PROG2070 – Quality Assurance: Winter 2017

Assignment 03

[Maximum points: 50]

Due Date: Week of Mar. 20, 2017, in Class

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.

This assignment will be demonstrated in class. Any assignment not demonstrated in class will receive a 20% penalty. An up to 10% penalty is applied if no hard copy is handed in or handed in incomplete.

In this assignment we will move from unit testing, into integration testing. This assignment we will perform an iterative intra-system integration test using the top-down approach.

You are to create a C# console application that converts times. This program converts time between Seconds, Minutes, Hours and Days. Please create some sort of interface that allows the user to enter a time, what the time unit is, and what unit the user wishes to convert the time to. There should also be some sort of exit function. Any exceptions thrown do not crash the program. Once a conversion is complete, the program should present the user with the original convert interface (it should not exit after a single conversion).

In this Assignment, you will need to create a class as follow:

```
public static class TimeConversion
```

This class contains three methods. The methods should work as follows:

```
public static double Convert(double value, string convertFrom, string convertTo)
```

This method should be passed three variables collected within the Program.cs: a double representing the time value, and two strings representing the unit converting from and the unit converting to. Each string should be passed individually to the ModifyInput method. The Convert method should then pass the two ModifyInput results to the GetMulitplier method. The result of the GetMultiplier method should be multiplied by the time value to get a final result. This result should be returned from the Convert method to the Program.cs.

```
private static string ModifyInput(string input)
```

This method should convert the following:

`seconds`, `Seconds`, `s` or `S` becomes: `seconds`

`minutes`, `Minutes`, `m` or `M` becomes: `minutes`

`hours`, `Hours`, h or `H` becomes: `hours`

`days`, `Days`, `d` or `D` becomes: `days`

If the input is anything other than the valid 16 inputs listed above, the ModifyInput method should throw an `ArgumentException` with the message "Incorrect time unit". Be sure your Program.cs handles this correctly.

The ModifyInput method should return the converted string.

```
private static double GetMultiplier(string convertFrom, string convertTo)
```

The GetMultiplier method will check what the two strings passed to it are, and return the correct multiplier to convert from the convertFrom unit to the convertTo unit. This value should be returned as a double. Note: You are guaranteed that this method will only receive one of four inputs because the input has previously been checked by the ModifyInput method: `seconds`, `minutes`, `hours` or `days`.

Your completed system will be assembled in the following manner:

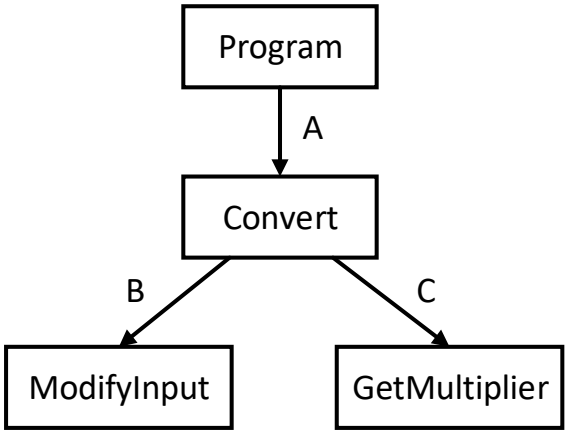

Figure 1

Do not validate the unit input within your console application Program class, as the ModifyInput method validates the unit input.

This assignment will require you to create three projects.

Project A:

This program is testing the interaction between Program and Convert, as shown on Edge A in Figure 1. This project should contain your completed Program. The Convert method within the TimeConversion class, modified so any method call it attempts calls a stub instead. You will need to create two methods within the class, called ModifyInputStub and GetMultiplierStub.

Write **four** integration test cases that test the interaction between Program and Convert. For each test case, record your input, your expected output, your observed output, and a one sentence explanation to why you chose this test.

Project B:

This program is testing the interaction between Convert and ModifyInput, as shown on Edge B in Figure 1. This project should contain your completed Program. The Convert method within the TimeConversion

class, modified so its attempt to call GetMultiplier calls a stub instead. You will need to create one methods within the class, called GetMultiplierStub.

Write **four** integration test cases that test the interaction between Convert and ModifyInput. For each test case, record your input, your expected output, your observed output, and a one sentence explanation to why you chose this test.

Project C:

This program is testing the interaction between Convert and GetMultiplier, as shown on Edge B in Figure 1. This project will also be testing the entire hierarchy (the completed system). This project should contain your completed Program, and the completed TimeConversion class.

Write **four** integration test cases that test the interaction between Convert and GetMultiplier. For each test case, record your input, your expected output, your observed output, and a one sentence explanation to why you chose this test.

Test cases in this assignment are to be written and observed, and can be placed in a chart (you do not need to program them, like we did for unit tests).

Please clearly label any print out, and stable them together in the order listed below. The format for submitting the assignment is as follows:

1. **Demonstrate your program in class.**
2. **Printouts Handed in Class**:
    a. Assignment Cover sheet properly filled (found on eConestoga).
    b. Assignment Rubric left blank (found on eConestoga).
    c. Print out of source code from Program and TimeConversion class in Project A.
    d. Print out of source code from Program and TimeConversion class in Project B.
    e. Print out of source code from Program and TimeConversion class in Project C.
    f. Print out containing your 12 test cases and the results from the tests. Be sure it is clearly labelled which tests ran during which projects.
3. **eConestoga Submission**: A single compressed (.zip format) archive file containing the three project folders of your source code (so I can run it) and the doc file with test cases and results.