Calc-U-Later Testing Documents

*Note that BigDecimal produces a number of arbitrary precision, thus the maximum or minimum number that it can produce is limited to the user's memory allocated. As such these test cases will not be provided since they will vary from user to user.

Test cases will be provided in the following format:

The button press being tested	
User Input sequence	*Expected output displayed*
Respective user input sequence	*Respective output*
Explanation of what the respective test was testing for	
…	
Final discussion as to why tests were derived like they were, and what the intended purpose of the tests were to uncover.	

All Buttons	
Input	**Output**
1	1
Testing that pressing number one produces a one on screen in the User input field	
2	2
Testing that pressing number two produces a two on screen in the User input field	

3	3
Testing that pressing number three produces a three on screen in the User input field	
4	4
Testing that pressing number four produces a four on screen in the User input field	
5	5
Testing that pressing number five produces a five on screen in the User input field	
6	6
Testing that pressing number six produces a six on screen in the User input field	
7	7
Testing that pressing number seven produces a seven on screen in the User input field	
8	8
Testing that pressing number eight produces an eight on screen in the User input field	
9	9
Testing that pressing number nine produces a nine on screen in the User input field	

0	0
Testing that pressing number zero produces a zero on screen in the User input field	
π	3.14159
Testing that pressing π produces a 3.14159 on screen in the User input field	
#, Sin	Sin
Testing that pressing sin produces sin on screen in the User input field	
#, Cos	Cos
Testing that pressing cos produces cos on screen in the User input field	
+/-	-# or #
Testing that pressing the sign change operator appropriately changes the value on screen to either a negative or a positive.	
.	#. or .
Testing that pressing the . button correctly places a decimal in the user input field on screen	
3, !	6
Testing to ensure that factorials are produced correctly on screen	

0, Enter	0
Testing to make sure that the Enter button pushes the appropriate number to the Calculated Value Display as well as the history display.	
0, Enter, Clear	Start a new calculation
Testing to ensure that the Clear button removes all text from both the user input field, as well as the Calculated value field. History field should display “Start a new calculation”.	
These test have been derived to ensure that all buttons on the calculator function as desired. Each button should produce a result on screen to the user so they know that it is working as intended. These tests cover all metrics in this regard as each button has its own respective test to ensure completeness.	

X	
Input	Expected Output
2, 10, X	20
Testing to see if basic multiplication produces desired result.	
0, 10, X	0
Testing to see if a number multiplied by zero equals 0	
10,0.5, X	5
Testing to see if a number multiplied by a fraction produces correct result.	

5, 1, X	5
Testing to see if a number multiplied by one equals itself.	
5, -1, X	-5
Testing to see if a positive number multiplied by a negative number results in a negative number	
-3, -2, X	6
Testing to see if a negative number multiplied by a negative number results in a positive number	
These tests cover all possibilities of multiplication. They span both the negative and positive real numbers to ensure results are accurate and intended.	

/	
Input	Expected Output
16, 4, /	4
Testing to see if basic division produces correct result	
0,4, /	0
Testing to see if zero divided by a number results in 0	
3, 0, /	Error – Unable to divide by zero!

Testing to see if divide by zero is handled appropriately without crashing the program.	
1, 3, /	0.33333
Testing to see if infinite digits are handled without crashing program as well as ensuring results are to desired accuracy.	
1, 2, /	0.5
Testing to see if division into an exact fraction produces correct result.	
0.5, 0.5, /	1
Testing if division by two fractions produces correct result.	
10, 0.5, /	20
Testing if division of a whole number by a fraction produces correct result.	
7, -1, /	-7
Testing to ensure a positive number divided by a negative number produces a negative result.	
These tests cover all division scenarios across both the positive and negative real numbers to ensure results are to established accuracy and work as intended.	

+	
Input	Expected Output

3, 2, +	5
Testing to see if basic addition of two positive numbers produces correct result	
5, -1, +	4
Testing to see if addition of a positive number and a negative number produces correct result	
-1, -9, +	-10
Testing to see if addition of two negative numbers produces correct result	
8, 0, +	8
Testing to see if addition of zero results in the same number	
These tests cover all addition scenarios across both positive and negative real numbers to ensure the sums produced are accurate.	

-	
Input	Expected Output
3, 2, -	1
Testing to see if basic subtraction of two positive numbers produces correct result	
5, -1, -	6

Testing if subtraction of a positive number and a negative number produces correct result.	
-3, -2, -	-1
Testing if subtraction of two negative numbers produces correct result.	
3, 0, -	3
Testing if subtraction of zero results in the same number being produced.	
These tests cover all subtraction scenarios across both positive and negative real numbers to ensure the differences produced are accurate.	

Sin	
Input	Expected Output
π, sin	0.00000
Ensuring sin produces a result that is within the agreed upon accuracy	
π, 2, /, sin	0.99999
Ensuring sin produces a result within the agreed upon accuracy	
These tests are to ensure that sin produces a value that is expected. These values are not 100% exact but are within the agreed upon tolerances specified.	

Cos	
Input	Expected Output
π, cos	-0.99999
Ensuring sin produces a result that is within the agreed upon accuracy	
π, 2, /, cos	0.00000
Ensuring cos produces a result within the agreed upon accuracy	
These tests are to ensure that cos produces a value that is expected. These values are not 100% exact but are within the agreed upon tolerances specified.	