Brielan Beamon: beamonbh@g.cofc.edu
Logan Smith: smithla1@g.cofc.edu
Scott White: whites2@g.cofc.edu

Dr. Bowring
CSCI 362
September 27, 2016

Test Plan

Introduction

Our project is centered around Sugar Labs, a free and open source software that focuses on allowing people around the world to learn more about the world around them for free. Sugar Labs is a spin off of One Laptop Per Child, constituting a simple Linux-based operating system. Due to the complexity of emulating and then testing an entire operating system, we have opted to test individual components of Sugar Labs — which are referred to as activities.

The Testing Process

Our testing shall focus on the calculator activity of Sugar-Labs. We plan to observe the behavior of that particular activity and verify that it functions as it should. In order to do this, we plan to write a variety of test cases that establish correct behavior on both normal and fringe cases.

Requirements Traceability

As Sugar Labs is intended for people without access to other methods of learning, it is imperative that the information provided by Sugar Labs is accurate. For a person engaged in learning, nothing can be more damaging than viewing and committing to memory false information and knowledge. In addition, those using Sugar Labs may have no other way to verify what they learn through the activities therein. It is with that in mind that we come to the most important requirement (that we will trace our test cases too) for Sugar Labs: accuracy/reliability.

Tested Items

We will be testing the calculator activity. More specifically, we will be testing the file functions.py, which handles all the calculations for the calculator.

At this point, the individual methods we are testing include:
pow(x, y) — returns x raised to the y power
add(x, y) — returns x + y
mul(x, y) — returns x * y
div(x, y) — returns x / y
sqrt(x) — returns √x

Test Case Format

To insure consistency in the testing procedure, and to allow for adding any number of test cases, all test cases to be used by the project should adhere to the following specifications:

1. Name and type of the File
    a. File name: testCase*.txt
        i. The asterisk represents a unique number, which will allow for identification of that particular test case
    b. File type: text
        i. Only text (.txt) files will be interpreted as test cases and run accordingly.
2. Structure of File Contents
    a. The following sections must be included in the following order, and on it's own line, for the test case to be used successfully:

```
Test Case ID: *1*

    Requirement being tested:
        *2*

    Component being tested:
        *3* - *4*
        Path to file: *5*

    Method being tested: *6*

    Test input(s) including command-line arguments: *7*

    Expected outcome(s): *8*
```

Placeholder	Necessary Information	Notes
1	Test case ID number	Must match number in file name
2	Pertinent requirement	Each test case must be traceable to a requirement
3	Sugar Labs activity name	

Placeholder	Necessary Information	Notes
4	Activity python file	
5	The path to the activity python file	This path will be from the home directory of the project formatted as “dir1/dir2/...dir*n*/file.py”
6	The method to be tested	Includes entire method signature
7	Parameters for the method call	
8	Expected outcome	

Testing Schedule

We will be developing a testing framework over the course of this academic semester. We plan to meet progress deadlines on the following dates:

October 18
Submit an architectural description of our framework, full how-to documentation, a set of at least 5 of the eventual 25 test cases.

November 10
Complete the design and implementation of our testing framework as specified in Deliverable #3, and create 25 test cases that our framework will automatically use to test our H/FOSS project (Sugar Labs).

November 22
Design and inject 5 faults into the code we are testing that will cause at least 5 tests to fail, but hopefully not all tests to fail. Report the results of running the testing framework.

Test Recording Procedures

To record the results of our tests, we will aggregate the following information about each test case: test case identification, method signature, associated input, expected output, received output, and whether or not the test case passed or failed. This data will be aggregated in a professional, easy to read html file stored in a subdirectory of the project directory. This test report will contain the following sections:

TestCaseID, Tested Method, Input, Expected Output, Actual Output, and Pass/Fail.

The section TestCaseID will contain a hyperlink to the individual test case file, allowing for further viewing of information pertinent to the test case.

Hardware and Software Requirements

The required hardware for our testing is simply a working computer with available keyboard input and a monitor to view results.
For software, a current version of Linux is required (Ubuntu is suggested/preferred) along with the following dependencies: python2.7, sugar3, pygame, python-gtk2, python-gobject, and gob2.

Constraints

At the moment, there are no know constraints affecting the progress of our testing.