Group 16, LLC

Command Cruncher

Test Case

Version 1.1

Revision History

Date	Version	Description	Authors
01/Dec/24	1.0	Initial team contact with document	Sneha Thomas, Hannah Prosch, Daniel Van Dalsem, Emma Roy, Nifemi Lawal
08/Dec/24	1.1	Finishing up the document.	Sneha Thomas, Hannah Prosch

Table of Contents

Test Case

1. Purpose

The purpose of this test case specification is to validate the functionality and robustness of the Command Cruncher, which is an arithmetic expression evaluator. It aims to ensure that the evaluator adheres to the specified requirements like correct parsing, operator precedence, parenthesis handling, and error detection.

2. Test case identifier

- Table with test cases in section 5**

3. Test item

1. Expression parsing:
    a. Tokenizing the input correctly
    b. Creating the correct tree data structure
2. Operator Precedence:
    a. Proper handling of operators
3. Parentheses handling:
    a. Accurate evaluation of nested and extra parentheses
4. Error handling:
    a. Correct handling of division by zero, unmatched parentheses, and invalid character errors

Relevant references:

- Requirements Specification: EECS 348 Term Project Document
- Design Specification Document

4. Input specifications

5. Output specifications

ID	NAME	PURPOSE	INPUTS	EXPECTED OUTPUT	OBSERVED OUTPUT	PASS / FAIL
TC001	Addition	Add values	3+4	7	**7**	**PASS**
TC002	Parentheses precedence - Subtraction	Test priority with parentheses	8 - (5 – 2)	**5**	**5**	**PASS**
TC003	Multiplication and division	Check multiplication and division	10 * 2/5	4	**4**	**PASS**
TC004	Exponentiation	Validate	2**3	8	**8**	**PASS**

		exponentiation				
TC005	Mixed Operators	Ensure mixed operators compute right	4*(3+2)%7-1	5	**5**	**PASS**
TC006	Complex Addition with Extraneous Parentheses	Handle redundant parentheses	(((2 + 3))) + (((1 + 2)))	8	**8**	**PASS**
TC007	Mixed Operators with Extraneous Parentheses	Test redundant parentheses with mixed ops.	((5 * 2) - ((3 / 1) + ((4 % 3))))	6	**6**	**PASS**
TC008	Unary Negation and Exponentiation	Validate unary negation/exponents	+2**(-3)	0.125	**0.125**	**PASS**
TC009	Unmatched Parentheses	Detect missing parentheses	2*(4+3-1	Thrown Exception	**Error: Syntax Error at token position 7**	**PASS**
TC010	Incorrect Operator Usage	Catch divide-by-zero errors	4/0	DivideByZero Exception/ Error	**Error: Division by zero**	**PASS**
TC011	Missing Operand	Identify syntax errors	((4*2)+(-))	Thrown Exception	**Error: Syntax Error at token position 9**	**PASS**
TC012	Negative inputs	Test expressions with negatives	-(2+3)*4	-20	**-20**	**PASS**
TC013	Repeating decimals	Check if certain fractions result in repeating decimals	1/3	0.333333	**0.333333**	**PASS**
TC014	Unary Chaining	Check double unary operators	--5	5	**5**	**PASS**
TC015	Invalid Characters	Detect invalid	7&3	Thrown Exception	**Error: Illegal**	**PASS**

		characters			**character: &**	
TC016	Exit String	Ensure program ends on "stop"	stop	Program Ends	**No output**	**PASS**
TC017	Parenthesis multiplication	Handles implied multiplication with parenthesis	(1+6)(1*2)	14	**14**	**PASS**

6. Environmental needs

6.1.1 Hardware

- Processor: Intel i5 or equivalent
- Memory: 4GB RAM minimum

6.1.2 Software

- Operating System: Windows 10/Linux/macOS
- Compiler: GCC or Clang supporting C++11 or higher
- Tools: Unit testing frameworks

***6.1.3** Other*

- Reliable power supply for testing environments

7. Special procedural requirements

- Ensure the program is compiled without warnings or errors
- Use command-line arguments to pass test inputs where applicable

8. Inter-case dependencies

- Standard operations should be tested first, before edge cases and error-inducing operations