Kansas Instruments

Doing Math With C++

Test Cases

Version 1.0

Revision History

Date	Version	Description	Author
<02/12/2023>	<1.0>	Added all the test cases	Eric Loseke, Achinth Ulagapperoli, Will Hedges, Manish Singh

Table of Contents

Test Case

1. Purpose

This Test Case Specification document for the Doint Math with C++ defines a test case for an item that should be tested.

2. Test case identifiers, test items, input specifications, output specifications

2. Test case identifier	3. Test Case Description	4. Test Data	5. Expected Results	5. Actual Results	5. Pass/ Fail
Add 101	simple addition	1+2	3	3	Pass
Sub 101	simple subtraction	2-1	1	1	Pass
Sub 102	simple subtraction with negative result	1-2	-1	-1	Pass
Mult 101	simple multiplication	1*2	2	2	Pass
Mult 102	multiplication with negative numbers	-2*3	-6	-6	Pass
Mult 103	multiplication with negative numbers order flipped	3*-2	-6	-6	Pass
Div 101	simple division	4/2	2	2	Pass
Div 102	simple division with decimal answer	1/2	0.5	0.5	Pass
Div 103	simple division with negative numbers	-6/3	-2	-2	Pass
Div 104	simple division with negative numbers order flipped	6/-3	-2	-2	Pass
Spaces 101	testing spaces	2 + 2	4	4	Pass
Modulo 101	simple modulo with no remainder	6%2	0	0	Pass
Modulo 102	simple modulo with remainder	6%4	2	2	Pass

Modulo 103	modulo with negative numbers	-6%5	4	4	Pass
Modulo 104	modulo with negative numbers but flipped	8%-6	-4	-4	Pass
Exponents 101	simple exponent	2^3	8	8	Pass
Exponents 102	exponent with negative	2^-3	0.125	0.125	Pass
Parenthesis 101	simple parenthesis	(1+1)+(3+2)	7	7	Pass
Parenthesis 102	parenthesis with negative	8 - (5 - 2)	5	5	Pass
MISC 101	The multiplication and division operators are applied from left to right, resulting in the final answer.	10 * 2 / 5	4	4	Pass
Mixed Operators101	This expression combines multiple operators and parentheses to correctly calculate the result step by step.	4 * (3 + 2) % 7 - 1	5	5	Pass
Complex Addition with Extraneous Parentheses	While there are multiple sets of extraneous parentheses, they do not affect the validity of the expression. The addition is performed correctly.	(((2 + 3))) + (((1 + 2)))	8	8	Pass
Mixed Operators with Extraneous	This expression combines various operators with	((5 * 2) - ((3 / 1) + ((4 % 3))))	6	6	Pass

Parentheses	multiple sets of extraneous parentheses, but they do not change the order of operations or the final result.				
Nested Parentheses with Exponents	This expression includes nested parentheses and exponentiation, creating complexity, but it adheres to the correct order of operations.	(((2 ^ (1 + 1)) + ((3 - 1) ^ 2)) / ((4 / 2) % 3))	4	4	Pass
Combination of Extraneous and Necessary Parentheses:	This expression includes both extraneous parentheses and necessary parentheses to clarify the order of operations. It evaluates correctly.	(((((5 - 3))) * (((2 + 1))) + ((2 * 3))))	12	12	Pass
Extraneous Parentheses with Division	Extraneous parentheses are added for clarity, but they do not affect the validity of the expression. The division, multiplication, and subtraction are performed correctly	((9 + 6)) / ((3 * 1) / (((2 + 2))) - 1)	-60	-60	Pass
Multiplication Using Parentheses	This expression demonstrates using parentheses	5 (2 + 3)	25	25	Pass
Unmatched Parentheses	This expression has unmatched opening and	2 * (4 + 3 - 1	Handled error	Handled error	Pass

	closing parentheses, making it invalid.				
Operators Without Operands	The * operator lacks operands on the left, making the expression invalid.	* 5 + 2	Handled error	Handled error	Pass
Incorrect Operator Usage	Division by zero is undefined in mathematics, so this expression is invalid.	4/0	Handled error	Handled error	Pass
Invalid Characters	The & character is not a valid arithmetic operator, so this expression is invalid in the context of arithmetic operations.	7 & 3	Handled error	Handled error	Pass
Mismatched Parentheses	The parentheses are not properly matched, with one closing parenthesis missing, making the expression invalid.	(((3 + 4) - 2) + (1)	Handled error	Handled error	Pass
Invalid Operator Usage	This expression attempts to divide by zero, which is mathematically undefined, rendering the expression invalid	((5 + 2) / (3 * 0))	Handled error	Handled error	Pass
Invalid Operator Sequence	The expression contains an operator - without a valid operand on its left,	((2 -) 1 + 3)	Handled error	Handled error	Pass

	making it invalid.				
Missing Operand	There is a missing operand after the - operator, making the expression invalid.	((4 * 2) + (-))	Handled error	Handled error	Pass
Invalid Characters	The @ character is not a valid arithmetic operator in this context, causing the expression to be invalid	((7 * 3) @ 2)	Handled error	Handled error	Pass
Integer Overflow	the program is constrained, numbers that exceed int 32 boundaries result in overflow	999*999*67978	handled error	handled error	Pass

6. Environmental needs

Our program only requires the user to install the ‘g++’ compiler on their system to compile the file before they run it.