Fancy Calculator Inc.

Arithmetic Expression Evaluator in C++

Test Case

Version <1.0>

Revision History

Date	Version	Description	Author
<12/10/2024>	<1.0>	<Tests cases, input specifications, and output specifications have been added.>	<Bryant Goseland>

Table of Contents

Test Case

1. Purpose

The Test Cases document is a crucial step in the process of completing the Arithmetic Expression Evaluator project. This document will be used to identify, test, and fix potential faults within the program. Each test case will be designed to run specific sub-sections of the code. The outcome will be carefully documented with the following details: ID, Description, Input Data, Expected Results, Actual Results, and Pass/ Fail status. All details will be properly organized into a table within this document.

2. Test case identifier

The Test Case identifier will be used in the data table to document each distinct test case. The following information will describe how an identifier is created.

TC – This will be the beginning of every identifier

-TOK or –PAR or –MAI – Occurs after TC, will identify the file that contains the code the test case is testing

-## - Will be the number of the test case for the file. Will increase by one for each test case

Examples: TC-TOK-01, TC-PAR-01, TC-MAI-01, TC-TOK-02

3. Test item

TC-MAI-01	Tests the end program condition. Should be an easy process for the user.
TC-MAI-02	Tests the program's response to an empty input.
TC-MAI-03	Tests maximum input length for the program to process.
TC-MAI-04	Tests the precision limit of the output number. The program should have a precision of seven at the current setting.
TC-TOK-01	Tests decimal numbers by adding a decimal next to an integer. This will ensure the user is able to create more accurate equations to what their needs are.
TC-TOK-02	Tests invalid characters within an expression. This ensures that the tokenizer rejects non-arithmetic inputs.
TC-TOK-03	Tests handling of negative numbers in an arithmetic expression.
TC-TOK-04	Tests handling of multiple decimal points in a number.
TC-TOK-05	Tests the usage of spaces and/or tabs in an arithmetic expression.
TC-TOK-06	Tests mismatched parenthesis in the expression. Ensures that the parser can detect errors.
TC-TOK-07	Tests operators without operands
TC-TOK-08	Tests missing operand with unary operators
TC-PAR-01	Tests division by zero in an arithmetic expression.
TC-PAR-02	Tests multiple nested parentheses.
TC-PAR-03	Tests empty parenthesis evaluation.
TC-PAR-04	Tests modulo by zero in an arithmetic expression.

TC-PAR-05	Tests addition of two numeric constants
TC-PAR-06	Tests subtraction with parenthesis
TC-PAR-07	Tests PEMDAS order with multiplication and division
TC-PAR-08	Tests exponentiation of numeric constants
TC-PAR-09	Tests mixed operators, should evaluate according to PEMDAS
TC-PAR-10	Tests unnecessary amounts of parenthesis
TC-PAR-11	Tests mixed operators with an unnecessary amount of parenthesis
TC-PAR-12	Tests nested parenthesis with exponents
TC-PAR-13	Tests a combination extraneous and necessary parenthesis
TC-PAR-14	Tests multiplication of two numeric constants using parenthesis
TC-PAR-15	Tests parenthesis with unary operators
TC-PAR-16	Tests negation of parenthesis
TC-PAR-17	Tests multiple negations of a single numeric constant
TC-PAR-18	Tests unary negation with exponentiation

4. Input specifications

TC-MAI-01	Input “end” or “END” into the input line
TC-MAI-02	Input nothing. After the program prompts an input, hit “enter” or “return”
TC-MAI-03	Input a very long expression. E.g. “1 + 1 + 1 + 1 + 1” ... repeated to a thousand ones being added together
TC-MAI-04	Input “1.11111111111111111111”
TC-TOK-01	Input “14.001 + 0.1” into the input line
TC-TOK-02	Input “3 + 5 @ 4”
TC-TOK-03	Input “–9 + 10”
TC-TOK-04	Input “3.14.15 + 9”
TC-TOK-05	Input “4 + 6”
TC-TOK-06	Input “(9 + 11”
TC-TOK-07	Input “*5+2”
TC-TOK-08	Input “(4 * 2) + (-))”
TC-PAR-01	Input any number over zero. E.g. “9 / 0”
TC-PAR-02	Input “((((1 + 2) * 3) - 4) / 5)”
TC-PAR-03	Input “10 + ()”
TC-PAR-04	Input “1+10**23+1%0”
TC-PAR-05	Input “3+4”

TC-PAR-06	Input “8-(5-2)”
TC-PAR-07	Input “10*2/5”
TC-PAR-08	Input “2**3”
TC-PAR-09	Input “4 * (3+2) % 7 –1”
TC-PAR-10	Input “(((2+3)))+(((1+2)))”
TC-PAR-11	Input “(5 * 2) - ((3 / 1) + ((4 % 3)))”
TC-PAR-12	Input “(((2 ** (1 + 1)) + ((3 - 1) ** 2)) / ((4 / 2) % 3))”
TC-PAR-13	Input “(((((5 - 3))) * (((2 + 1))) + ((2 *3))))”
TC-PAR-14	Input “(12.4)(2)”
TC-PAR-15	Input “+(-2) * (-3) – ((-4) / (+5))”
TC-PAR-16	Input “-(+1) + (+2)”
TC-PAR-17	Input “---------1”
TC-PAR-18	Input “+2 ** (-3)”

5. Output specifications

TC-MAI-01	Should fully end the program and allow the user to use terminal commands.
TC-MAI-02	Returns “Failed to evaluate tokens, please try again”
TC-MAI-03	The program has handled over 500,000 individual inputs from testing and accurately displayed the result. The limit relies on the user’s IDE and system hardware.
TC-MAI-04	Returns “1.111111” Which does reflect the current precision set to seven. This setting can be changed manually by altering the PRECISION variable at the top of the AEEmain.cpp file.
TC-TOK-01	Returns “14.101”. Signifies that decimal values are evaluated properly.
TC-TOK-02	Returns “Unknown Character Error. Failed to tokenize the expression due to errors.”
TC-TOK-03	Returns “1” as expected. The program properly handles negatives.
TC-TOK-04	Returns “Invalid Input Error. Failed to tokenize the expression due to errors.”
TC-TOK-05	Returns “10” as expected. The program can handle any number of spaces.
TC-TOK-06	Returns “Unmatched Parenthesis Error. Failed to tokenize the expression due to errors.”
TC-TOK-07	Returns “Invalid Input Error. Failed to tokenize the expression due to errors” showing that the program properly detects invalid operand sequences.
TC-TOK-08	Returns “Failed to evaluate tokens, please try again” showing that the program properly detects missing operands.

TC-PAR-01	Returns “Divide by zero error. Failed to evaluate tokens, please try again”, indicating that the program handles division by zero.
TC-PAR-02	Returns “1” as expected. The program can properly handle numerous parentheses as long as they are closed.
TC-PAR-03	Returns “Failed to evaluate tokens, please try again”. The program properly evaluates the expression if the parentheses are empty.
TC-PAR-04	Returns “Mod by zero error. Failed to evaluate tokens, please try again” indicating the program handles modulo by zero.
TC-PAR-05	Returns “7” showing that the program can properly handle addition
TC-PAR-06	Returns “5” showing that the program can properly handle subtraction with parenthesis
TC-PAR-07	Returns “4” showing that the program reads from left to right when the priority of the operators is the same.
TC-PAR-08	Returns “8” showing that exponentiation is properly calculated
TC-PAR-09	Returns “5” showing that mixed operators are correctly evaluated according to PEMDAS order.
TC-PAR-10	Returns “8” showing that extra parenthesis works properly
TC-PAR-11	Returns “6” showing that extra parenthesis works properly with various operators
TC-PAR-12	Returns “4” showing that exponents work with nested parenthesis
TC-PAR-13	Returns “12” showing that different combos of parenthesis evaluation still work
TC-PAR-14	Returns “24.8” showing that multiplication using side by side parenthesis works
TC-PAR-15	Returns “6.8” showing that unary operators affect the inside of parenthesis
TC-PAR-16	Returns “1” showing that negation is properly applied to the inside of parenthesis
TC-PAR-17	Returns “-1” showing that negation is calculated correctly
TC-PAR-18	Returns “0.125” showing that negation properly applies to the exponent

6. Environmental needs

6.1.1 Other

If any other requirements are needed for the test case, it will be added to the table below

TC-MAI-02	Depending on the user’s keyboard, the equivalent of “enter” (Windows default) or “return” (macOS default) will need to be pressed.
TC-MAI-03	The limit will vary by user based on their IDE and system.