Cryptic Coders

Boolean Logic Simulator
Software Requirements Specifications

Version <1.0>

[Note: The following template is provided for use with the Unified Process for EDUcation. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

Date	Version	Description	Author
20 MAR 24	1.0	Filled out the document	Mark, Brett, Muskan, Aiden, Ty

Table of Contents

Software Requirements Specifications

1. Introduction

This Software Requirement Specification document outlines the requirement for the Boolean Logic Simulator in C++. It should be capable of performing basic Boolean calculations such as AND, OR, NOT, NAND and XOR. The program should be able to handle complex logic circuits with multiple gates and input/output signals.

1.1 Purpose

The purpose of this Software Requirements Specification it to establish the constraints and requirements that need to be considered when planning and developing our software.

1.2 Scope

The uses of the Boolean Logic Simulator will consist of: solving sets of expression such as (F @ T) | (T @ F) and produce the correct output. Students and Teachers can utilize this software to do Boolean calculations.

1.3 Definitions, Acronyms, and Abbreviations

SRS- Software Requirements Specifications

1.4 References

00-2024-EECS348-project-description.pdf

1.5 Overview

The remainder of the SRS elaborates and provides more specific information about each individual constraint/ requirement. The document is organized in a way that describes all the requirements in section 2 and goes over the specific requirements in section 3. Section 4 is used sort through and classify each requirement, which makes implementation easier.

Overall Description

1.6 Product perspective

1.6.1 *System Interfaces* – What the user will be interacting with

1.6.2 *User Interfaces*

1.6.3 *Hardware Interfaces* – What machine the user will use to run the software.

1.6.4 *Software Interfaces*

1.6.5 *Communication Interfaces* – How the users doubts/questions will be handled.

1.6.6 *Memory Constraints*

1.6.7 *Operations* – What kinds of operation the software can solve.

1.7 Product functions

1.8 User characteristics

1.9 Constraints

1.10 Assumptions and dependencies

1.11 Requirements subsets – Any additional requirement that will improve the software.

2. Specific Requirements

2.1.1 Implement logical operations for AND, OR, NOT, NAND and XOR

3.1.2 Develop a mechanism to parse user-provided Boolean expressions in infix notation, respecting operator precedence and parenthesis.
3.1.3 Allow users to define truth values (True/False) for each variable represented by T and F.
3.1.4 Calculate the final truth value of the entire expression and present it clearly (True or False).
3.1.5 Implement robust error handling for invalid expressions, missing parentheses, unknown operators, or other potential issues, and provide informative error messages.
3.1.6 Ensure that your program can handle expressions enclosed within parentheses (including seemingly excessive but correctly included pairs of parentheses) to determine the order of evaluation.

2.2 Use-Case Specifications

Calculation of Boolean expressions by students or teachers

What kind of logic we are using

2.3 Supplementary Requirements

Time constraints: Due by End of semester

User friendly interface

Produced clear and legible output.

Handles errors correctly

3. Classification of Functional Requirements

Functionality	Type
Operator Support	Essential
Expression Parsing	Essential
Truth Value Input	Essential
Evaluation and Output:	Essential
Error Handling	Essential
Parenthesis Handling	Essential
Time constraints	Essential
User friendly interface	Desirable
Produced clear and legible output	Desirable

4. Appendices

N/A