Team 14

Arithmetic Expression Evaluator in C++

Test Case

Version 1.0

Revision History

Date	Version	Description	Author
8/12/24	1.0	Initial Version	Team 14

Table of Contents

Test Case

1. Purpose

Within this test case specification document, we will outline the identifiers, purposes, inputs, and expected outputs of all test cases that will be used in development of the Arithmetic Expression Evaluator in C++. The purpose of this document is to ensure the parser correctly handles various arithmetic expressions, including basic operations like addition, subtraction, multiplication, and division, as well as more complex expressions involving parentheses, operator precedence, and possible error handling scenarios such as division by zero or invalid input. This is done through several test cases determined by and evaluated through a black-box based software testing methodology. This document helps maintain consistent testing procedures, identify potential bugs, and verify that the evaluator meets its design requirements and performs efficiently across a range of inputs.

2. Test cases

Test Case Identifier	Name	Purpose	Inputs	Expected Outputs	Observed Outputs	Pass/ Fail
TC-01	**Addition**	Valid Expression: This expression adds two numeric constants, resulting in a valid calculation.	Terminal Input: 3 + 4	Terminal Output: 7	Terminal Output: 7	Pass
TC-02	**Subtraction with Parentheses**	Valid Expression: The parentheses ensure that the subtraction inside them is performed first, leading to the correct result.	Terminal Input: 8 - (5 - 2)	Terminal Output: 5	Terminal Output: 5	Pass
TC-03	**Multiplication and Division**	Valid Expression: The multiplication and division operators are applied from left to right, resulting in the final answer.	Terminal Input: 10 * 2 / 5	Terminal Output: 4	Terminal Output: 4	Pass
TC-04	**Exponentiation**	Valid Expression: The ** operator calculates 2 raised to the power of 3.	Terminal Input: 2 ** 3	Terminal Output: 8	Terminal Output: 8	Pass

TC-05	**Mixed Operators**	Valid Expression: This expression combines multiple operators and parentheses to correctly calculate the result step by step.	Terminal Input: 4 * (3 + 2) % 7 - 1	Terminal Output: 5	Terminal Output: 5	Pass
TC-06	**Complex Addition with Extraneous Parentheses**	Valid Expression: While there are multiple sets of extraneous parentheses, they do not affect the validity of the expression. The addition is performed correctly.	Terminal Input: (((2 + 3))) + (((1 + 2)))	Terminal Output: 8	Terminal Output: 8	Pass
TC-07	**Mixed Operators with Extraneous Parentheses**	Valid Expression: This expression combines various operators with multiple sets of extraneous parentheses, but they do not change the order of operations or the final result.	Terminal Input: ((5 * 2) - ((3 / 1) + ((4 % 3))))	Terminal Output: 6	Terminal Output: 6	Pass
TC-08	**Nested Parentheses with Exponents**	Valid Expression: This expression includes nested parentheses and exponentiation, creating complexity, but it adheres to the correct order of operations.	Terminal Input: (((2 ** (1 + 1)) + ((3 - 1) ** 2)) / ((4 / 2) % 3))	Terminal Output: 4	Terminal Output: 4	Pass
TC-09	**Combination of Extraneous and**	Valid Expression: This expression includes both extraneous	Terminal Input: (((((5 - 3))) * (((2 + 1))) + ((2 * 3))))	Terminal Output: 12	Terminal Output: 12	Pass

	Necessary Parentheses	parentheses and necessary parentheses to clarify the order of operations. It evaluates correctly.				
TC-10	**Extraneous Parentheses with Division**	Valid Expression: Extraneous parentheses are added for clarity, but they do not affect the validity of the expression. The division, multiplication, and subtraction are performed correctly.	Terminal Input: ((9 + 6)) / ((3 * 1) / (((2 + 2))) - 1)	Terminal Output: -60	Terminal Output: -60	Pass
TC-11	**Combining Unary Operators with Arithmetic Operations**	Valid Expression: This expression combines unary + and - operators with multiplication, division, and addition.	Terminal Input: +(-2) * (-3) - ((-4) / (+5))	Terminal Output: 6.8	Terminal Output: 6.8	Pass
TC-12	**Unary Negation and Addition in Parentheses**	Valid Expression: Unary negation and addition operators are used within parentheses, followed by addition.	Terminal Input: -(+1) + (+2)	Terminal Output: 1	Terminal Output: 1	Pass
TC-13	**Negation and Addition with Negated Parentheses**	Valid Expression: This expression demonstrates nested unary negations and additions, with some values negated and others added.	Terminal Input: -(-(-3)) + (-4) + (+5)	Terminal Output: -2	Terminal Output: -2	Pass

TC-14	**Unary Negation and Exponentiation**	Valid Expression: The unary + and - operators are used with exponentiation to calculate a fractional result.	Terminal Input: +2 ** (-3)	Terminal Output: 0.125	Terminal Output: 0.125	Pass
TC-15	**Combining Unary Operators with Parentheses**	Valid Expression: This expression combines unary operators with parentheses and arithmetic operations.	Terminal Input: -(+2) * (+3) - (-4) / (-5)	Terminal Output: -6.8	Terminal Output: -6.8	Pass
TC-16	**Unmatched Parentheses**	Invalid Expression: This expression has unmatched opening and closing parentheses, making it invalid.	Terminal Input: 2 * (4 + 3 - 1	Terminal Output: Error (Parentheses are mismatched or missing.)	Terminal Output: Error (Parentheses are mismatched or missing.)	Pass
TC-17	**Operators Without Operands**	Invalid Expression: The * operator lacks operands on the left, making the expression invalid.	Terminal Input: * 5 + 2	Terminal Output: Error (Operator with no operand.)	Terminal Output: Error (Operator with no operand.)	Pass
TC-18	**Incorrect Operator Usage**	Invalid Expression: Division by zero is undefined in mathematics, so this expression is invalid.	Terminal Input: 4 / 0	Terminal Output: Error (Division by zero)	Terminal Output: Error (Division by zero)	Pass
TC-19	**Missing Operator**	Invalid Expression: The expression lacks an operator between 5 and (2 + 3), making it invalid.	Terminal Input: 5 (2 + 3)	Terminal Output: Error (Operand missing operator.)	Terminal Output: Error (Operand missing operator.)	Pass
TC-20	**Invalid Characters**	Invalid Expression: The	Terminal Input: 7 & 3	Terminal Output: Error	Terminal Output:	Pass

		& character is not a valid arithmetic operator, so this expression is invalid in the context of arithmetic operations.		(Unknown characters)	Error (Unknown characters)	
TC-21	**Mismatched Parentheses**	Invalid Expression: The parentheses are not properly matched, with one closing parenthesis missing, making the expression invalid.	Terminal Input: (((3 + 4) - 2) + (1)	Terminal Output: Error (Parentheses are mismatched or missing.)	Terminal Output: Error (Parentheses are mismatched or missing.)	Pass
TC-22	**Invalid Operator Usage**	Invalid Expression: This expression attempts to divide by zero, which is mathematically undefined, rendering the expression invalid.	Terminal Input: ((5 + 2) / (3 * 0))	Terminal Output: Error (Division by zero)	Terminal Output: Error (Division by zero)	Pass
TC-23	**Invalid Operator Sequence**	Invalid Expression: The expression contains an operator - without a valid operand on its left, making it invalid.	Terminal Input: ((2 -) 1 + 3)	Terminal Output: Error (Operator with no operand.)	Terminal Output: Error (Operator with no operand.)	Pass
TC-24	**Missing Operand**	Invalid Expression: There is a missing operand after the - operator, making the expression invalid.	Terminal Input: ((4 * 2) + (-))	Terminal Output: Error (Operator with no operand.)	Terminal Output: Error (Operator with no operand.)	Pass

TC-25	**Invalid Characters**	Invalid Expression: The ^ character is not a valid arithmetic operator in this context, causing the expression to be invalid.	Terminal Input: ((7 * 3) ^ 2)	Terminal Output: Error (Unknown characters)	Terminal Output: Error (Unknown characters)	Pass
TC-26	**Multiple Consecutive Operators**	Invalid Expression:Multiple consecutive operators without valid operands between them make the expression invalid	Terminal Input: 5 + - * 3	Terminal Output: Error: Operator with no operand.	Terminal Output: Error: Operator with no operand.	Pass
T-27	**Special Cases of Division by Zero**	Invalid Expression: Division by zero is undefined	Terminal Input: 10 / (2 -2)	Terminal Output: Error: Division by zero.	Terminal Output: Error: Division by zero.	Pass
T-28	**Precedence of many operators**	Valid Expression: This expression test precedence and associativity of mixed operators	Terminal Input: 3 + 5 * 2 - 8 / 4	Terminal Output: 11	Terminal Output: 11	Pass