Team International
Deliverable #3
Project: Celestia
10.21.2016
Members: Gui Costa, Megan Landau, Tony Tang

Task:

Design and build an automated testing framework that you will use to implement your test plan per the Project Specifications document.

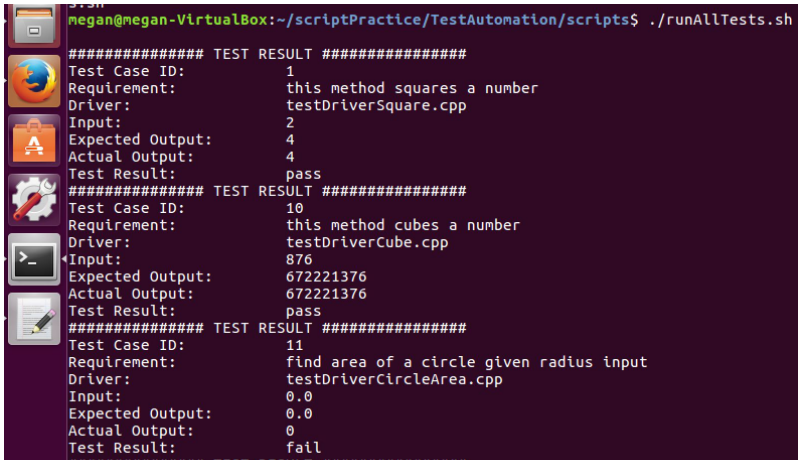
Report Task and Framework Description:

- Description of the task:
    - The task is to build a testing framework using a structure that separates the test case text files into one folder (TestCases); the program executables in another (TestCasesExecutables); and the main script and any helper scripts in another(Scripts). The main script is all that needs to be run via the command line and will output a report in the form of an .html file to the Reports folder.
- Architectural description of project framework:
    - The project's framework was organized by partitioning the test cases, scripts, reports, and executables in different folders. This type of architectural allowed the project to have a uniform structure. The test script inside the scripts folder could easily navigate through other directories by using the command "../directory" and then perform the task needed in that directory.

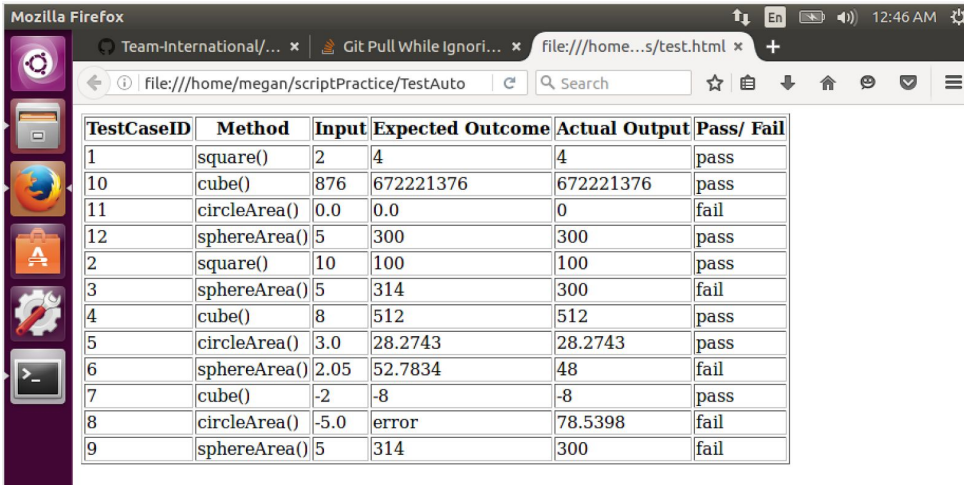
Team experience:

- The team met in three stages: first, we met to discuss the task and come up with a plan for how the testing framework would be implemented and how we could accomplish the above task. Our first challenge was creating the test case .txt files, which we were unsure how to parse using a bash script. After deciding the format for the test case files, we created a few based on our work from Deliverable #2.
- We had already created a .cpp driver for the square() method and knew how to compile and run it from the command line. So our next step was to write a bash script that would parse a testCase.txt file, make a variable out of line 15 (which is the **Input** value) and then pass that variable to the compiled driver.
- Our next struggle was to receive a return value back from the driver which we could then store into a variable in order to compare the **Actual Output** with the **Expected Output**. We discussed the problem with Dr. Bowring and he suggested we use the driver to create a temporary .txt file and then read that text file and compare the values using this procedure. We believed, however, that there had to be a way to create a variable from the return value of the driver and use the "==" equals comparator to determine any difference in the two values.
- After successfully figuring this out, the team was able to determine whether a single testCase.txt file had a pass / fail outcome.

- With this achieved, the team then used a for loop to run through all of the testCase.txt files within the **testCases** folder of the framework. The script would then output all the stored variables read from the testCase.txt file and the results of the test. This is shown below:



- We are still having some troubles with conversions between Int and Float values (shown in Test Case ID 11 above) and we hope to resolve this in order to have our successful test cases pass if they should be passing.
- Our next challenge was to append the results of the tests to an .html file which would open a browser window displaying a table of the pass / fail outcomes. First we coded echos and learned how to create a simple table. We then coded a loop so we could append multiple rows to the table and finally, we were able to populate the table with actual values gleaned from the testCase.txt files.
- Then we used what we learned from the first bash script we wrote months ago (myList.sh) to command the .html file to open up in a browser window:



TestCaseID	Method	Input	Expected Outcome	Actual Output	Pass/ Fail
1	square()	2	4	4	pass
10	cube()	876	672221376	672221376	pass
11	circleArea()	0.0	0.0	0	fail
12	sphereArea()	5	300	300	pass
2	square()	10	100	100	pass
3	sphereArea()	5	314	300	fail
4	cube()	8	512	512	pass
5	circleArea()	3.0	28.2743	28.2743	pass
6	sphereArea()	2.05	52.7834	48	fail
7	cube()	-2	-8	-8	pass
8	circleArea()	-5.0	error	78.5398	fail
9	sphereArea()	5	314	300	fail

- Success! Our next steps will be to have the test cases output to the .html file in order so that the table is easier to read. We will also be adding text to the .html file so that it explains what the table is showing, gives an explanation of the Celestia project and some information about

who we are and why we are testing it. We are also planning on formatting the .html file so that it is more visually appealing. We also have a tentative plan to order the table information by test suites (suites will be based on grouping the test cases together according to the method they will be testing).

- Furthermore, we plan on adding different types of test cases that test other components from the Celestia project besides the mathlib.h library. We also need to add at least 13 more test cases, which should be relatively easy now since we have a system in place. This means we will also be creating two more driver .cpp files in order to test other components or methods.
- Overall, the team is very excited about the distance we have come since beginning this project. We all started out with minimal experience in shell scripting and little to no experience with GitHub and now we have a script that navigates our testing framework and also almost 100 commits to our GitHub repository. We have written in bash and C++ and have been able to compile and run programs from the terminal, which none of us had been adept at before this semester. So, overall, we are very happy with our progress and seem to be working very efficiently and are content with the amount of collaboration amongst our team members.

How-To Documentation:

The following information provides steps for how to run the automated script named **runAllTests.sh.** These steps explain how to download and run this shell command using the command line in a **Linux OS**.

1. First, click on the clone button on the Team-International repository on Github, the button is named [Clone with HTTPS]

2. Copy the file path for the repository

3. Next, enter git clone followed by the name of the file path to clone the Team-International repository from Github

```
git clone https://github.com/CSCI-362-02-2016/Team-International.git
```

4. Then, make a folder and name it something, for example teaminternationalrepo

5. This will be where the Team-International github repository will be located on your computer

6. The following are the instructions needed in order to run the TestAutomation script

7. Next, type git pull to obtain the current working repository from Github

```
git pull
```

8. Type in your username

9. Type in your password

10. Press Enter

Now, you are ready to run the automated script. In order to run **runAllTests.sh**, you will need to navigate to the script folder located in the TestAutomation directory with the command line.

11. Type “ls” in the terminal to display the files in the current directory. You will see a project folder named “scripts”

12. Inside this folder, enter **./runAllTests.sh** to run the script

```
./runAllTests.sh
```

At this point you should see the test cases printed in the terminal as they are run and a browser window should open with the test results displayed in an HTML table.