Prime Numbers
COMP 445
10 Points

The purpose of this assignment is to give you practice writing asynchronous code that does not block the UI thread in a GUI application. You are to modify a given application that finds all the prime numbers between two given numbers.

1. Clone or download the starter code from https://github.com/fmccown/PrimeNumbersGUI

2. Look through the code to familiarize yourself with what is currently happening when the Start button is pressed. The appropriate controls are disabled, the for loop finds all the prime numbers between the entered range, and then the controls are enabled. While the loop is executing the UI thread is unavailable to process other messages from Windows, so the window locks-up.

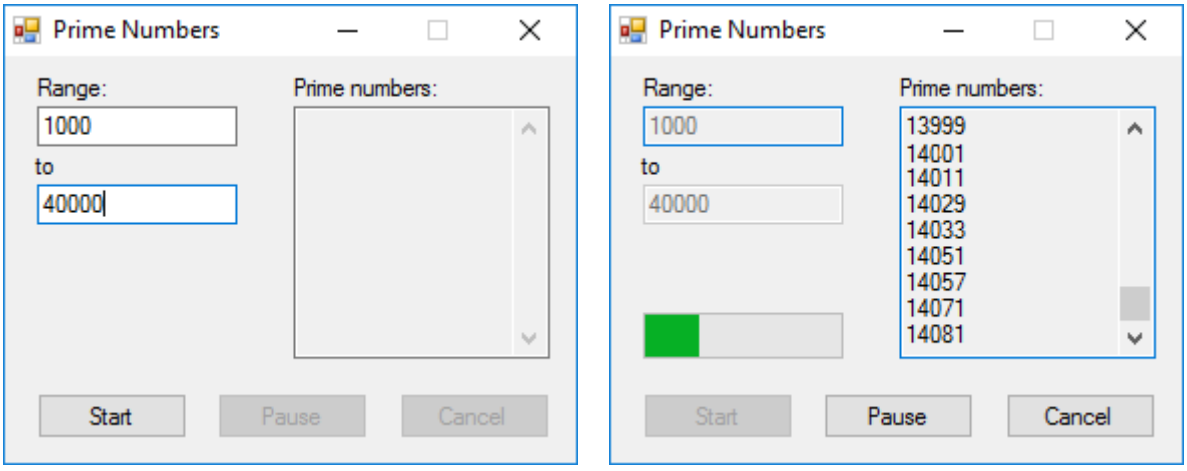
3. Convert the code to run asynchronously by making `startButton_Click()` an async method.

4. Create a Task in `startButton_Click()` to run asynchronously that executes the for loop that finds prime numbers. You may want to put that block of code into a new method since `startButton_Click()` is so long.

5. Run your program in debug mode, and verify that an exception is thrown when you press the Start button because the `numbersTextBox` and the `progressBar1` are modified by a thread in the thread pool (the Task) instead of the UI thread.

6. Fix the `AddNumberToTextBox()` method so `numbersTextBox` and the `progressBar1` are modified by the UI thread instead of the thread that is finding the prime numbers.

7. Run your program and verify that the prime numbers are being output correctly and the window is no longer frozen while the for loop executes. Verify that Start is disabled after it is pressed and re-enabled when all the primes have been found. Pause and Cancel should be enabled while the primes are being found and disabled once they're all found. Pressing Pause and Cancel does nothing.



8. Add the `CancellationToken` to the Task so the for loop can be cancelled, and add the appropriate code to cancel the Task when the cancel button is clicked.

9. Run your program and press Cancel when the program is calculating primes. Verify that the Task stops finding primes when Cancel is pressed.

10. The program currently throws an exception if the user enters text or nothing at all in the input fields. Add data validation by calling `Int32.TryParse()` or calling `Convert.ToInt32()` in a try-catch. Also verify that the starting number is larger than the ending number. Use `MessageBox.Show()` to display appropriate error messages. The dialog boxes should show the `MessageBoxIcon.Error` icon.

11. **3 bonus points** for making the Pause/Resume button work. The Pause button should cancel the BW just like the Cancel button, and the Pause button should change to a Resume button when clicked. Clicking the Resume button should re-start the Task using the last number it was computing before the Task was cancelled. So if the Task was paused when the for loop was about to calculate if 15,233 is prime, the for loop should resume the calculation with 15,233. The Resume button becomes a Pause button when clicked. To add this functionality, keep track of the last number the for loop was working on when the Task was cancelled in a class-level variable. If the Resume button re-starts the Task, use the class-level variable to restart the for loop. You'll likely have to edit how the buttons are enabled and disabled when the Task is paused and resumed.

Submit your completed project as a zip file to Easel before class on the due date.