HL Group

Boolean Logic Simulator in C++ Software Requirements Specifications

Version <1.0>

Revision History

Date	Version	Description	Author
02/28/24	1.0	Initial Template Filled Out	Changwen Gong

Table of Contents

Software Requirements Specifications

1. Introduction

The purpose of this SRS (Software Requirements Specification) document is to explain and describe the requirements and specifications for the Boolean Logic Simulator in C++. This project's specific description, timelines, deadlines can be found in the SDP, or the Software Development Plan.

The project's functionality and technical systems can be found in this document. The SRS explains the project's context, constraints, assumptions, as well as any requirements that are a part of the Boolean Logic Simulator in C++.

1.1 Purpose

The purpose of the SRS is to describe how the Boolean Logic Simulator in C++ can evaluate any user inputted Boolean expressions with the following operators:

- AND (&)
- OR (|)
- NOT (!)
- NAND or NOT AND (@)
- XOR or NOT OR ($)

The project should also be able to handle expression parsing, giving proper priority and precedence to parentheses with proper error handling. The user should also be able to define and input truth values (T or F, representing True or False respectively) for each variable input, and the program should be able to evaluate the input and output the resulting truth value.

1.2 Scope

This SRS document describes all systems and any subsystems of the Boolean Logic Simulator in C++. It explains the expected functionality and any non-functional requirements demanded by the project. Diagrams and Use-Case modeling will be related to the specific functionality of the project in its entirety.

1.3 Definitions, Acronyms, and Abbreviations

Project/Program: Boolean Logic Simulator in C++ as described by the SDS. See *Section 1.4 References for SDS*

SRS/This document: Software Requirements Specifications. See *Section 1.4 References for SRS*

SDP: Software Development Plan. See *Section 1.4 References for SDP*

1.4 References

SRS: Accessible at https://github.com/KNEternity/348HL/tree/main/docs

SDP: Accessible at https://github.com/KNEternity/348HL/tree/main/docs

1.5 Overview

The following sections provides an overall description of the project, specific requirements that are needed, classification of the functional requirements, and finally an appendix for further referencing.

Overall Description

The project will be a command-line program that will accept Boolean expressions as inputs. It will then parse the expression, handle any errors that may occur, calculate the corresponding values (True or False) related to that expression, and finally output the result. This program will be implemented with C++ and may consist of a variety of coding techniques including but not excluded to parsing, data structures, and/or algorithmic designs.

1.6 Product perspective

1.6.1 System Interfaces

The Boolean logic simulator will not have any system interfaces.

1.6.2 User Interfaces

The Boolean logic simulator will include a simply command-line user input interface. The user will be able to type via keyboard a Boolean expression as an input and receive an output via the screen.

1.6.3 Hardware Interfaces

The Boolean logic simulator will not have any hardware interfaces.

1.6.4 Software Interfaces

The Boolean logic simulator will not have any software interfaces; however, the integrated development environment (IDE) that the project will be run on may have software implications.

1.6.5 Communication Interfaces

The Boolean logic simulator will not have communication interfaces.

1.6.6 Memory Constraints

The Boolean logic simulator will have very minimal memory constraints as it only needs to store the logic and expression input and output a singular result.

1.6.7 Operations

The Boolean logic simulator will have the following functional operations:

- Take in user input
- Parse Boolean expressions
- Evaluate Boolean expressions
- Handle any errors
- Output result

1.7 Product functions

The Boolean Logic Simulator in C++ should be able to recognize, parse, and evaluate the following Boolean operators:

- AND (&)
- OR (|)
- NOT (!)
- NAND or NOT AND (@)
- XOR or NOT OR ($)

The project must be able to parse parentheses as well for precedence. It should then calculate the final truth value of the entire expression inputted and present it as either True or False. Robust error handling should also be implemented to handle any invalid inputs (such as but not limited to invalid variables, missing parentheses, unknown operators, or any other potential input issues).

1.8 User characteristics

This project is intended for students, professors, programmers, and any other individuals who have a need to evaluate Boolean expressions.

1.9 Constraints

The project must be implemented using the C++ programming language. It should also be able to run on popular operating systems, including but not limited to: Linux and Windows.

1.10 Assumptions and dependencies

The assumptions and dependencies of this program are listed below:

- The user can input any Boolean expressions (valid or invalid)
- The user has access to the program and the input command line
- There is a proper environment for the program to be ran in

1.11 Requirements subsets

The requirements for this program can be separated into the following categories: Functional Requirements and Supplementary Requirements. Functional Requirements outline what is functionally required for the program to properly parse, evaluate, and output the result. This section will also include requirements that directly affect the behavior of the program. Supplementary Requirements include any non-functional requirements and development constraints that are required be the program.

2. Specific Requirements

2.1 Functionality

This section and the following subsections describe the functional requirements associated with the project.

2.1.1 Accept User Inputs

The Boolean logic simulator must allow the user to input an expression and any corresponding truth values via the command line.

2.1.2 Parse Boolean Expressions

The Boolean logic simulator must parse the expression and raise any errors within the user input. It must recognize any Boolean logic operators (&, |, !, @, $) as well as parentheses and precedence handling.

2.1.3 Evaluate Boolean Expressions

The Boolean logic simulator must evaluate the input expression in its entirety as well as any truth values related to the variables inputted by the user and return a truth value for the whole expression.

2.1.4 Output Truth Value Results

The Boolean logic simulator must output the result for the user to see.

2.2 Use-Case Specifications

The following use-case model diagram depicts the functional requirements of the project:

2.3 Supplementary Requirements

This section describes the supplemental requirements as needed by the Boolean logic simulator. All documentation and Readme can be found here. See *Section 1.4 Reference.*

3. Classification of Functional Requirements

Functionality	Type
Accept User Input	Essential
Parse Boolean Expressions	Essential
Evaluate Boolean Expressions	Essential
Output Truth Value Results	Essential
Error Handling	Essential
Print Specific Errors	Desirable
Modularity and Scalability	Optional
User Input Specific Error Identification	Optional

4. Appendices

Appendices will either be specified as a part of this document or not. As of the current revision, there are no appendices.