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

Dr. Bowring
CSCI 362
November 15, 2016

Automated Testing Framework

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.

Tested Artifacts

The following test cases are the 25 test cases we have written for this project. With these 25 test cases, we attempt to verify that the calculator activity functions properly. We decided to look at five functions: pow(a, b), add(a, b), div(a, b), sqrt(a), and cos(a). For pow and add, we proposed a wide scope to verify that the functions yielded correct results for all types of input. For div and sqrt, our focus was two pronged. We wanted to verify that the functions yielded correct results for valid input values, but we also wanted to verify that the functions behaved properly when presented with certain inputs. As an example, it is impossible to divide a number by zero, and when doing so the function should give us a divide by zero exception. In a similar vein, it is impossible to take the square root of a negative number (without using imaginary numbers). This is in line with how the two functions should perform, so we wanted to insure that not only would these two functions succeed where they should; they should also fail where they are expected to as well.

Test Case 1	
Requirement to Test	Correct and accurate exponentiation
Component to Test	functions.py
Method to be Tested	pow(a, b)
Input	2, 10
Expected Output	1024

Test Case 2	
Requirement to Test	Correct and accurate exponentiation
Component to Test	functions.py
Method to be Tested	pow(a, b)
Input	4.0, 2.5
Expected Output	32.0

Test Case 3	
Requirement to Test	Correct and accurate exponentiation
Component to Test	functions.py
Method to be Tested	pow(a, b)
Input	16, 0.5
Expected Output	4.0

Test Case 4	
Requirement to Test	Correct and accurate exponentiation
Component to Test	functions.py
Method to be Tested	pow(a, b)
Input	16.0, -0.5
Expected Output	0.25

Test Case 5	
Requirement to Test	Correct and accurate exponentiation
Component to Test	functions.py
Method to be Tested	pow(a, b)
Input	0, 5
Expected Output	0

Test Case 6	
Requirement to Test	Correct and accurate addition
Component to Test	functions.py
Method to be Tested	add(a, b)
Input	20, 5
Expected Output	25

Test Case 7	
Requirement to Test	Correct and accurate addition
Component to Test	functions.py
Method to be Tested	add(a, b)
Input	-5, -10
Expected Output	-15

Test Case 8	
Requirement to Test	Correct and accurate addition
Component to Test	functions.py
Method to be Tested	add(a, b)
Input	-40, 5
Expected Output	-35

Test Case 9	
Requirement to Test	Correct and accurate addition
Component to Test	functions.py
Method to be Tested	add(a, b)
Input	12.5, 5
Expected Output	17.5

Test Case 10	
Requirement to Test	Correct and accurate addition
Component to Test	functions.py
Method to be Tested	add(a, b)
Input	4.5, 7.25
Expected Output	11.75

Test Case 11	
Requirement to Test	Correct and accurate division
Component to Test	functions.py
Method to be Tested	div(a, b)
Input	120, 8
Expected Output	15

Test Case 12	
Requirement to Test	Correct and accurate division
Component to Test	functions.py
Method to be Tested	div(a, b)
Input	25.8, 3.0
Expected Output	8.6

Test Case 13	
Requirement to Test	Correct and accurate division
Component to Test	functions.py
Method to be Tested	div(a, b)
Input	28.4, -4
Expected Output	-7.1

Test Case 14	
Requirement to Test	Correct and accurate division
Component to Test	functions.py
Method to be Tested	div(a, b)
Input	-12, -6
Expected Output	2

Test Case 15	
Requirement to Test	Correct and accurate division
Component to Test	functions.py
Method to be Tested	div(a, b)
Input	120, 0
Expected Output	ERROR - “Can not divide by zero”

Test Case 16	
Requirement to Test	Accurate calculation of square roots
Component to Test	functions.py
Method to be Tested	sqrt(a)
Input	16
Expected Output	4.0

Test Case 17	
Requirement to Test	Accurate calculation of square roots
Component to Test	functions.py
Method to be Tested	sqrt(a)
Input	9.0
Expected Output	3.0

Test Case 18	
Requirement to Test	Accurate calculation of square roots
Component to Test	functions.py
Method to be Tested	sqrt(a)
Input	0.00000001
Expected Output	0.0001

Test Case 19	
Requirement to Test	Accurate calculation of square roots
Component to Test	functions.py
Method to be Tested	sqrt(a)
Input	0
Expected Output	0.0

Test Case 20	
Requirement to Test	Accurate calculation of square roots
Component to Test	functions.py
Method to be Tested	sqrt(a)
Input	-4
Expected Output	ERROR - “math domain error”

Test Case 21	
Requirement to Test	Accurate determination of cosine values
Component to Test	functions.py
Method to be Tested	cos(a)
Input	-3.141592653599790
Expected Output	-1.0

Test Case 22	
Requirement to Test	Accurate determination of cosine values
Component to Test	functions.py
Method to be Tested	cos(a)
Input	0
Expected Output	1.0

Test Case 23	
Requirement to Test	Accurate determination of cosine values
Component to Test	functions.py
Method to be Tested	cos(a)
Input	3.14159265359979
Expected Output	-1.0

Test Case 24	
Requirement to Test	Accurate determination of cosine values
Component to Test	functions.py
Method to be Tested	cos(a)
Input	6.283185307179590
Expected Output	1.0

Test Case 25	
Requirement to Test	Accurate determination of cosine values
Component to Test	functions.py
Method to be Tested	cos(a)
Input	12.566370614359200
Expected Output	1.0