C++ Exceptions

C++ has a set of standard exception classes that can be used and extended. Logical and run-time errors are defined in <stdexcept> header file and is derived from <exception>. It is good to use the std:: exception classes when you can over your own exception classes. If you catch references, you can take advantage of polymorphism with exceptions.

Exceptions are caught in a bottom-down hierarchy. Specific derived classes exceptions are handled first, followed by less specific groups of exceptions that is, up to the base classes and, finally, a catch(...) handler. Handlers of the specific derived objects must appear before the handlers of base classes. This is because handlers are tried in order of appearance. It's, therefore, possible to write handlers that are never executed; for example, by placing a handler for a derived class after a handler for a corresponding base class.

Read this website:
https://www.acodersjourney.com/top-15-c-exception-handling-mistakes-avoid/

Exercise 1 Write a function calculate_average() which takes four int arguments which are grades for four courses in the semester and returns their average as a float. The calculate_average() function should take only valid range for marks which is between 0 - 100, inclusive. If the marks are out of range, throw an OutOfRangeException. Write the specification for this function as comments above it specifying the preconditions, postconditions and invariants and what exceptions are thrown when so that the caller understands how it behaves.

Exercise 2 Write a program that will convert dates from numerical month/day/year to alphabetic format. Thus, 01/1/1970 would be January 1st, 1970 and 5/2/19 would be May 2nd, 2019. You should write four exception classes; (1) DateFormatError, (2) MonthFormatError, (3) DayFormatError, and (4) YearFormatError. Just assume there are only 29 days in February.

Class Hierarchy	Description
exception	This is the base class
bad_alloc	Thrown by new, an allocation request fails.
bad_cast	Thrown by dynamic_cast when failed cast to a reference type.
bad_exception	Thrown when an exception doesn't match any catch clause.
bad_typeid	Thrown by typeid operator when the operand for typeid is a NULL pointer.
The logical errors are normally caused by programmer mistakes.	
logic_error	As the base class for all exceptions thrown to report errors presumably detectable before the program executes, such as violations of logical preconditions.
domain_error	As the base class for all exceptions thrown to report a domain error.
invalid_argument	As the base class for all exceptions thrown to report an invalid argument.
length_error	As the base class for all exceptions thrown to report an attempt to generate an object too long to be specified.
out_of_range	As the base class for all exceptions thrown to report an argument that is out of its valid range.
The run-time errors normally occur because of mistakes in either the library functions or in the run-time system	
runtime_error	As the base class for all exceptions thrown to report errors presumably detectable only when the program executes.
overflow_error	As the base class for all exceptions thrown to report an arithmetic overflow.
range_error	As the base class for all exceptions thrown to report a range error.
underflow_error	As the base class for all exceptions thrown to report an arithmetic underflow.
ios_base::failure	The member class serves as the base class for all exceptions thrown by the member function clear() in template class basic_ios.