char\* ex) {

std::cout << ex;

}

return 0;

}

* 1. exception!
  2. **An exception occurred!**
  3. The program will terminate.
  4. Compiler error

1. In C++, can we have a try block without any catch blocks?
   1. Yes
   2. **No**
2. In C++, can we catch an exception by a pointer?
   1. **Yes**
   2. No
3. What is the purpose of using the "nothrow" keyword with "new" in C++?
   1. **To suppress the exception if the memory allocation fails**
   2. To allocate memory on the stack instead of the heap
   3. To allocate memory without initializing it
   4. To allocate memory with a specific alignment
4. Can we catch multiple exceptions using a single catch block in C++?
   1. **Yes**
   2. No
5. In C++, can a function that does not throw any exceptions be called inside a try block?
   1. **Yes**
   2. No
6. What is the purpose of using the "noexcept" specifier in C++?
   1. **To specify that a function will not throw any exceptions**
   2. To specify that a function will always throw exceptions
   3. To indicate that a function can only throw specific exceptions
   4. To indicate that a function can throw any type of exception
7. What is the output of the following code?

#include <iostream>

int main() {

try {

throw 10;

}

catch (int x) {

std::cout << "Caught an integer: " << x;

}

catch (...) {

std::cout << "Caught something else.";

}

return 0;

}

* 1. **Caught an integer: 10**
  2. Caught something else.
  3. The program will terminate.
  4. Compiler error

1. What is the purpose of std::exception in C++?
   1. To handle exceptions automatically
   2. **To serve as a base class for user-defined exceptions**
   3. To terminate the program on exception
   4. To specify exceptions in the code
2. In C++, can we use "throw" without an argument?
   1. Yes
   2. **No**
3. What happens when this C++ program is compiled?

#include <iostream>

#include <string>

#include <cstdlib>

using namespace std;

class A

{

int a;

public:

A(){}

};

class B: public A

{

int b;

public:

B(){}

};

void func()

{

B b;

throw b;

}

int main()

{

try{

func();

}

catch(B \*b){

cout<<"Caught B Class\n";

}

catch(A a){

cout<<"Caught A Class\n";

}

}

* 1. Caught B Class
  2. **Caught A Class**
  3. Compile-time error
  4. Run-time error

1. Which of the following statements are true about Catch handler?

i) It must be placed immediately after try block T.

ii) It can have multiple parameters.

iii) There must be only one catch handler for every try block.

iv) There can be multiple catch handler for a try block T.

v) Generic catch handler can be placed anywhere after try block.

* 1. Only i, iv, v
  2. Only i, ii, iii
  3. **Only i, iv**
  4. Only i, ii

1. What is the output of this program?

#include <iostream>

using namespace std;

int main()

{

try

{

throw 'b';

}

catch (int param)

{

cout << "Int Exception";

}

catch (...)

{

cout << "Default Exception";

}

cout << "After Exception";

return 0;

}

* 1. **Default Exception After Exception**
  2. Int Exception After Exception
  3. Int Exception
  4. Default Exception

1. What is the output of this program?

#include <iostream>

using namespace std;

int main()

{

try

{

throw 10;

}

catch (...)

{

cout << "Default Exceptionn";

}

catch (int param)

{

cout << "Int Exceptionn";

}

return 0;

}

* 1. Default Exception
  2. Int Exception
  3. **Compiler Error**
  4. None of the above

1. What is the output of this program?

#include <iostream>

using namespace std;

int main()

{

int P = -1;

try {

cout << "Inside try";

if (P < 0)

{

throw P;

cout << "After throw";

}

}

catch (int P ) {

cout << " Exception Caught";

}

cout << " After catch";

return 0;

}

* 1. Inside try Exception Caught After throw After catch
  2. **Inside try Exception Caught After catch**
  3. Inside try Exception Caught
  4. Inside try After throw After catch