<Company Name>

Calculator Project

Test Case

Version <1.0>

Revision History

Date	Version	Description	Author
12/03/2023	1.0	Specify Test Cases for Calculator Project	Sean Brady, Chris Harvey, Kristin Boeckmann, Naran Bat, Peter Walsdorf, William Morris

Table of Contents

Test Case

1. Purpose

The purpose of the test suite outlined below is to ensure that every aspect and individual module of our application is functioning as expected. Tests range from comically simple (1 + 2 = 3) up to complex mathematical expressions containing nested parentheses to ensure that the application functions correctly for any approved use-case.

Test Case Identifier	**Item Tested**	**Inputs**	**Outputs**
test_basic_addition	Addition Operation, Parser	“1 + 2”	3
test_basic_subtraction	Subtraction Operation, Parser	“2 - 1”	1
test_basic_negative_number	Addition Operation, Parser	“-1 +2”	1
test_basic_multiplication	Multiplication Operation, Parser	“2 * 3”	6
test_basic_mulplication_negative	Multiplication Operation, Parser	“2 * -3”	-6
test_large_multiplication	Multiplication Operation, Parser, Larger numbers	“10000 * 10000”	100000000
test_basic_division	Division Operation, Parser	“2 / 4”	0.5
test_basic_division_negative_1	Division Operation, Parser	“-4 /16”	-0.25
test_basic_division_negative_2	Division Operation, Parser	“4 / -16”	-0.25
test_divide_by_zero	Division Operation, Zero Division Error Handling, Parser	“3 / 0”	DIVIDED BY ZERO
test_basic_modulus	Modulus Operation, Parser	“3 % 4”	1
test_modulus_by_zero	Modulus Operation, Zero Division Error Handling, Parser	“3 % 0”	DIVIDED BY ZERO
test_basic_exponentiation	Exponentiation Operation,	“2^3”	8
test_basic_exponentiation	Exponentiation Operation,	“2**4”	16

test_parenthesis	Parenthesis Operation, Multiply Operation, Add Operation	“(1 + 2) * 3”	9
test_parenthesis_1	Parenthesis Operation, Multiply Operation, Add Operation	“(3 * 2) + 1”	7
test_parenthesis_2	Parenthesis Operation, Subtraction Operation, Add Operation	“8 - (5 - 2)”	5
test_nested_parentheses	Nested Parentheses with Exponents	“(((2 ^ (1 + 1)) + ((3 - 1) ^ 2)) / ((4 / 2) % 3))”	4
test_combination_parentheses	Combination of Extraneous and Necessary Parentheses	“(((((5 - 3))) * (((2 + 1))) + ((2 * 3))))”	12
test_extraneous_parentheses	Extraneous Parentheses with Division	“((9 + 6)) / ((3 * 1) / (((2 + 2))) - 1)”	-60
test_unary_operators	Combining Unary Operators with Arithmetic Operators	“+(-2) * (-3) - ((-4) / (+5))”	6.8
test_unary_negation_addition	Combining Unary Operators with Arithmetic Operators	“-(-1) + (+2)”	1
test_negation_addition_negated	Negation and addition with Negated Parentheses	“-(-(-3)) + (-4) + (+5)”	-2
test_unary_negation_exponentiation	Unary Negation and Exponentiation	“+2 ^ (-3)”	0.125
test_unbalanced_parenthesis_error	Parenthesis Operation, Unbalanced Parenthesis Error Handling, Multiply Operation, Add Operation,	“(1 + 2))”	UNBALANCED PARENTHESIS
test_invalid_operators_without_operands	Operators Without Operands	“* 5 + 2”	OPERATORS WITHOUT OPERANDS
test_invalid_missing_operator	Missing Operator	“5 (2 + 3)”	MISSING OPERATOR
test_invalid_characters	Invalid Characters	“7 & 3”	INVALID CHARACTER/S
test_invalid_missing_operand	Missing Operand	“((4 * 2) + (-))”	MISSING OPERAND/S

test_complex_expression_1	All but Mod	“3 ^ 3 / (5 - 4) - 29”	-2
test_negative_complex_expression_1	Most	“-29 + 3 ^ 3 / (5 - 4)”	-2
test_lots_of_whitespace	Addition, Parenthesis	“ 1 + (3+2)”	6
test_variable_assignment	Multiplication, Variable Assignment	“A= 2” “3 * A”	Variable A Assigned 6
test_variable_assignment_1	Addition, Variable Assignment	“A=2” “B=3” “A + B”	Variable A Assigned Variable B Assigned 5
test_whitespace_variable_assignment	Addition, Variable Assignment, Parser	“ A = 2” “ 5 + A”	Variable A Assigned 7
test_variable_assignment_error_1	Variable Assignment Error Handling	“3 * A”	Unknown Variable A
test_variable_assignment_error_2	Variable Assignment Error Handling	“A = 2” “A ^ B”	Unknown Variable B
test_variable_assignment_nested_error_1	Divide by Zero Error Handling, Variable Assignment	“A=1” “B=0” “ A / B”	Variable A Assigned Variable B Assigned DIVIDED BY ZERO
test_variable_assignment_nested_error_2	Divide by Zero Error Handling, Variable Assignment	“A=1” “B=0” “ A % B”	Variable A Assigned Variable B Assigned DIVIDED BY ZERO

2. Test case identifier

Test case identifiers were chosen with the test cases most important operation in mind. All identifiers are valid function names in C++, so that they can be implemented directly and easily searched. All identifiers are included in the above table.

3. Test item

Test items are tied to modules outlined in the Software Requirements document and errors in the error handling section. Most test cases above test multiple items, so multiple combinations of each item were utilized to provide a more complete test suite.

4. Input specifications

Each string in the Input Specification column of the above table was written to be directly inputted into the application prompt (without surrounding double quotes)

5. Output specifications

Each string in the Output Specification column of the above table was written to directly represent the expected output of the inputs, line by line (without surrounding double quotes).