Huang and Co.

Boolean Expression Evaluator

User's Manual

Version 1.0

Revision History

Date	Version	Description	Author
27/04/24	1.0	Completed tasks and reviewed	Andrew Huang, Elizabeth Miller, Hunter Long, Katherine Swann, Keaton Xu

Table of Contents

User's Manual

1. Purpose

This user manual provides a step-by-step guide on how to use the Boolean Expression Evaluator (B.E.E.) to solve Boolean logic expressions, and lists the features it contains as well as the steps to take in case of errors.

2. Introduction

The B.E.E. is made to calculate Boolean logic expressions that are inputted. The gates implemented in B.E.E. are the following: AND, OR, NOT, NAND, XOR. The truth values of expressions can be defined by the user, and the B.E.E. has the ability to handle expressions of any complexity through its parenthesis handling and error handling.

How to install and run B.E.E.:

1. To install the B.E.E., navigate to the Github repository: https://github.com/andrewhuang2019/EECS348Project .
2. Download the repository to your local machine
3. Navigate to the src folder in the project directory using the terminal.
4. Use your terminal to compile main.cpp using the command 'g++'.
5. See the output using the command './a.out'.

3. Getting started

When the B.E.E. runs, its first feature provides a prompt to define custom True or False variables (see 'Advanced features'.)

After the first feature, the B.E.E. will ask the user for a Boolean expression. If the previous feature was selected, this expression should be entered with the user-defined variables for true and false; if not, it should be entered with "T" for true and "F" for false (see "Examples".)

Finally, press "enter" on the keyboard, and the B.E.E.. will automate the hard work of condensing this expression down to a simple "True" or "False".

```
Do you want to use your own variables for T and F?(Y/N): N
Enter expression: (((((T | F) & F) | (T & (T | F))) @ (T @ T)) $ (! (T | F)))
Expression: (((((T | F) & F) | (T & (T | F))) @ (T @ T)) $ (! (T | F)))
Evaluation: True
```

4. Advanced features

As an alternative to writing "T" for true and "F" for false, the B.E.E. allows users to instead define custom variables to represent true and false. If this option is desired, the user should input "Y" in response to the first prompt. Following this, the B.E.E. will then prompt the user to provide one custom variable each for true and false. After these are defined, the user should then provide the equation using these variables.

5. Troubleshooting

If an invalid expression is entered, an error message will be displayed and the program will abort. Common examples of invalid expressions include missing matching parentheses, unsupported operators, and incorrect variable instantiation. When an error message is encountered, this is indicative of an invalid expression being entered, so the program should be run again with a valid expression.

6. Examples

The B.E.E. allows for numerous logical operations to be used through the use of different symbols. These are:

- AND (&): Returns true if both terms are true

```
Enter expression: T & F
Expression: T & F
Evaluation: False
```

- OR (|): Returns true if at least one of the terms are true

```
Enter expression: T | F
Expression: T | F
Evaluation: True
```

- NAND (@): Returns true unless both terms are true

```
Enter expression: T @ F
Expression: T @ F
Evaluation: True
```

- XOR ($): Returns true if exactly one of the terms are true

```
Enter expression: T $ F
Expression: T $ F
Evaluation: True
```

Additionally, any section of the expression can be chosen to be calculated first by grouping it within parentheses. These can be nested to give priority as the expression is gradually condensed down, in the fashion of the typical order of operations.

```
Enter expression: (T | F) & F
Expression: (T | F) & F
Evaluation: False
```

```
Enter expression: T | (F & F)
Expression: T | (F & F)
Evaluation: True
```

Finally, NOT can be used by putting an exclamation mark (!) before a variable or a parentheses-grouped section of the expression.

```
Enter expression: !F & T
Expression: !F & T
Evaluation: True
```

```
Enter expression: !(F | F) & T
Expression: !(F | F) & T
Evaluation: True
```

7. Glossary of terms

- **Boolean:** a data type that holds a value of either True or False.
- **Boolean expression:** an expression consisting of Booleans and operations performed on them, which can be condensed to a single True or False value.

8. FAQ

Q: Can the B.E.E. support multiple expressions at once?
A: No. The B.E.E. can only handle one provided expression at a time.

Q. Does the B.E.E. support lowercase letters as True and False variables?

A: Yes!

Q. What operating systems does the B.E.E. support?

A. The B.E.E. is supported across all operating systems.

Q. Can the B.E.E. evaluate arithmetic expressions?

A. No.

Q. Does the B.E.E. cost money to install or operate?

A. No! The B.E.E. is free and open-source.