Assignment instructions

As always, read ALL the instructions before starting any design or coding of the applications.

This module focuses on the use of validation and exception handling. This assignment also focuses on the use of those concepts.

For this exercise, you will modify a program without a step by step guide as show in the text. You must be using Visual Studio for this exercise.

First you need to download the project. This is the same project that you started with in the previous module. I know there is a lot going on this week, so in an effort to save some time, you may work using your Module 6 Submission, or you can start with a fresh copy of it. You may download the entire original application code from this link Download link. The zip file was created with 7Zip so you may need to download that application to unzip it correctly. There is a link to 7Zip in the Module 6 assignment.

Note that I am listing all of the assignment requirements that are likely repeats from the Module 6 Assignment for clarity. If you're using your Module 6 code, just double-check these requirements against what you already have done.

Modify the code with the following Specifications and Criteria:

Design of the Application UI:

Some of the aspects of the UI design criteria may already be in place. You should double check them though and change those that need to be changed.

- Change the information on the form so that the Windows Title Bar will show "Valid ID Required" instead of Form1
- Name the form itself to `frmRealID`
- Change the name of the files for the form to match its name
  - For example, this means in the Solutions Explorer, the .cs File for the form should say **`frmRealID.cs`**
- Have the form start up in the windows center position
- The Tab Order for the form should be Left Operand Text Box, Right Operand Text Box, + Button, - Button, * Button, / Button and finally the % Button.
- You are also to create a Clear button and an Exit button.
- All names of the buttons, labels, and textboxes should be changed to meaningful names. If you use a fresh copy of the files, the names and event handlers are defaults as assigned by Visual Studio when they were created.
  - Just a note of caution, remember that since the button click code is already there, changes need to be made so that the button click code and the buttons still communicate

- Change the form background from the default gray color to a color of your choice.
- CHANGE FOR MODULE 7: use a smaller font size for the display label, so all validation messages can be read.

Operations of the Application:

- The present operation should not change from the User's point of view. Buttons should act the same way as they do now.

- When the user clicks the "Exit" button, the application closes
  - Tie the Escape Key to the "Exit" Button using the form property covered
- When the user clicks the "Clear" button, the operand textboxes and display label are to be emptied.
- The + Button
  - Adds the Right and Left Operand as it does now.
- The - Button
  - Subtracts the Right operand from the Left operand as it does now.
- The * Button
  - Multiplies the Right and Left Operand as it does now.
- The / Button
  - Divides the Left Operand by the Right Operand as it does now.
- The % Button

- Performs the Modulus Operation by dividing the Left Operand by the Right Operand and giving the remainder back as it does now.

- As in Module 6, you are to use a method to perform the actual calculations.
- Here is the change for this module, add validation code and/or exception handling to EACH button
  - For validation, you are welcome to use "Try/Catch" blocks, a series of if/else statements, or a combination of these, as long as all other requirements are met
  - If either number is negative in the modulus operation, then a validation check fails and an error message needs to be displayed
  - Each error or validation check failure needs a message associated with it.

    - For instance, you should not use generic error messages such as "invalid input".
- Things to validate
  - Validate that the user actually does enter something on each of the two text boxes
  - Validate that Modulus operation is not being done with negative numbers.
  - Handle any exception created during the conversion of the textbox contents to a number

- Handle any exception created by dividing by zero OR prevent it by validation
- Handle any "unexpected" exceptions in the entire application
- **Display all messages to the user about invalid numbers or exceptions in the label, do not use a MessageBox**
- **Do not do the methodology where the user fixes one error to just get another because you did not display all the error messages at once**
  - Note: if the user tries to run the application with an empty field, the application should tell the user fields cannot be empty. It does NOT need to say all error messages related to that operand field. It does, however, need to display valid messages for both operands if they are present.

- Extra credit opportunity. For a couple points of extra credit, display error messages in one color and valid responses in another. For instance, if the user enters an error, the text might be red (your choice), but when the user enters valid data, the display should be a different color, such as black (again, your choice). If you choose to attempt this, make sure the text is legible on the background of the label (don't put red text on a pink background).

Coding Specifications for the Application:

- None of the controls have meaningful names, Change the controls to have meaningful names.

- Be consistent throughout the program on your naming of controls
- Use an empty string to set a text property to empty
- example: `textbox1.Text = "";`
- Remember, a label text property can be set just like a textbox property
- Put a multiple line comment at the top of the code, just under all those using statements, that contains your name, the class information, and due date of the exercise on separate lines
- In the chapter, they cover validation using methods, you should do methods but for this exercise, methods are not a requirement for validation

NOTE: One goal is that the program gracefully handles any exceptions. During testing, the debug mode should never come up. Your program should gracefully handle all exceptions. Gracefully means it does not crash.

Screen Shots needed for Submission:

- Make a screen shot for each of the following points in the creation process
- The form after you have completed all the visual changes specified in the UI design. Many of the changes will not show in a screen shot but the color change will

- Show the code for the validation and exception handling you create in as few screen shots as possible
  - Highlight the method code in the screen shot(s)
- Run the application in debug mode, screen shot the whole screen showing the form starting up

- For the operations screen shots, show one screen shot depicting each of the following scenarios
  - Error message for missing information in the textboxes
  - Error message for non numbers in the textboxes
  - Divide by Zero message
  - Negative Numbers on Modulus message
  - Show multiple errors happening together and the resulting messages all displayed together
  - You will not be able most likely to produce an "unexpected" error but the code will need to be there to handle those unexpected errors
  - Show various errors and validation checks on the buttons.
  - Show each error message that you can produce (multiple screenshots required)