<Company Name>

<Calculator>
Software Requirements Specifications

Version <1.0>

Revision History

Date	Version	Description	Author
<26/09/23>	<1.0.0>	First additions	Xavier Ruyle
<01/10/23>	<1.0.0>	Partially completed sections 2 and 3	Henry Hoopes

Table of Contents

Software Requirements Specifications

1. Introduction

1.1 Purpose

The purpose of this SRS is to explain the behavior of the calculator. It describes what the calculator needs as well as the intended users that it will be directed toward.

1.2 Scope

The SRS applies to the Calculator, which is made to parse, and calculate mathematical expressions. It provides the user with the ability to calculate basic arithmetic with operators using a user interface in the command line. The calculator is designed for users who want a simple and minimalistic calculator through a command line interface.

1.3 Definitions, Acronyms, and Abbreviations

Calculator: The program itself

UI: User interface

IFN: Infix Notation

PN: Postfix Notation

EXPR: Expression

1.4 References

N/A – no references at this moment

1.5 Overview

The rest of the SRS will contain the overall description which will detail the general factors which affect the calculator software and its requirements. It will then go into detail about the specific requirements that are needed to create the calculator. It will then classify the functional requirements in a table format and categorize them by type. Lastly, the appendices are available for further reading.

2. Overall Description

2.1 Product perspective

2.1.1 System Interfaces

This project won't interface with another system, but will use C++ libraries to utilize stack and queue data structures

2.1.2 User Interfaces

The user will interact with our software through the command line. Our program will receive typed input from the user and will print output to the command line.

2.1.3 Hardware Interfaces

There won't be any additional hardware interfaces. The software will receive input and send output using the standard input/output library in C++.

2.1.4 Software Interfaces

For this project, our software will interface with C++ libraries to include I/O functionality and stack and queue data structures.

2.1.5 Communication Interfaces

This project will perform all the computations on one machine and won't communicate with other systems. There will be no communication interfaces.

2.1.6 Memory Constraints

There won't be any necessary memory constraints on the software. It will be running on a general-purpose computer, so there will be much more memory available than our program will use.

2.1.7 Operations

The software will perform the following operations: reading in an mathematical expression from the user, parsing that expression into mathematical operations, controlling the flow of mathematical operations, and printing the result of the expression to the user.

2.2 Product functions

The product will take in a typed arithmetic expression from the user, evaluate the result of the expression, and print the result to the user. The software must handle the arithmetic operations in the correct order. It must be able to evaluate expressions inside parentheses and detect when there are unmatched parentheses. In addition, it must be able to handle invalid operations, such as division by 0 or invalid operators. In order

to handle arithmetic expressions, it must be able to recognize the numeric constants as integers.

2.3 User characteristics

This software should be usable for the largest possible group of users. So, the only requirement on the users is that they understand how to type input into the program. Even if they don't understand how to format input, our software should respond to illegal input and instruct the user on the correct form of input.

2.4 Constraints

The project must be developed in the language C++. Also, the product should have all code well documented for ease of legibility. In addition, we are required to use a data structure, such as a stack or a tree, to represent the arithmetic expression's structure.

2.5 Assumptions and dependencies

This project is dependent on the C++ standard library continuing to provide the classes we require (stdio, stack, queue) and not removing them in a potential update to C++, which is highly unlikely.

2.6 Requirements subsets

There are future requirements that will be mandatory at one point but are optional at this stage in the development process. They are listed below as Expanded Operator Support and Floating Point Constants requirements.

3. Specific Requirements

3.1 Functionality

3.1.1 <Command Line Input>

The product should receive an arithmetic expression from the user through input in the command line.

3.1.2 <Expression Parsing>

The software should parse the expression input into a data structure that stores the structure of the expression, so the program can calculate the expressions by order of operations.

3.1.3 <Numeric Constant>

The software should recognize numeric constants in the user input and assume they are integer form.

3.1.4 <Operator Support>

*The software should be able to handle the following operators + (addition), - (subtraction), * (multiplication), / (division), % (modulo), and ^ (exponential).*

3.1.5 <Parentheses Support >

The software should be able to evaluate expressions enclosed inside parentheses. In addition, the software should be able to detect unmatched parentheses and notify the user of the invalid input.

3.1.6 <Operator Precedence>

The software should evaluate the expression in order of operator precedence. So, expressions inside parentheses should be evaluated first, then exponential expressions, then multiplication, division, or modulo expressions, and finally addition and subtraction expressions.

3.1.7 <Error Handling>

The software will handle invalid input from the user, such as incorrect operators, or operators without

numbers. When these expressions are input, the program will notify the user of the error in their input.

3.1.8 <Expanded Operator Support>

*The software will expand the supported operators to ** (exponential).*

3.1.9 <Floating Point Constants>

The software will handle floating point values in addition to integer values for numeric constants.

3.2 Use-Case Specifications

There aren't any use-case specifications for this project at this stage in the development process.

3.3 Supplementary Requirements

3.3.1 <Development Language>

The software should be written in the programming language C++.

3.3.2 <Comments>

The software should contain clear comments to describe the functionality of the program source code.

4. Classification of Functional Requirements

Functionality	Type
Command Line Input	Essential
Expression Parsing	Essential
Numeric Constant	Essential
Operator Support	Essential
Parenthesis Support	Essential
Operator Precedence	Essential
Error Handling	Essential
Expanded Operator Support	Desirable
Floating Point Constant	Desirable

5. Appendices