Lab Assignment 5 (Submission Deadline: 9:30 AM, October 25, 2023)

1. Create a C# program that intentionally throws a DivideByZeroException when dividing by zero. Catch the exception and handle it gracefully.
2. Write a program that attempts to access an array element at an index that is out of bounds. Use a try-catch block to handle the IndexOutOfRangeException.
3. Create a C# program that uses a try-catch block to handle an exception when converting a string to an integer using int.Parse(). Handle the FormatException that may occur.
4. Implement a C# program that uses a custom exception class. Create a custom exception and throw it in your code when a specific condition is met.
5. Build a C# program that demonstrates the use of multiple catch blocks for different exception types. Handle exceptions such as IndexOutOfRangeException, FormatException, and InvalidOperationException.
6. Create a C# program that includes nested try-catch blocks. Throw an exception in an inner block and catch it in the outer block. Explain the flow of execution.
7. Implement a program that divides two numbers entered by the user. Handle exceptions like division by zero and invalid input. Continue to prompt the user for valid input until a valid division is performed.
8. Develop a C# program that demonstrates how to use the throw statement to rethrow an exception. Catch the rethrown exception and handle it appropriately.
9. Develop a program that simulates a simple calculator with basic arithmetic operations (addition, subtraction, multiplication, and division). Use exception handling to catch and handle various type of exceptions that may occur.
10. **Scenario**
    You are developing a simple e-commerce application in C#. One of the features is a shopping cart that allows users to add items to their cart. The cart is represented as an array of integers, where each integer corresponds to an item's price. Users can input the price of an item they want to add to the cart. You want to handle exceptions gracefully to ensure a smooth user experience. If the user enters an invalid price, your code should catch and handle the exception appropriately.
    Question:
    Write a C# program that simulates adding items to a shopping cart. The program should take user input for the price of items and store them in an array. Implement exception handling with multiple catch blocks to handle various scenarios. Specifically, you should handle the following

exceptions:

- If the user enters a negative price, catch and handle the exception as a "NegativePriceException." Display a message indicating that the price entered is invalid.
- If the user enters a non-numeric value (e.g., a string), catch and handle the exception as a "FormatException." Display a message indicating that the input is not a valid price.
- If the user enters a price that exceeds a predefined maximum value (e.g., 10000), catch and handle the exception as a "PriceTooHighException." Display a message indicating that the price entered is too high.

Note: The program should continue to prompt the user for prices until a valid price is entered. After each valid price is entered, add it to the shopping cart array. Once the user is done adding items, display the total price of the items in the cart. Ensure that the program uses multiple catch blocks to handle the specific exceptions mentioned above and provides informative error messages.