CS 405 Module Four Exceptions Activity Guidelines and Rubric

Overview

You are a senior software developer on a team of software developers who are responsible for a large banking web application. Your manager has learned that there is some code that seems to terminate abruptly, crashing to the desktop without any warnings or information displayed to the user. The other developers spent a lot of time tracing down the issue, and they narrowed it down to some suspect code. However, the decision was made not to change the suspect code, and instead to capture the errors in the code and display appropriate errors to the user instead of just crashing to the desktop. The other developers know this would require using exceptions, but they are not sure what to do. They have asked you to help.

Prompt

You will learn to do the following:

- Create an exception try-block, which will wrap code that can potentially throw exceptional errors
- Catch an exception derived from std::exception
- Catch all unhandled exceptions
- Throw a standard C++ exception
- Create, throw, and catch a custom exception

The following are a few key notes:

- The source code has been commented using TODO to explain the detailed rules you must follow.
- While it is possible to throw exceptions not derived from std::exception, do not do this. It is an anti-pattern and a bad practice.
- Do not forget that you can leverage capabilities provided by the standard C++ library to help you achieve success.

Please comment on any changes you make in the code to explain your intent. You will also create a brief written summary on whether the C++ exception catch-all handler is a good or bad idea and why, any issues you encountered, and how you resolved those issues.

Specifically, prepare the following in a static testing summary:

- Exceptions Coding: Complete the coding activity by successfully implementing a custom exception class; throwing custom and standard exceptions; implementing standard, custom, and catch-all catch handlers; and displaying a message to the user including the exception message text (what() method) whenever an exception is caught.
- C/C++ Program Functionality and Best Practices: Demonstrate industry standard best practices, including in-line comments and appropriate naming conventions to enhance readability of code, and develop functional C/C++ code that illustrates software design pattern approach.
- Summary of Process: Provide a summary of the debugging that is thorough and systematic, including specific types of bugs, and that accurately describes the corrections.

To complete this assignment, download the Exceptions source code file to use as guidance as you move through the activity. You will use your development environment to complete this activity.

What to Submit

Submit the completed source code as a ZIP file. Also, include a Word document that contains a screenshot of the application console output and your brief process summary.

Module Four Exceptions Activity Rubric

Criteria	Proficient (100%)	Needs Improvement (75%)	Not Evident (0%)	Value
Exception Coding	Completes the coding activity by successfully implementing a custom exception class; throwing custom and standard exceptions; implementing standard, custom, and catch-all catch handlers; displays a message to the user including the exception message text (what() method) whenever an exception is caught	Completes most of the coding activity, but needs to address one of the following: does not implement a custom exception class; does not throw custom and standard exceptions; does not implement standard, custom, and catch-all catch handlers; does not display a message to the user including the exception message text (what() method) whenever an exception is caught	Does not attempt criterion	70
C/C++ Program	Demonstrates industry standard best	Shows progress toward proficiency, but	Does not attempt criterion	10

Functionality and Best Practices	practices, including in-line comments and appropriate naming conventions to enhance readability of code; develops functional C/C++ code that illustrates software design pattern approach	with errors or omissions; areas for improvement may include implementation of software design patterns in code		
Process Summary	Provides a summary of the debugging that is thorough and systematic, including specific types of bugs; accurately describes the corrections and has a screenshot of the application console output, from the compiler, and of the error list	Provides a summary of the debugging, but the summary lacks detail, the approach is unsystematic, specific types of bugs are not differentiated, or the corrections are not fully described, or a screenshot is missing	Does not attempt criterion	20
			Total:	100%