**Module Four Exceptions Summary**

Joseph Veneski

Department of Computer Science, Southern New Hampshire University

CS-405 Secure Coding

Professor Kaan Esendemir

March 30, 2024

**Module Four Exceptions Summary**

**Summary:**

To begin this assignment the do\_division() and divide() methods were addressed first. The instructions stated to throw an exception to deal with divide by zero errors which was done in this snippet:

A screen shot of a computer error

Description automatically generated

This exception is then handled in the do\_division() method with a try-catch:

A computer code with colorful text

Description automatically generated

Next steps were to catch a standard exception from the do\_even\_more\_custom\_application\_logic():

A computer screen shot of a black background

Description automatically generated

Handled in the do\_custom\_application\_logic() method: A computer screen shot of a program

Description automatically generated

The assignment also required throwing and handling a custom exception derived from std::exception. A custom exception class was created: A screen shot of a computer code

Description automatically generated

Then a custom exception is thrown in the do\_custom\_application\_logic() method:

A black screen with white text

Description automatically generated

Finally, the main function requires exception handlers to catch exceptions and print messages to the console. A try-catch block is wrapped around the main and handles the custom exception, standard exception, and then any unhandled exceptions with a catch-all:

A computer screen shot of a program

Description automatically generated

The modifications to the code result in the following console output when run:

A screenshot of a computer program

Description automatically generated

The code handles division by zero exceptions anticipating possible bugs ahead of time. Custom exception handling is implemented and can be utilized and adapted as the custom application logic functions grow. A catch-all exception handler works as a safety net but handling as many detailed exception cases over relying on the catch-all is safer.