<Arithmetic Expression Evaluator in C++>

Test Case

Version <1.0>

Revision History

Date	Version	Description	Author
<10/12/24>	<1.0>	<Test Case Document>	<Samantha Adorno, Mason West, Evan Rogerson, Ben Haney, Nick Heyer, Rahul Nesan, Zain Cheema>

Table of Contents

Test Case

1. Purpose

This Test Case Specification document for the Arithmetic Parser project defines a test case to validate the correct parsing and evaluation of arithmetic expressions. It ensures compliance with specified requirements for functionality, accuracy, and error handling.

2. Test case identifier

The test case identifier for this project will be:
TC01…N

3. Test item

Arithmetic Parser

- **Feature Tested**: Evaluation of arithmetic expressions including operator precedence, parentheses, and mixed operations.
- **References**:
  a) **Requirements Specification**: Software-Requirements-Spec
  b) **Design Specification**: Software-Architecture-Spec
  c) **User Guide**: User-Manual

4. Input specifications

Input	Details	Value
Arithmetic Input	A valid expression	3 + 5 * (2 – 4) / 2
Expression Format	Infix Notation	Standard Mathematical Notation
Constants	Constants used during testing	None
Input Relationships	Parentheses for grouping and operator precedence.	Parentheses overrule precedence

5. Output specifications

Output Type	Details	Expected Value
Evaluated Result	The result of the arithmetic expression	A valid number i.e. 1
Error Handling	A resulting error due to wrong input	An error message indicating the problem

Test case ID	Test case description	Test data	Expected results	Actual results	Pass/fail status
TC01	Verify unary negation and addition in parentheses	-(+1) + (+2)	1	1	Pass
TC02	Verify negation and addition with negated parentheses	-(-(-3)) + (-4) + (+5)	-2	-2	Pass

TC03	Verify unary negation and exponentiation	2 ** (-3)	0.125	0.125	Pass
TC04	Verify combining unary operators with parentheses	-(+2) * (+3) - (-4) / (-5)	-6.8	-6.8	Pass
TC05	Verify combining unary operators with arithmetic operations	6.8	6.8	6.8	Pass
TC06	Verify extraneous parentheses with division	((9 + 6)) / ((3 * 1) / (((2 + 2))) - 1)	-60	-60	Pass
TC07	Verify combination of extraneous and necessary parentheses	(((((5 - 3))) * (((2 + 1))) + ((2 * 3))))	12	12	Pass
TC08	Verify nested parentheses with exponents	(((2 ** (1 + 1)) + ((3 - 1) ** 2)) / ((4 / 2) % 3))	4	4	Pass
TC09	Verify mixed operators with extraneous parentheses	((5 * 2) - ((3 / 1) + ((4 % 3))))	6	6	Pass
TC10	Verify complex addition with extraneous parentheses	(((2 + 3))) + (((1 + 2)))	8	8	Pass
TC11	Verify mixed operators	4 * (3 + 2) % 7 - 1	5	5	Pass
TC12	Verify exponentiation	2 ** 3	8	8	Pass
TC13	Verify multiplication and division	10 * 2 / 5	4	4	Pass
TC14	Verify subtraction with parentheses	8 - (5 - 2)	5	5	Pass
TC15	Verify addition	3 + 4	7	7	Pass
TC16	Verify the system's behavior when attempting division by zero	10/0	Error	Error	Pass
TC17	Verify nested parentheses with all operations	((3 + (2 * 4)) - ((1 ** 3) / (5 % 3)))	10.5	10.5	Pass
TC18	Verify unary negation	-5	-5	-5	Pass
TC19	Verify modulo with operations	10 + 15 % 4	13	13	Pass
TC20	Verify unary with parenthesis	-(2 + 3)	-5	-5	Pass
TC21	Verify unary operators with basic arithmetic operators	-(-(-9)) + (-7) + (+4)	-12	-12	Pass
TC22	Verify unmatched parenthesis	4 * (2 + 6 - 3	Error	Error	Pass

TC23	Verify operator sequence behavior	(7 *)+(44)	Error	Error	Pass
TC24	Verify exponents with base 1	1^9	Error	Error	Pass
TC25	Verify behavior of operator without operands	(1+9) + 4/7/	Error	Error	Pass
TC26	Verify pemdas	(2+3) * 4	20	20	Pass
TC27	Verify edge cases	1000000000 * 1000	1000000000000	1000000000000	Pass
TC28	Verify performance stress test	2 + 3 * 5 - (4 * (6 + 2)) / 10 + 100	102	102	Pass
TC29	Verify white space handling	2 +3	5	5	Pass
TC30	Verify mismatched parenthesis	((2 + 3) * 4	Error	Error	Pass
TC31	Verify Decimal Addition and Subtraction	5.5 + 4.5 - 7.2	2.8	2.8	Pass
TC32	Verify Decimal Multiplication	5.5*4.5	24.75	24.75	Pass
TC33	Verify Decimal Division	5.5/4.5	1.22222	1.22222	Pass
TC34	Verify Decimal works with parentheses	(2.3 * 6) + (5.9 - 55)	-35.3	-35.3	Pass
TC35	Verify Decimal stress test	(12345.678*98765.4321−54321.12345)/0.0001+6789.98765	1.22E+13	1.22E+13	Pass
TC36	verify % 0 return error	5%0	error	error	Pass
TC37	verify invalid character	5^3	error	error	Pass
TC38	verify 0 exponent	0**0	1	1	Pass
TC39	verify negative subtraction	5-(-5)	10	10	Pass
TC40	Verify invalid input	five plus two	Error	Error	Pass
TC41	Verify exponent 0 equals 1	4 ** 0	1	1	Pass
TC42	Verify floating point precision	1 / 3	0.333333	0.333333	Pass
TC43	Verify mixed operators without parentheses	1 + 2 * 3	7	7	Pass
TC44	Verify chained exponentiation	2 ** 3 ** 2	512	512	Pass
TC45	Verify maximum operator count	1 + 2 - 3 + 4 - 5 + 6 - 7 + 8 - 9 + 10	7	7	Pass
TC46	Verify fractional modulo	5.5 % 2.2	1.1	1.1	Pass

TC47	Verify leading and trailing whitespace	3 + 2 * 2	7	7	Pass
TC48	Verify empty input		error	error	Pass
TC49	Verify mixed integer and decimal operations	1.2 + 5	6.2	6.2	Pass
TC50	Verify edge cases with negative 0	-0	0	0	Pass
TC51	Verify elimination of lots of white space	1 + 11	12	12	Pass
TC52	Verify elimination of leading white space	1+1	2	2	Pass

6. Environmental needs

6.1.1 Hardware

Any modern computer capable of running a C++ application.

6.1.2 Software

- Operating System: Windows/Linux/MacOS.
- Compiler: GCC

7. Special procedural requirements

- Ensure the parser library or executable is properly built before testing.
- Run the test case in an isolated environment to avoid interference from other test cases.
- Enable debug logging for detailed traceability if results are incorrect.

8. Intercase dependencies

- Ensures basic parser initialization and validation of empty inputs.
- Verifies error handling for invalid arithmetic expressions.