**EvalEx**

Test Case

Version 1.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 29/11/23 | 1.0 | Added all detail for the test cases. | Tommy Lam, Will Whitehead, Joshua Lee, Max Biundo, Joe Hotze, Gunther Leuchtefeld |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Purpose 4

2. Test case identifier 4

3. Test item 4

4. Input specifications 4

5. Output specifications 4

6. Environmental needs 4

6.1.1 Hardware 4

6.1.2 Software 4

6.1.3 Other 4

7. Special procedural requirements 5

8. Intercase dependencies 5

**Test Case**

# Purpose

*This Test Case Specification document for the EvalEx defines a test case for an item that should be tested.*

***NOTE: for sections 2, 3, 4, and 5: It is OK to use a table like the one proposed in class, also suggested on the project part 5 description.***

| Test Case ID | Test Case Description | Test Data | Expected Results | Actual Results | Pass/Fail |
| --- | --- | --- | --- | --- | --- |
| Add-01 | Simple addition for two numeric constants | 3 + 4 | 7 | 7 | Pass |
| Add-02 | Complex addition with extraneous parentheses | (((2 + 3))) + (((1 + 2))) | 8 | 8 | Pass |
| Sub-01 | Subtraction with parentheses | 8 - (5 - 2) | 5 | 5 | Pass |
| Sub-02 | Complex subtraction with extraneous parentheses | (((2 - 3))) - (((1 - 2))) | 0 | 0 | Pass |
| Mul-01 | Simple multiplication operator | 10 \* 2 | 20 | 20 | Pass |
| Mul-02 | Multiplication and division operators applied from left to right | 10 \* 2 / 5 | 4 | 4 | Pass |
| Mul-03 | Complex Multiplication | 8 \* 3\*8 | 192 | 192 | Pass |
| Div-01 | Simple division operator | 15 / 3 | 5 | 5 | Pass |
| Div-02 | Complex division with extraneous parentheses | (((20 / 5))) / (((10 / 10))) | 4 | 4 | Pass |
| Div-03 | complex division | (5 / 8) / 9^3 | 0.000857339 | 0.000857339 | Pass |
| Exp-01 | Simple exponentiation operator | 2 ^ 3 | 8 | 8 | Pass |
| Exp-02 | Nested exponentiation | (2 ^ (3 ^ 3)) | 134,217,728 | 134,217,728 | Pass |
| Exp-03 | Nested exponentiation with sign | (-(2^(2^4))) | -65,536 | -65536 | Pass |
| Exp-04 | Exponentiation with a negative base and exponent | (-2) ^ (-3) | -0.125 | -0.125 | Pass |
| Mix-01 | Mixed operators with combines multiple operators and parentheses | 4 \* (3 + 2) % 7 - 1 | 5 | 5 | Pass |
| Mix-02 | Mixed operators with exponents and modulo | (((2 ^ (1 + 1)) + ((3 - 1) ^ 2)) / ((4 / 2) % 3)) | 4 | 4 | Pass |
| Mix-03 | Mixed operators with extraneous parentheses | ((5 \* 2) - ((3 / 1) + ((4 % 3)))) | 6 | 6 | Pass |
| Mix-04 | Combination of extraneous and necessary parentheses | (((((5 - 3))) \* (((2 + 1))) + ((2 \*  3)))) | 12 | 12 | Pass |
| Mix-05 | Combination of extraneous parentheses | ((9 + 6)) / ((3 \* 1) / (((2 + 2))) - 1) | -60 | -60 | Pass |
| Mix-06 | Mixed operators with subtraction and division | 15 - 3 / 3 | 14 | 14 | Pass |
| Una-01 | Unary negation and addition operators | -(+1) + (+2) | 1 | 1 | Pass |
| Una-02 | Nested unary negation and addition | -(-(-3)) + (-4) + (+5) | -2 | -2 | Pass |
| Una-03 | Unary operators with exponentiation | +2 ^ (-3) | 0.125 | 0.125 | Pass |
| Una-04 | Unary operators with parentheses and arithmetic operations | -(+2) \* (+3) - (-4) / (-5) | -6.8 | -6.8 | Pass |
| Una-05 | Unary operators with more complex precedence | +(-2) \* (-3) - ((-4) / (+5)) | 6.8 | 6.8 | Pass |
| Una-06 | Unary operators with subtraction and parentheses | -(5 - 2) | -3 | -3 | Pass |
| Una-07 | Nested unary operators with exponentiation | -(-(+2) ^ 3) | 8 | 8 | Pass |
| Una-08 | Unary operators with mixed operations | -(+4) / (2 \* 2) | -1 | -1 | Pass |
| Err-01 | Divide by zero simple | 4 / 0 | Divide by zero | Divide by zero | Pass |
| Err-02 | Invalid character in expression for & | 7 & 3 | Invalid character in expression | Invalid character in expression | Pass |
| Err-03 | Divide by zero within nested parentheses | ((5 + 2) / (3 \* 0)) | Divide by zero | Divide by zero | Pass |
| Err-04 | Invalid character in expression for @ | ((7 \* 3) @ 2) | Invalid character in expression | Invalid character in expression | Pass |

# Test case identifier

# Add: Tests addition function

Sub: Tests subtraction function

Mul: Tests multiplication function

Div: Tests division function

Exp: Tests exponentiation function

Mix: Tests multiple different mathematical functions in one test case  
 Una: Tests unary operators

# Output specifications

The program has a few different outputs. The first and most preferable output is when the program passes without any errors. In this case, the output is just a number. The second case that can happen is that the expression can end in an error. This error message can vary based on what the error is. For example, if a user tries to divide by zero, the program will output ‘Divide by Zero.’ If the program sees an invalid character expression, it will send that error message.

# Environmental needs

### Hardware (nothing particular for the arithmetic expression project)

N/A

### Software (nothing particular for the arithmetic expression project)

We used Windows / Mac operating systems, and the gcc compiler to compile our program.

### Other

N/A