University of Kansas

Arithmetic Expression Evaluator

Test Case

Version 1.2

Revision History

Date	Version	Description	Author
12/3/24	1.0	Created table for parts 2-5	Eliza
12/5/24	1.1	Filled out some test cases for table	Eliza
12/10/24	1.2	Added test cases from Canvas	Lillian

Table of Contents

Test Case

1. Purpose

This test case document for the AEE (Arithmetic Expression Evaluator) defines test cases for items that should be tested to ensure quality.
The sections of the test cases will be ordered in a sequence denoted by its test case ID. The test case ID is understood in two parts, the first few characters, which indicate which part of the program is being tested, and a number for each test case in the specified part of the program. Each test case ID contains a description for how it is tested, input specifications, and output specifications.

Test Case ID	Test Item	Input Specifications	Output Specifications
mak0001	Testing makefile compilation. Log onto KU Linux terminals. Navigate to the directory that holds all the files using terminal commands like cd and ls. Once in the directory, ensure all .cpp and .h files are there, along with the makefile titled makefile or Makefile. run command make and make sure each .cpp and .h pair has a corresponding .o file.	Using terminal commands, navigate to the directory with all files and start make file with command “make”.	Corresponding .o files for each .cpp and .h pair and an .exe file for the total application.
mod001	Test file individually using a supporting main function to pass input. Pass double digit number for left operand and single digit for right operand, no decimals or negatives. Result should be checked using an external calculator and should be single or double digit with no decimals or negatives.	10%4	2
mod002	Test file individually using a supporting main function to pass input. Left operand is double digit decimal and the right operand is single digit negative with decimal places. Since the right operand is negative and regular modulus does not calculate for negatives return should be an error of invalid expression.	10.00%-4.00	error of invalid expression
mod003	Test file individually using a supporting main function to pass input. Right operand should be single digit smaller than the left operand. Left operand should be a single digit with decimal places and larger than the right operand. Return should be single digit with no decimal places. Result should be checked using an external calculator.	2%2.75	error of invalid expression
mudi001	Test file individually using a supporting main function to pass input. Results should be checked using an external calculator.	10/2	5
mudi002	Test file individually using a supporting main function to pass input.	10/0	error: division by zero is not allowed

expo001	Test file individually using a supporting main function to pass input.	4**2	16
adsu001	Test file individually using a supporting main function to pass input.	5+3+2	10
valid001	Tests capability of addition and processing of input	3 + 4	7
valid002	Tests capability of subtraction, handling of parentheses, and processing of input	8 - (5 - 2)	5
valid003	Tests capability of multiplication, division, and processing of input	10 * 2 / 5	4
valid004	Tests exponentiation	2**3	8
valid005	Tests mixing of operators	4 * (3+2) % 7 - 1	5
valid006	Tests handling of extraneous parentheses	(((2+3)))+(((1+2)))	8
valid007	Tests handling of extraneous parentheses + mixed operators	((5*2)-((3/1)+((4%3))))	6
valid008	Tests nested parentheses with exponents	(((2**(1+1))+((3-1)**2))/((4/2)%3))	4
valid009	Tests extraneous and necessary parentheses	(((((5-3)))*(((2+1)))+((2*3))))	12
valid010	Tests extraneous parentheses with division	((9+6))/((3*1)/(((2+2)))-1)	-60
valid011	Tests the combination of unary operators with arithmetic operations	+(-2)*(-3)-((-4)/(+5))	6.8
valid012	Tests unary negation and addition	-(+1)+(+2)	1
valid013	Tests negation and addition with negated parentheses	-(-(-3))+(-4)+(+5)	-2
valid014	Tests unary negation with exponentiation	+2**(-3)	0.125
valid015	Tests the combination of unary operators with parentheses	-(+2)*(+3)-(-4)/(-5)	-6.8
invalid001	Tests unmatched opening/closing parentheses	2*(4+3-1	Error
invalid002	Tests operators without operands	*5 + 2	Error
invalid003	Tests division by zero	4/0	error: division by zero is not allowed
invalid004	Tests missing operators	5(2+3)	Error
invalid005	Tests invalid characters	7&3	Error

invalid006	Tests mismatched parentheses	(((3+4)-2)+(1)	Error
invalid007	Tests invalid operator usage	((5+2)/(3*0))	error: division by zero is not allowed
invalid008	Tests invalid operator sequence	((2-)1+3)	Error
invalid009	Tests missing operand	((4*2)+(-))	Error
invalid010	Tests invalid characters in a larger sequence	((7*3)^2)	Error
quit001	Tests ability of program to quit when “q” inputted	q	Program terminates
quit002	Tests ability of program to quit when “Q” inputted	Q	Program terminates

Test Case ID	Expected Result	Actual Result	Pass/Fail Status
mak0001	Corresponding .o files for each .cpp and .h pair and an .exe file for the total application.	make: ‘implementation’ is up to date.	P
mod001	2	2	P
mod002	error of invalid expression	error: input error or invalid input	P
mod003	error of invalid expression	error: input error or invalid input	P
mudi001	5	5	P
mudi002	error: division by zero is not allowed	error: Division by zero is not allowed.	P
expo001	16	16	P
adsu001	10	10	P
valid001	7	7	P
valid002	5	5	P
valid003	4	4	P
valid004	8	8	P
valid005	5	5	P
valid006	8	8	P
valid007	6	6	P
valid008	4	4	P

valid009	12	12	P
valid010	-60	-60	P
valid011	6.8	6.8	P
valid012	1	1	P
valid013	-2	-2	P
valid014	0.125	0.125	P
valid015	-6.8	-6.8	P
invalid001	Error	Error	P
invalid002	Error	Error	P
invalid003	Error	error: division by zero is not allowed	P
invalid004	Error	Error	P
invalid005	Error	Error	P
invalid006	Error	Error	P
invalid007	error: division by zero is not allowed	error: division by zero is not allowed	P
invalid008	Error	Error	P
invalid009	Error	Error	P
invalid010	Error	Error	P
quit001	Program terminates	you quit, program terminates	P
quit002	Program terminates	you quit, program terminates	P

2. Environmental Needs

2.1.1 Hardware

The Arithmetic Expression Evaluator requires very basic computer hardware or any common computer hardware that is widely available in 2024. Anything from a laptop to a raspberry pi can run the AEE.

2.1.2 Software

The Arithmetic Expression Evaluator (AEE) has very basic needs to be executable. To be handleable, it needs a reasonably working computer with a directory and file system that supports and can hold .cpp, .h, .exe, and other such files. The AEE also needs a command line interface to do very basic work, preferably the KU Linux servers command line interface, as was specified in the KU EECS 348 Fall 2024 class. Other software requirements to consider are an operating system, which can help the user navigate the file system and command line interface.

2.1.3 Other

Numbers that exceed the capacity of a C++ double to handle are not permitted since this is an abuse of hardware capacity. If numbers do exceed the capacity of a C++ double, the Arithmetic Expression Evaluator (AEE) will result in an overflow and output error messages explaining “undefined behavior”. To work with some of the Evaluators (such as Modulus), the group included the C++ library Cmath.

3. Special procedural requirements

Test cases must be executed on KU Linux servers since that is where this project was specified to be executed. The project can work in other setups but is not as optimal, so valid test cases must be done on these servers. The KU Linux servers are only accessible by KU EECS students and staff, as they require a KU EECS username and password to log on. If a tester is unable to log on to the KU EECS Linux servers, it is possible to compile and test the AEE through another command line, though it may result in undefined behavior.