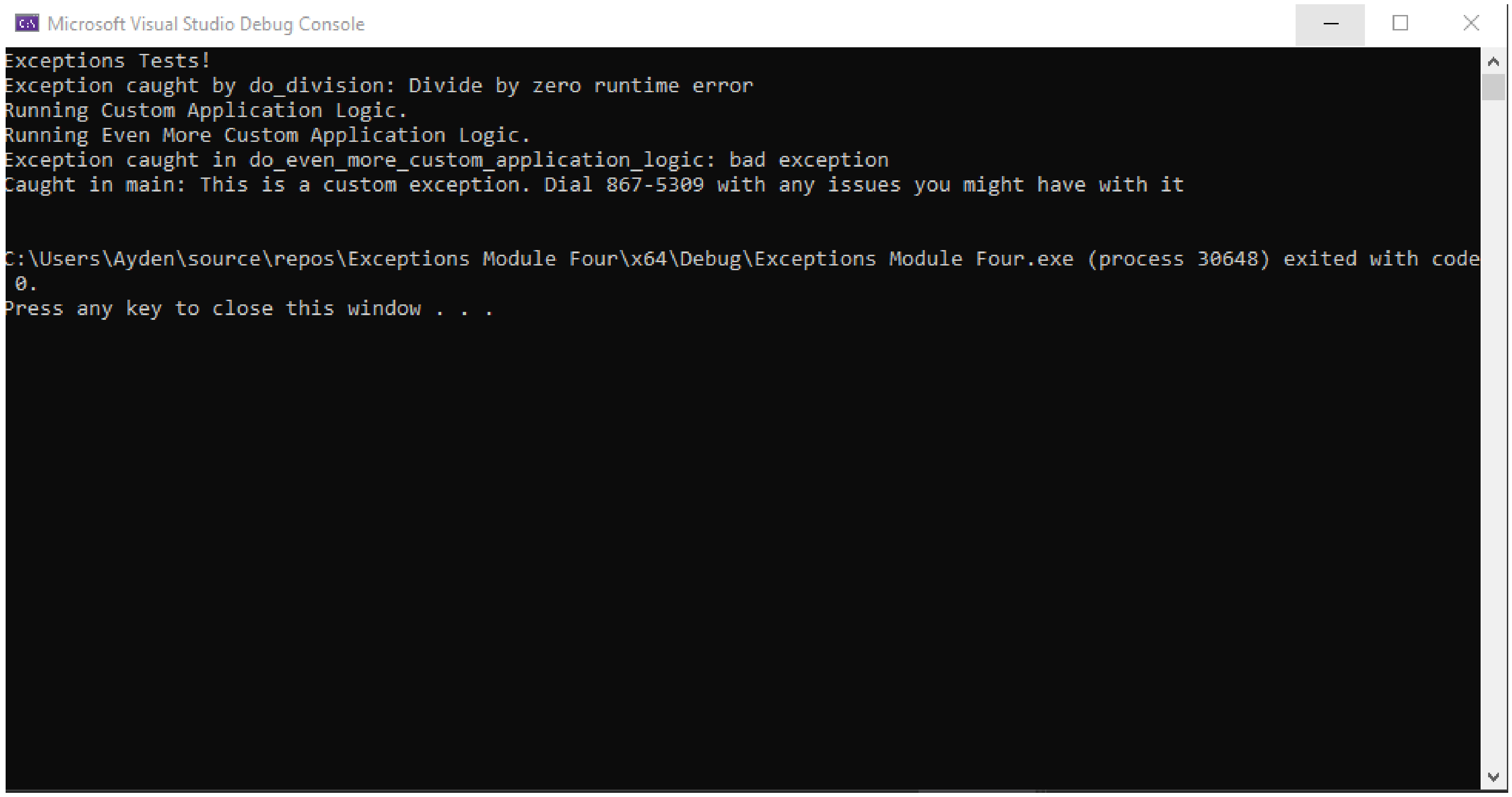
A quick summary of my process: Well, catching exceptions isn't much more difficult than adding in the try-catch blocks that tries out a function and catches the exception if it goes wrong. I created a custom exception by deriving my custom class from the std::exception class, with a custom exception error I made. The next block required me to just throw an exception, so I threw "bad exception" because it reminds me of reprimanding a dog, as if I am reprimanding the exception: "Bad exception, no!" There weren't any instructions as to what to throw other than a standard exception, so I decided to be humorous. Well, at least I found it funny! After that I just had to make a try-catch block for every place that I needed to catch exceptions. The only other exception that I had to define was one for dividing by zero, which I obviously chose as a divide-by-zero runtime error, since that's what it is. The main() function just ran all the other functions and caught errors where it needed to.



```
xceptions Tests!
xception caught by do_division: Divide by zero runtime error
unning Custom Application Logic.
unning Even More Custom Application Logic.
xception caught in do_even_more_custom_application_logic: bad exception
aught in main: This is a custom exception. Dial 867-5309 with any issues you might have with it


:\Users\Ayden\source\repos\Exceptions Module Four\x64\Debug\Exceptions Module Four.exe (process 30648) exited with code
 0.
ress any key to close this window . . .
```