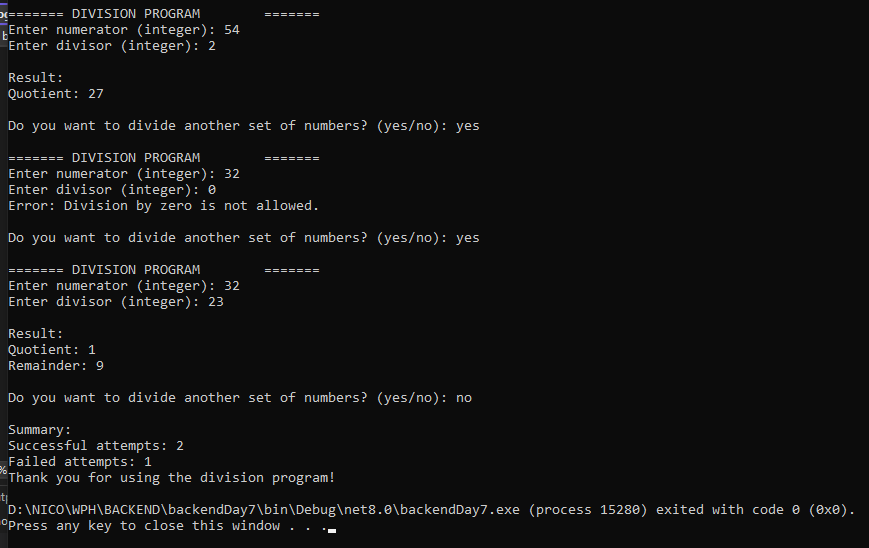
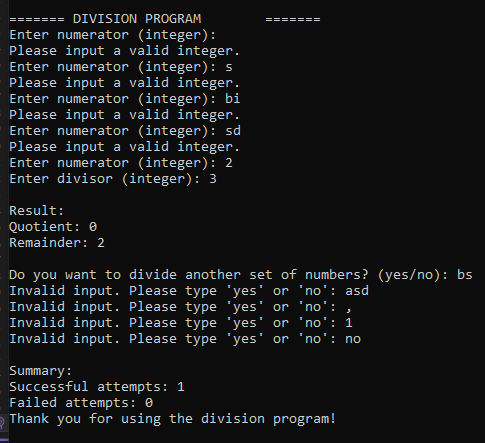


**OUTPUT**

****

**ERRO HANDLING**

****

**REFLECTION**

In my program, I separated the logic for division by putting it in its own method called DivideNumbers inside the ProcessNumbers class. I did this to keep my code more organized and avoid mixing everything in the Main() method. At first, I actually forgot to add the try-catch block, so when I tried dividing by zero, the program crashed. After realizing that, I added proper exception handling in the Main method to catch errors and display a clear message when something goes wrong, like dividing by zero.

I also created a custom exception called CustomDivideByZeroException, which is thrown when the divisor is zero. This made the error more meaningful and specific to the situation, instead of using the default system exception. For input validation, I used a method that checks if the user's input is a valid integer using int.TryParse. If it isn’t, the program tells the user and asks again until a valid number is entered, which helps avoid random errors from invalid input.

To let the user try again, I used a do-while loop, and I added another loop to make sure they type only "yes" or "no" when asked if they want to try again. This makes the program feel more controlled and user-friendly. I also displayed messages that clearly show the result, like the quotient and remainder, or explain if there was an error. One challenge I faced was learning how to make a custom exception and place the error handling in the right spot, but once I figured it out, the program worked much better. Overall, I learned a lot about structuring code properly, validating input, and handling exceptions to make the program more reliable.