Kyler_Jack_Cyn_Malek_Kaleb_Jackson Co

The Logic Calculator

User’s Manual

Version 1

Revision History

Date	Version	Description	Author
29/4/24	1	Completing the Document	Kyler, Cyn, Malek, Jack, Jackson, Kaleb

Table of Contents

Test Case

1. Purpose

The user manual is a detailed description of how to use the software name **"The Logic Calculator"** launched by **Kyler_Jack_Cyn_Malek_Kaleb_Jackson Co.** from the perspective of the user. It will include sections detailing the software as a whole, how to get started using the program, all of the features provided, and how to use and interpret the output.

2. Introduction

The purpose of our Boolean Logic Calculator is to return a True or False value of a user input expression. It was built with the user in mind and is meant to be easy to understand and interact with. It has features to handle errors of different types and let the user know what specifically has caused the error.

The software takes inputs from the user with specific user input requirements. The user can use this software to evaluate Logic Boolean expressions. If the user's input has logical or syntactical errors, the program will output the result of the expression, otherwise it would inform the user of the possible mistakes in their input. To run our program, users have to access the provided GitHub link below and download all of the files in the "src" folder and install C++ compiler as well as choosing their text editor of choice. We recommend using the VScode editor.

The software source code is stored inside of this GitHub repository: https://github.com/ErikT2003/CynTranEECS348.git. Once accessed this GitHub repository, users go to the *"src"* folder and download all of the files within this folder into their desired destination. Then they can open this program with their text editors, IDE of their choice. Once the program runs, it will prompt the user to enter a Boolean Logic Expression. The program will evaluate this expression and output the result of the expression in either T or F

3. Getting started

The program will prompt "Enter Boolean Expression:" in the terminal field. The user input their expression here. The valid input from user includes: T (True), F (False), & (AND), | (OR), ! (NOT), @ (NAND), $ (NOR), # (XOR), * (XNOR).

The user will input their expression inside the terminal field of their chosen editor where the program will prompt, requesting the user to input their expression. The user can input multiple variables and expressions, however, the program will interpret this as one expression and will only produce one output. Users can enter as many unique expressions as they want but they have to wait for the program to finish evaluating their previous entered expression and prompt the user to input another expression.

The program will check for any error in the user's input and evaluate the user expression, else the program will inform the user of their mistake and prompt another request for user input. The output for the expression is either True or False. This does not mean that the input from the user is correct or incorrect, but this indicates the boolean result conducted from the user's input.

4. Advanced features.

The program can identify white space within the user's expression input, therefore the user won't need to worry about entering extra space in the expression. The program can still evaluate the expression as normal.
The program can identify if user expression is invalid, and informing user what kind of error in user input such as if it is missing parentheses, mix matched parentheses, invalid operator, missing operator, missing operand, invalid operand. and the program can distinguish the differences between these errors and inform the user which error they make

The program has the capability to break down the input expression into smaller components, these components will be passed into stack when parsing the expression, this will allow more accurate reading from the user and avoid ambiguity. This feature also improves the program sensitivity to identify the error and differentiate the difference between these errors to inform the user. This assists users in understanding and correcting their input expression.

The program also has the capability to perform multiple operand such as: & (AND), | (OR), ! (NOT), @ (NAND), $ (NOR), # (XOR), * (XNOR). The program distinguishes the difference between these symbols, and produces correct output based on the user's input. The program can identify if the expression is wrong or not, if the expression wrongly uses the operand and inform the user of the error.

5. Troubleshooting

A few common errors that may be encountered are unbalanced parentheses, invalid symbols, or invalid operations. If an unbalanced parentheses error is returned, then the expression that was entered most likely has not a balanced amount of both left parentheses “(“ and right parentheses “)”. By checking the expression that was entered and verifying that (1) Every opened parentheses “(“ has a corresponding closing parentheses “)” and (2) There are no extra closing parentheses “)”, this error should be able to be fixed.

If an invalid symbols error is encountered, most likely a symbol was entered in the expression that does not correlate to any operand or truth value. Check that the expression that you input ONLY contained these following symbols and letters: &, |, !, @, $, T, F. Anything other than these will not be recognized by our program and it will not know what to do with it, therefore an error will be returned.

An invalid operation error most likely means that a binary operand only received a single value. For example, the AND (&) operation requires two truth values to work. If you input T & , then it does not have a second value to run and therefore it is not a valid expression. Make sure that every time a binary operation is input, it has two values so it can properly evaluate the truth value.

6. Examples

<table>
<tr><td>Expression: (T | F) $ F</td><td>Evaluation: True</td></tr>
<tr><td>Expression: ! (T & T)</td><td>Evaluation: False</td></tr>
<tr><td>Expression: (F @ T) | (T @ F)</td><td>Evaluation: False</td></tr>
<tr><td>Expression: (T $ T) & F</td><td>Evaluation: False</td></tr>
<tr><td>Expression: ! F | ! T</td><td>Evaluation: True</td></tr>
<tr><td>Expression: (((((T | F) & F) | (T & (T | F))) @ (T @ T)) $ (! (T | F)))</td><td>Evaluation: True</td></tr>
<tr><td>Expression: ((F $ ((T | F) & (F @ (T | F)))) | (T $ (T & F)))</td><td>Evaluation: False</td></tr>
<tr><td>Expression: (((! (T $ F)) & (T @ T)) | ((F | T) & (T $ T)))</td><td>Evaluation: False</td></tr>
<tr><td>Expression: (((T @ T) $ (F @ T)) | ((!T) & (T | (!T))))</td><td>Evaluation: True</td></tr>
<tr><td>Expression: ((F @ T) $ (T | (F & F))) & (T & (T @ (!T)))</td><td>Evaluation: False</td></tr>
</table>

7. Glossary of terms

Valid Expression: expression that meet with the specific requirement of the program
Operator: & (AND), | (OR), ! (NOT), @ (NAND), % (NOR), $ (XOR), * (XNOR) are operators that contribute to how the operation is computed.
GitHub: Platform used to store source code for the program.

8. FAQ

Why is the program written in C++?

- Because C++ is a powerful and fast programming language allowing fast computation which is especially useful in this type of program.

How can you know the program accurately evaluates the expression?

-We created multiple test cases, and we test case proof the program. We make sure that the program produces the answer as we have previously evaluated ourselves.

-We created multiple scenarios where the program can make the mistakes and find the reason why the program evaluated incorrectly. We then identify and fix the bug making sure that our program produces the correct answer.

Why doesn’t the program run?

-We used the VScode text editor for this program which required some extensions for c++ to be installed. Checking all of the extensions on your VScode, if you use any other text editor, make sure you have all of the component installed before running the program.

What is the contact information?

-If the program doesn’t work reachout to this email for further assistance: cyntran28@ku.edu