1. Summary:  
     
   **Custom Exception Implementation:**
   * **Issue:** The initial code did not include a custom exception derived from std::exception.
   * **Correction:** Implemented a CustomException class that inherits from std::exception and overrides the what() method to provide a specific error message. This helps in clearly identifying exceptions thrown by custom application logic.
2. **Standard Exception in Application Logic:**
   * **Issue:** The function do\_even\_more\_custom\_application\_logic() was intended to throw a standard exception, but no exception was being thrown.
   * **Correction:** Added a throw std::runtime\_error("Standard exception thrown in do\_even\_more\_custom\_application\_logic."); to simulate an error scenario. This ensures that the exception handling mechanism in do\_custom\_application\_logic() is properly tested.
3. **Exception Handling in Custom Application Logic:**
   * **Issue:** The call to do\_even\_more\_custom\_application\_logic() was not wrapped in a try-catch block, so any thrown exception would have terminated the program.
   * **Correction:** Wrapped the call in a try-catch block that catches std::exception and prints a message along with the exception’s what() output. This allows the program to handle the exception gracefully and continue processing. After handling the exception, the function then throws a CustomException to test catching of custom exceptions in main().
4. **Division by Zero Handling:**
   * **Issue:** The divide() function did not check for a division by zero error, which is a critical runtime bug.
   * **Correction:** Added a check in divide() to detect if the denominator is zero, and if so, throws a std::invalid\_argument with an appropriate error message. This adheres to safe programming practices by preventing undefined behavior from division by zero.
5. **Specific Exception Handling in Division:**
   * **Issue:** The function do\_division() was missing a try-catch block that specifically handled exceptions thrown by divide().
   * **Correction:** Implemented a try-catch block in do\_division() that catches std::invalid\_argument exceptions. This ensures that the division error is handled locally and the error message is displayed to the user.
6. **Comprehensive Exception Handling in Main:**
   * **Issue:** The main function initially lacked a comprehensive exception handling mechanism, potentially leaving some exceptions uncaught.
   * **Correction:** Wrapped the entire main() logic in a try-catch block that:
     + First catches the custom CustomException to handle errors specific to the custom logic.
     + Then catches any std::exception to handle standard exceptions.
     + Finally, includes a catch-all handler (catch (...)) to ensure that any unexpected exceptions are caught, thus preventing the program from crashing ungracefully.
7. **Inline Documentation and Best Practices:**
   * **Issue:** The original code was sparse on inline comments, making it harder to understand the purpose behind each modification and the overall flow of exception handling.
   * **Correction:** Added detailed inline comments throughout the code to explain the intent behind each change. This not only clarifies the functionality but also adheres to industry best practices for maintainable code.

Screenshot:

A screenshot of a computer program

AI-generated content may be incorrect.