

---

**Huang and Co.**

---

**Boolean Expression Evaluator**

**User's Manual**

**Version 1.0**

Boolean Expression Evaluator	Version: 1.0
User's Manual	Date: 27/04/24
01082009	

## 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

Boolean Expression Evaluator	Version: 1.0
User's Manual	Date: 27/04/24
01082009	

## Table of Contents

1. Purpose	4
2. Introduction	4
3. Getting started	4
4. Advanced features	4
5. Troubleshooting	4
6. Example of uses	4
7. Glossary	5
8. FAQ	5

Boolean Expression Evaluator	Version: 1.0
User's Manual	Date: 27/04/24
01082009	

# 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 truth values of these 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.

To install the B.E.E., navigate to the Github repository: <https://github.com/andrewhuang2019/EECS348Project>. Download the repository to your local machine, then navigate inside the project folder that is now on your machine. Use your terminal to run main.cpp.

## 3. Getting started

Usage of the B.E.E. begins with the opportunity to define custom variables. (For more information on this process, see "Advanced features".)

Following this option, the B.E.E. will ask the user for a Boolean expression. If the previous option 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. (For more information on operators and grouping with parentheses, 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 select "yes" 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

Boolean Expression Evaluator	Version: 1.0
User's Manual	Date: 27/04/24
01082009	

```
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.

Boolean Expression Evaluator	Version: 1.0
User's Manual	Date: 27/04/24
01082009	

## 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.