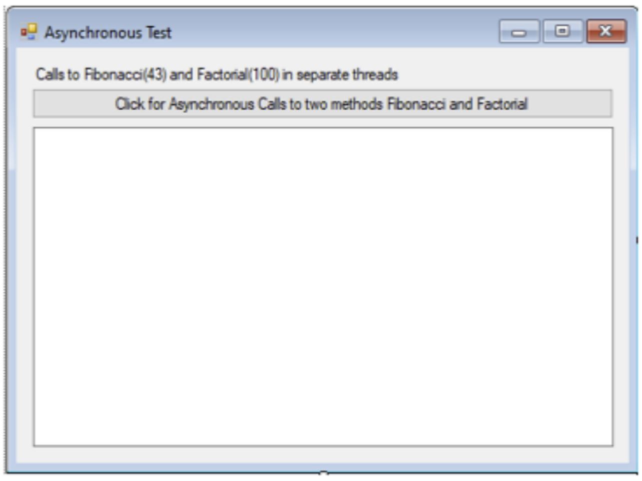
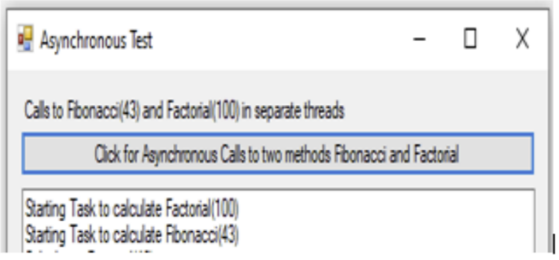
Exercise 01: Asynchronous Programming. Build the following app using Win Forms. *[20 marks]*

Factorial (for 100) and Fibonacci (for 43rd term) recursive implementation should be asynchronous (use of async and await; and Task.Run() method). See the screen shot below. Refer the code example covered in the class. For large factorial values, use BigInteger type.


Asynchronous Test
Calls to Fibonacci(43) and Factorial(100) in separate threads
Click for Asynchronous Calls to two methods Fibonacci and Factorial


Asynchronous Test
Calls to Fibonacci(43) and Factorial(100) in separate threads
Click for Asynchronous Calls to two methods Fibonacci and Factorial

```
Starting Task to calculate Factorial(100)
Starting Task to calculate Fibonacci(43)
```

Exercise 02: *[10 marks]*

Create a console app and implement an extension method – int **CharCount()** for built-in class **StringBuilder** to count the number of characters contained in a StringBuilder object including blank spaces.

[You need to add a separate ***StringBuilderExtensions.cs*** file containing class – **StringBuilderExtensions,** and adding **CharCount** method in there.]

In the class file, containing Main() method, you will be testing this method.

For example, if a StringBuilder object strobj = ”This is a demo test”, the number of characters contained in strobj is 19.

Exercise 03: *[10 marks]*

Create a console C# app in which you are required to create the following generic method and test it for passing integers array and doubles array.

- public static T **SubArray**(T a[ ], int *startIndex*, int *endIndex*) which returns a sub array containing the elements between - startIndex value and endIndex value (but not including both).

You need to add validations for startIndex such as it should not be negative and should not be greater than size of the array or endIndex. Also, for endIndex such as it can not be negative, less than startIndex or higher than the size of the array.

Exercise 04: Create a console app (.Net Framework) and do the following: *[10 marks]*

You are required to create the following method using lambdas (either expression or statement lambdas).

- void **Largest** (string1, string2, string3) {..} which displays the largest of three string values based on the length.

To test/call this method, you need to use built-in **Action<>** delegate predicate.