Jayhawks

Arithmetic Expression Evaluator
Software Requirements Specifications

Version 1.0

Revision History

Date	Version	Description	Author
10/4/2023	0.1	The initial Software Requirements Specs	Alexandra, Deborah, Riley, Timo, Victor, Ellia
10/14/2023	1.0	The first published and completed Specifications version	Alexandra, Deborah, Riley, Timo, Victor, Ellia

Table of Contents

Software Requirements Specifications

1. Introduction

This document contains our program requirements. This will serve as a reference for our implementation. The definitions, acronyms, abbreviations, and references that will be used will be found in the introduction. The rest of the document will be constituted of the following subsections: Overall Description, Specific Requirements, Classification of Functional Requirements, and Appendices. This document is subject to change.

1.1 Purpose

The purpose of the Software Requirements Spec is to specify all external program behaviors. It also describes nonfunctional requirements, design constraints, and other factors necessary to provide a complete description of our software's requirements.

The following people use the Software Requirements Spec:

- The **project manager** uses it to plan the project schedule and resource needs, and to track progress against the schedule.
- **Project team members** use it to understand what they need to do, when they need to do it, and what other activities they are dependent upon.

1.2 Scope

This *Software Requirements Spec* applies to the Arithmetic Expression Evaluator. The Arithmetic Expression Evaluator is a simple expression calculator that processes user entered expressions. This document influences this project and this project only.

1.3 Definitions, Acronyms, and Abbreviations

Not Applicable

1.4 References

- EECS348: Term Project in C++ (00-Project-Description.pdf), Fall 2023, Published by EECS 348
    - Can be obtained on Canvas.
    - This reference will describe the project objectives and tasks, and different valid and invalid expressions.
- Arithmetic Expression Evaluator Software Development Plan (Software_Development_Plan_9_13_2023.pdf), 9.13.2023
    - Document can be found here:

      https://github.com/t878a585/348_Project/blob/main/Artifacts/Software_Development_Plan_9_13_2023.docx
    - This reference will describe the vision of our program.

1.5 Overview

This *Software Requirements Spec* contains the following information:

Overall Description — Describes elements that affect the product's requirements.

Specific Requirements — provides a detailed description of all the software requirements.

Classification of Functional Requirements — Indicates the priority of each requirement.

Appendices — Additional information/documents that relate to this project

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

Not Applicable

2.1.2 User Interfaces

The user interface will be a command line based interface. A message will be output at the beginning indicating that the user can exit by pressing 'ctrl-c'. The user will be able to input an expression and upon hitting enter, the expression will be parsed, and the value of the expression will be returned. The user will be able to continue entering expressions as long as they would like. If the user types in an invalid expression or one that results in an invalid math operation (divide by zero) an error message will be displayed. The user will be able to continue inputting expressions after the error message is shown.

2.1.3 Hardware Interfaces

Not Applicable

2.1.4 Software Interfaces

Not Applicable

2.1.5 Communication Interfaces

Not Applicable

2.1.6 Memory Constraints

Not Applicable

2.1.7 Operations

Not Applicable

2.2 Product functions

The product functions are listed below:

- Arithmetic Operations: Addition, Subtraction, Multiplication, Division, Exponentiation, and Modulo
- Use of parentheses for grouping
- Integer value input (floats are not supported)
- User input handling – Accepts numeric input from user.
- User interface – Terminal based input and output.
- Error handling – Displaying clear error messages when invalid data input or expressions are detected.

2.3 User characteristics

While typical users will likely be students or employees, we expect that a wide variety of people will use this calculator.

2.4 Constraints

The constraints are listed below:

- The program must be written in either C or C++
- The program must be compatible with Linux.
- It must be able to accept another expression after having processed one already.

2.5 Assumptions and dependencies

The assumptions and dependencies are listed below:

- Assume that the user knows their way around Linux.
- Assume that the user knows their way around a Linux terminal.
- It must be able to accept another expression after having processed one already.

2.6 Requirements subsets

Not Applicable

3. Specific Requirements

3.1 Functionality

3.1.1 Addition

Mathematical expressions with addition will be allowed to be entered into the calculator. The addition operation will be designated by the "+" symbol.

3.1.2 Subtraction

Mathematical expressions with subtraction will be allowed to be entered into the calculator. The subtraction operation will be designated by the "-" symbol.

3.1.3 Multiplication

Mathematical expressions with multiplication will be allowed to be entered into the calculator. The multiplication operation will be designated by the "*" symbol.

3.1.4 Division

Mathematical expressions with division will be allowed to be entered into the calculator. The division operation will be designated by the "/" symbol.

3.1.5 Exponents

Mathematical expressions with exponent will be allowed to be entered into the calculator. The exponent operation will be designated by the "^" symbol.

3.1.6 Modulo

Mathematical expressions with modulo will be allowed to be entered into the calculator. The modulo operation will be designated by the "%" symbol.

3.1.7 Parenthesis

Mathematical expressions with parentheses will be allowed to be entered into the calculator. The parentheses will force the expression inside to be resolved before other surrounding operations.

3.1.8 Error Handling

There are two cases of invalid expressions: Mathematically invalid expressions and grammatically invalid expressions.
Examples of mathematically invalid expressions:

- Incorrect operator usage
- Ex.) Dividing by zero

Examples of grammatically invalid expressions:

- Operators without Operands
- Missing Operator
- Invalid Character
- Mismatched Parentheses
- Invalid Operator Sequence

Mathematically invalid expressions will be discovered while calculating the value of expressions.
Grammatically invalid expressions will be discovered because the grammar won't be processable and will indicate a generic grammar error.
Errors will be shown in the terminal after hitting enter.

3.1.9 Equation Entry

General mathematical expressions will be allowed to be entered in as expressed by the following BNF grammar:

<expression> ::= <expression> + <expression>
| <expression> - <expression>
| <expression> / <expression>
| <expression> * <expression>
| <expression> ^ <expression>
| <expression> % <expression>
| (<expression>)
| <integer>

<integer> ::= <digit> <integer>
| <digit>

<digit> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

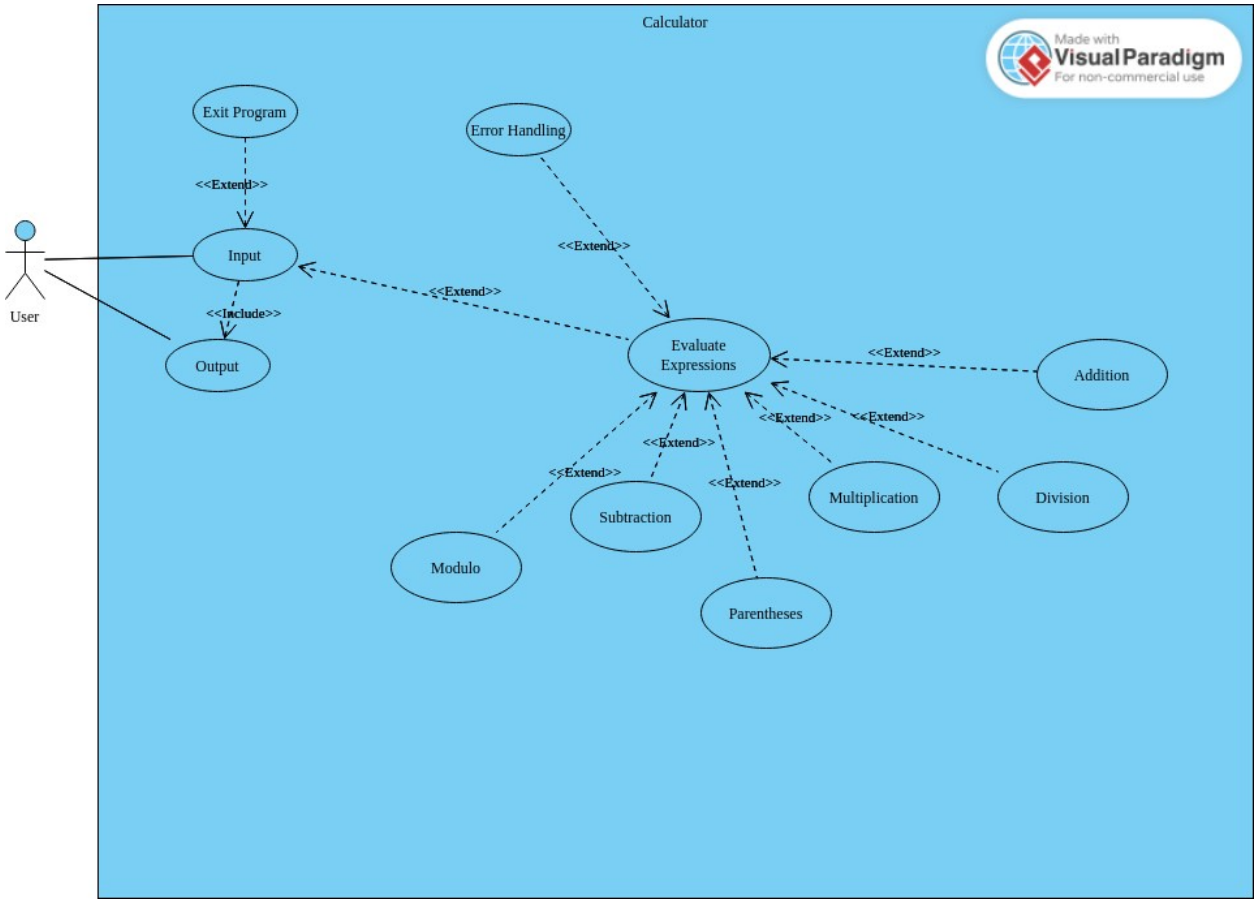
This only expresses the syntax as this is ambiguous grammar that can violate operator precedence.

The expressions will be computed taking operator precedence into consideration. The following will be the order for operations for calculating expressions (1 is the highest precedence):

1) ()
2) ^
3) *,/
4) %
5) -,+

Expressions will be allowed to have space within them, but they will be ignored. A user will be able to type in an expression in the described format and once they are done typing it out, they will be able to hit enter, and the value of the expression will be output.

3.2 Use-Case Specifications



3.3 Supplementary Requirements

- Expressions up to 1024 characters should not cause any errors.
- Must meet the speed expectations of a typical user.
- Code needs to be adequately commented.

4. Classification of Functional Requirements

Functionality	Type
Addition Expression	Essential
Subtraction Expression	Essential
Multiplication Expression	Essential
Division Expression	Essential

Modulo Expression	Essential
Parenthesis Handling	Essential
Error Handling	Essential
Ability for user to continue entering expressions	Desirable
Asking user for an equation	Essential
Menu for user	Optional

5. Appendices

We only referred to two external documents that can be found in section 1.4.