Project I: Software Implementation and Unit Testing

Assignment #2

Overview

This assignment should be done **individually**. Do your own work and **do not share** your work with others. Sharing work is an academic offense and is subject to penalty. Be aware that source code and documents are automatically checked by eConestoga against every other student's work in the course. Academic offenses will be reported to the College Registrar.

You are to create a series of tests for a C application that allows a user to work with and modify a rectangle. You will use GitHub as source control for this assignment.

What to do

Step 1: Setup GitHub and Visual Studio

Log into your GitHub.com account and create a repository for the base code by cloning the followingURL: https://github.com/gurpreet-conestoga/Assignment2_Project1.git

Open Visual Studio 2022. In the Get Started window, select Clone a repository.

For the Repository Location, use the URL just created in your personal GitHub account (DO NOT use the link above to clone your Visual Studio).

NOTE: If you need help performing the above tasks, refer to the instructions in Assignment #1.

You should be able to run the application now. Play around with it to see how it operates. The requirements of this program state:

- It only needs to remember one rectangle at a time
- Rectangle dimensions are to be measured using integers
- Both the length and width of the rectangle is to be limited to the range of values 1 to 99, inclusive.

In Visual Studio, add a **Native Unit Test Project** to the given solution. **Remember the naming conventions** for the Test Project, the Test Class, and the Test Methods. You will need to setup this Test Project to interact with the application project, as described in class.

Make regular commits to your personal repo during the course of this assignment, whenever it makes sense for your situation.

Step 2: Designing and Writing Unit Tests

Task 1

Write one unit tests for each of the following functions:

```
int getPerimeter(int *length, int *width);
int getArea(int *length, int *width);
```

The requirements of these functions is to return the perimeter and area for a rectangle with the given length and width dimensions. Since input is validated elsewhere in the program, these functions do not have a limit on the range of integers they allow. Each unit test is marked, based on the proper setup, execution, assert, naming and test choice. Cut and paste your test code in the box below:

```
#include "pch.h"
#include "CppUnitTest.h"

extern "C"   int getArea(int*, int*), getPerimeter(int*, int*);

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace FunctionTEST
{
	TEST_CLASS(FunctionTEST)
	{
	public:

		TEST_METHOD(Area)
		{
			//this test area functionality and result should give 2 * 2 = 4
			int Result = 0, l = 1, w = 1;
			Result = getArea(&l, &w);
			Assert::AreEqual(1, Result);
		}

		TEST_METHOD(Perimeter)
		{	//this test perimeter functionality and result should be 2(2+3) = 10
			int Result = 0, l = 2, w = 3;
			Result = getPerimeter(&l, &w);
			Assert::AreEqual(10, Result);
		}

		TEST_METHOD(TestMethod3)
		{
		}
	};
}
```

Task 2

Write three unit tests for each of the following functions:

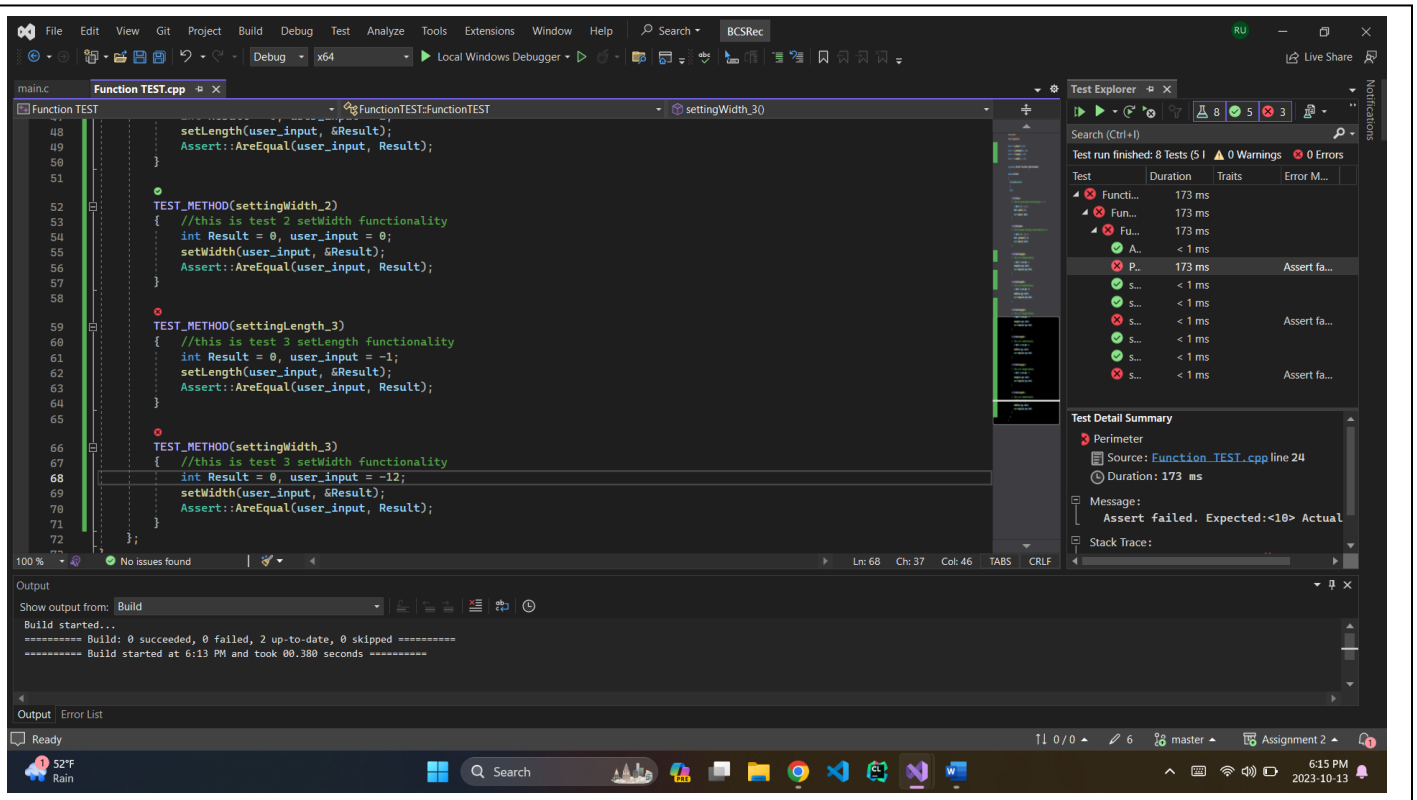
```
void setLength(int input, int *length);
void setWidth(int input, int *width);
```

The requirements of these functions is to set the length or width of a rectangle to be any integer from 1 to 99, inclusive. Each unit test is marked based on the proper setup, execution, assert, naming and test choice. Each unit test for the same function should attempt to test some different aspect of the function. Deductions will be given for tests that are testing similar inputs. For different inputs, think of usual, unusual, special or edge cases. Cut and paste your test code in the box below:

```
#include "pch.h"
#include "CppUnitTest.h"

extern "C" int getArea(int*, int*);
extern "C" int getPerimeter(int*, int*);
extern "C" int setLength(int, int*);
extern "C" int setWidth(int , int* );

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace FunctionTEST
{
	TEST_CLASS(FunctionTEST)
	{
	public:

		TEST_METHOD(Area)
		{	//this test area functionality and result should give 2 * 2 = 4
			int Result = 0, l = 1, w = 1;
			Result = getArea(&l, &w);
			Assert::AreEqual(1, Result);
		}

		TEST_METHOD(Perimeter)
		{	//this test perimeter functionality and result should be 2(2+3) =
10
			int Result = 0, l = 2, w = 3;
			Result = getPerimeter(&l, &w);
			Assert::AreEqual(10, Result);
		}

		TEST_METHOD(settingLength_1)
		{	//this is test 1 setLength functionality
			int Result = 0, user_input = 5;
			setLength(user_input, &Result);
			Assert::AreEqual(user_input, Result);
		}

		TEST_METHOD(settingWidth_1)
		{	//this is test 1 setWidth functionality
			int Result = 0, user_input = 99;
			setWidth(user_input, &Result);
			Assert::AreEqual(user_input, Result);
		}

		TEST_METHOD(settingLength_2)
		{	//this is test 2 setLength functionality
			int Result = 0, user_input = 2;
			setLength(user_input, &Result);
			Assert::AreEqual(user_input, Result);
		}

		TEST_METHOD(settingWidth_2)
		{	//this is test 2 setWidth functionality
			int Result = 0, user_input = 0;
			setWidth(user_input, &Result);
			Assert::AreEqual(user_input, Result);
		}

		TEST_METHOD(settingLength_3)
		{	//this is test 3 setLength functionality
			int Result = 0, user_input = -1;
			setLength(user_input, &Result);
			Assert::AreEqual(user_input, Result);
		}

		TEST_METHOD(settingWidth_3)
		{	//this is test 3 setWidth functionality
			int Result = 0, user_input = -12;
			setWidth(user_input, &Result);
			Assert::AreEqual(user_input, Result);
		}
	};
}
```

Task 3

Run your test suite of 8 unit tests against the application code. Include a screenshot of this run in the box below.



If any test fails, briefly explain why the test failed (1 to 2 sentences) and fix either the application or test code. If you had to fix any tests or the application, include the changed code with your explanation in the box below. Marks here are based on what issues were discovered and fixed, and how logical your explanations are if any are or are not found.

- The Perimeter function test failed because of the wrong formula written into the function.
  The changed main.c code:

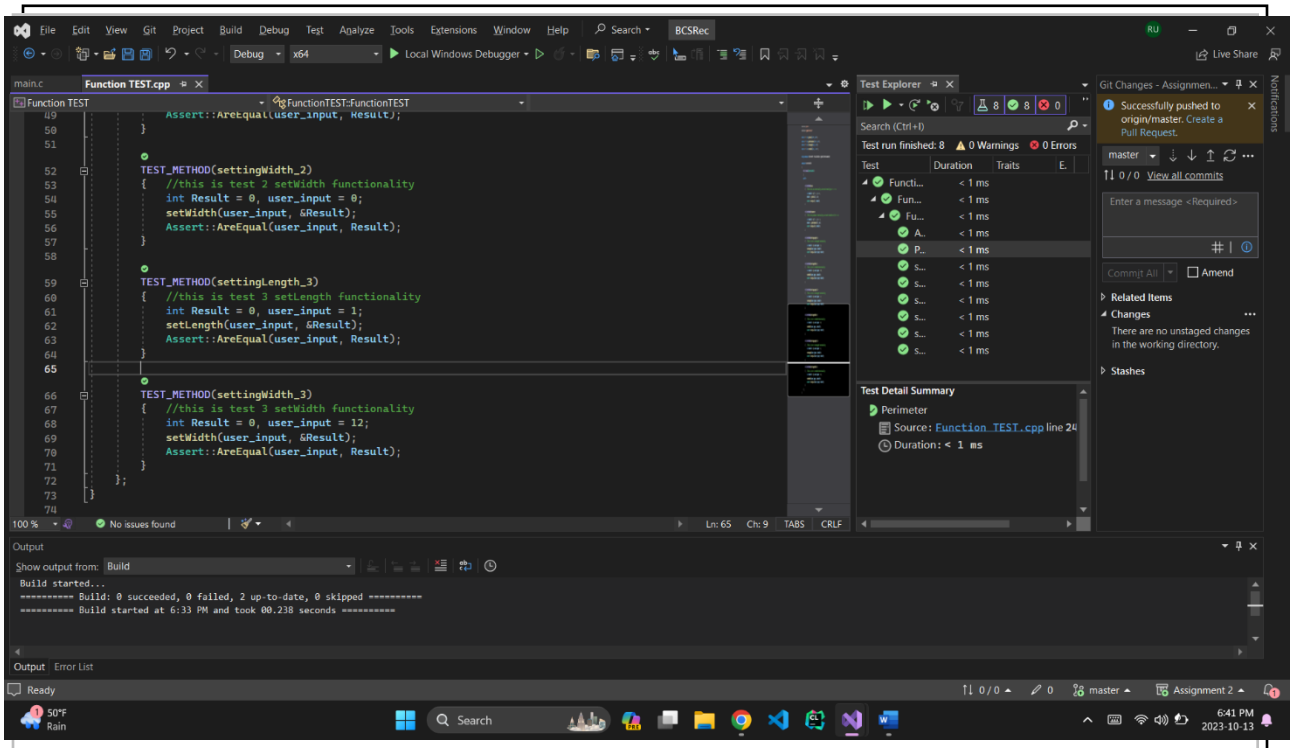
```
int getPerimeter(int *length, int *width) {
        int perimeter = 2*(*length + *width);
        return perimeter;
        }
```

- settingLength_3 & settingWidth_3 failed because the range of input is set from 0 to 99 and the user_input is in negative.

```
TEST_METHOD(settingLength_3)
{       //this is test 3 setLength functionality
        int Result = 0, user_input = 1;
        setLength(user_input, &Result);
        Assert::AreEqual(user_input, Result);
}

TEST_METHOD(settingWidth_3)
{       //this is test 3 setWidth functionality
        int Result = 0, user_input = 12;
        setWidth(user_input, &Result);
        Assert::AreEqual(user_input, Result);
        }
```

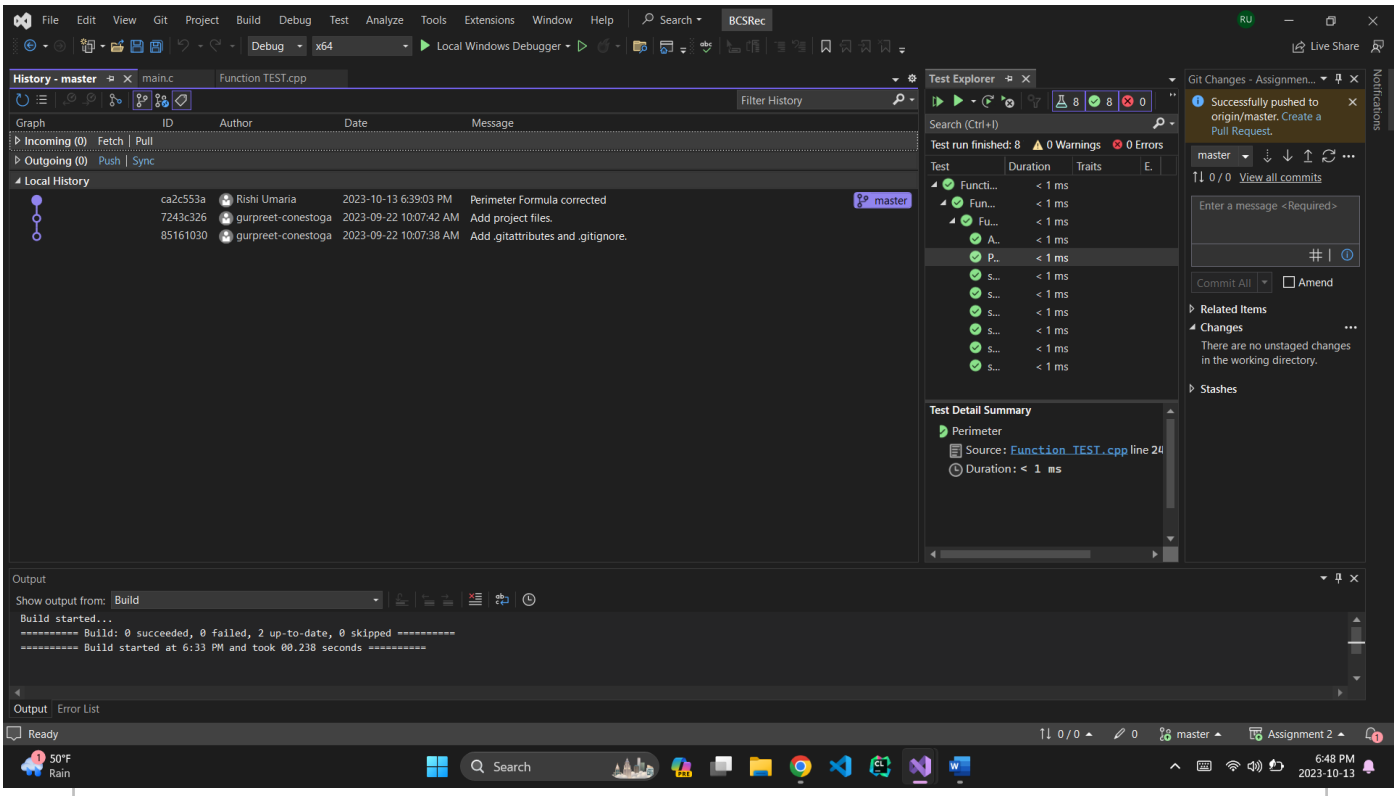
Here, perimeter formula was corrected with right formula i.e. 2*(length+width).
Thereafter, in test 3 of setLength & setWidth, the user_input was changed to positive numbers.
With this changes in test and application, now all the tests pass with success.

If you had any failing tests, after fixing the issue include another screenshot of all the tests passing in the box below:



Task 4

Push your local repo to your GitHub repository. Include a screenshot of your commits page in the box below:



Rubric

This assignment will be marked based on the following rubric information.

- PDF Form is complete and not re-formatted (2 marks)
- Task 1 – Quality and completeness of the two unit tests (4 marks)
- Task 2 – Quality and completeness of the six unit tests (4 marks)
- Task 3 – Execution of the 8 unit tests and explanations of failures (5 mark)
- Task 4 – Use of source control management (5 marks)

Total 20 marks.

What to Hand In

Upload the following files to eConestoga using the Assignment #2 link

- A copy of this PDF form properly filled out
- A single compressed (.zip format) archive file containing **the entire Solution folder** of your source code (so I can run it as required).

Do not reformat this form in any way. Just fill it out, save it and upload it. Modifying this form will result in a 10% penalty.