Lab 2: Creating a Test Driver

Patrick Di Salvo

October 6, 2019

1 Assignment Overview

For the following assignment, you are provided 5 files. main.c, Calculator.c, TestCalculator.c, Calculator.h, and TestCalculator.h. The .c files will be kept in the src folder, the .h files will be kept in the include folder and the executable will be kept in the bin folder. You will be given all the files other than TestCalculator.c. Your job is to define the function declarations in TestCalculator.h in the file TestCalculator.c as well as create a Makefile that compiles and runs an executable. For this assignment, I will be checking your code.

The general goal of this assignment is as follows. You are given a calculator with four functions, add, subtract, multiply and divide. Your goal is to write a test harness that tests the functionality of the aforementioned functions of the calculator. The way you test the functions is by comparing the return value of the calculator functions with some expected value. This assignment is to test your unit testing abilities, and a unit is defined by a single function. Therefore, you will need to write a testing function for each of the functions in Calculator.c. The testing functions are declared in TestCalculator.h. Once you have written each of these testing functions you need to write a test harness, a.k.a., a driver, which calls the testing functions and indicates to the user the results of the testing functions.

The Makefile will need to compile and execute an executable file. I should be able to simply type make and see the results of your executable. The executable should be kept in the bin folder.

2 Assignment Details

A detailed requirement will be provided for each function/file you need to write.

2.1 Function 1: testAdd

int testAdd(double expectedResult, double a, double b, double(*add)(double a, double b));

This function takes in 3 doubles, expectedResult, a, and b, as well as a function pointer called add. The function will return 0 if the result of add(a, b) is the same as the expectedResult. Else it will return -1.

2.2 Function 2: testSubtract

int testSubtract(double expectedResult, double a, double b, double(*subtract)(double a, double b));

This function takes in 3 doubles, expectedResult, a, and b, as well as a function pointer called subtract. The function will return 0 if the result of subtract(a, b) is the same as the expectedResult. Else it will return -1.

2.3 Function 3: testMultiply

int testMultiply(double expectedResult, double a, double b, double(*multilpy)(double a, double b));
This function takes in 3 doubles, expectedResult, a, and b, as well as a function pointer called multiply. The function will return 0 if the result of multiply(a, b) is the same as the expectedResult. Else it will return -1.

2.4 Function 4: testDivide

int testDivide(double expectedResult, double a, double b, double(*divide)(double a, double b));
This function takes in 3 doubles, expectedResult, a, and b, as well as a function pointer called multiply. The function will return 0 if the result of divide(a, b) is the same as the expectedResult. Else it will return -1.

2.5 Function 5: driver

void driver(double expectedSum, double expcetedDifference, double expectedProduct, double expectedQuotient, double a, double b);

*The driver is the main test harness you will need to write.

*The driver must create a calculator object.

*The driver calls the testAdd, testSubtract, testMultiply and testDivide with the values a, b, expctedSum, expectedDifference, expectedProduct and expectedQuotient respectively.

*The driver must use the calculator object's function pointers when passing the calculator functions to the testing functions.

*The driver must print to the terminal the values of a and b before calling of the test functions.

*The driver must print to the terminal the values of expectedSum, expectedDifference, expectedProduct and expectedQuotient before calling of the test functions.

*Before calling any particular test function in the driver, the driver must display to terminal what test function is being called.

*The driver must have 2 goto statements per test function. One for success and one for failure. If the function succeeds, goto the success label for that function and print that it succeeded. If the function fails, goto the fail label and print that it failed. You will need 8 goto statements in total with 8 unique goto labels.

*The details for the goto outputs are as follows. Every printf for every goto statement must print which function succeeded or failed. These are the functions in Calculator.c

*Every prntf statement in every goto label must display the expected result as well as the result from the function being tested in Calculator.c. For example, if the function testAdd returns 0 with an expected value of 3 and an 'a' value of 1 and a 'b' value of 2, then the driver must go to a label and the label must have a printf statement that displays the following information: The expected value, 3, the results of add(1, 2), and the fact that the function add succeeded.

*Every result of every printf call associated with a success label must be displayed in green to the terminal. Every result of every printf call associated with a failure label must be displayed in red to the terminal.

2.6 File 1: main.c

In your main, simply call driver with the values you want. You will need to call the driver multiple times. Once such that the driver succeeds for all cases, once such that the testAdd function fails, once such that the testSubtract function fails, once such that the testMultiply function fails and once such that the testDivide function fails.

2.7 File 2: Makefile

The makefile must compile the .c files into an executable. The executable must be kept in the bin folder. The executable must be called lab2. I must be able to type make in the terminal and see the results from the calls to driver in your main.c. Basically, I should be able to type make and grade your assignment, I should not have to run the

executable separately. The Makefile must work on a linux environment. The Makefile must be kept in the main folder of the assignment called Lab2

3 Grading

The entire lab is worth 2 % of your final grade.

The functions testAdd, testSubtract, testMultiply and testDivide are each worth 0.25% . The grading is all or nothing, there are no part marks for these functions.

The function driver is worth 0.5 % . The grading is all or nothing, there are no part marks for this function. You must implement every requirement for the function that is listed above.

The makefile is worth 0.5 %.The grading is all or nothing, there are no part marks for the Makefile.

4 Submission

Submit your assignment as a .tar file. Your assignment must be named LastnameFirstnameLab2.tar. For example, my submission would be called DisalvoPatrickLab2.tar Do not change any of the folder structure in the assignment. Put your Makefile in the main folder which is called Lab2.