**Victor Udeh  
CS405 M4-1  
Date:   
  
  
Summary of Process**

During the debugging process, I encountered and systematically addressed several issues related to the C++ code involving exception handling. Below is a detailed summary of the types of bugs encountered and the corrections made:

1. **Exception Specification Mismatch in what() Method**:
   * **Issue**: The compiler reported an error with the what() method of the CustomException class. The error indicated that the exception specification (noexcept) was more relaxed compared to the base class (std::exception), which uses \_NOEXCEPT.
   * **Solution**: I replaced noexcept with \_NOEXCEPT to match the compiler's expected specification for compatibility with the base class.
2. **Incorrect Use of noexcept in Function Definition**:
   * **Issue**: The do\_division() function had an incorrect noexcept specifier, which led to an error stating that a function body was expected after the function declarator.
   * **Solution**: I removed the noexcept specifier from the do\_division() function to allow proper exception handling. This resolved the compilation issue.
3. **Warnings Regarding C++11 Features**:
   * **Issue**: The compiler generated warnings about the use of C++11 features (override and auto) without explicitly specifying the C++11 standard.
   * **Solution**: To address these warnings, I compiled the code using the -std=c++11 flag, enabling C++11 features and removing the warnings.
4. **Division by Zero Exception**:
   * **Implementation**: The function divide() was modified to throw a std::overflow\_error exception when the denominator is zero, ensuring that the division by zero scenario is handled gracefully.
   * **Verification**: The exception was successfully caught in the do\_division() function, and an appropriate error message was displayed.
5. **Custom Exception Handling**:
   * **Implementation**: A custom exception (CustomException) was thrown in the do\_custom\_application\_logic() function and caught explicitly in main(). This verified the functionality of custom exception handling.
6. **Catch-All Handler**:
   * **Implementation**: Added a catch-all (catch(...)) handler in main() to ensure that any unexpected or uncaught exceptions are captured. This prevents the application from crashing abruptly.

**Summary of Corrections**

* Updated what() method in CustomException to use \_NOEXCEPT instead of noexcept for compatibility with the base class.
* Removed the incorrect noexcept specifier from the do\_division() function.
* Compiled the code using the -std=c++11 flag to enable C++11 features without warnings.
* Verified the handling of division by zero using std::overflow\_error.
* Implemented and verified custom and catch-all exception handling to enhance the robustness of the application.

These corrections ensured that the application handled exceptions properly without abrupt termination, providing a user-friendly experience with appropriate error messages.

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated