Code Commanders

Code Commanders

User's Manual

Version 1.1

Revision History

Date	Version	Description	Author
05/02/2024	1.0	Initial user manual version	Timothy Holmes
05/02/2024	1.1	Second user manual version	Timothy Holmes

Table of Contents

User's Manual

1. Purpose

This document will explain how the calculator is utilized and will provide descriptions of what functionality is available for each user.

2. Introduction

The software is used to solve Boolean expressions. It can solve equations using AND, OR, NOT, XOR, and NAND. The option to enter a new expression or view history of expressions will be available, also an exit will also be available to exit the program. To be able to run the program you can access the GitHub at https://github.com/TimoHolmes/EECS348-Project and open the code files and download the code to your computer. You can then open the files into any IDE (Integrated Development environment, Visual Studio Code). You then should be able to run the code and enter any Boolean expression you want solved.

Buttons featured:

The buttons needed to operate the calculator correspond to the buttons on the keyboard. These include the letters A-Z, Boolean operators, &, |,!, $, @, opening and closed parentheses ( ), and the numbers 1-24 on the numeric pad.

Symbol	Description	Keyboard button
A-Z	Corresponds to the letter being typed	Press the corresponding letter button
&	Corresponds to AND:	&
|	Corresponds to OR	|
!	Corresponds to NOT	!
@	Corresponds to NAND	@
$	Corresponds to XOR	$
(	Open parenthesis. This is for order of operations and begins closing off a function	(
)	Closing parenthesis. This closes the function if inside the parenthesis and is for order of operations	)
1	Corresponds to 1. This will be used to start writing an expression or use one variable in a function	1
2	Corresponds to 2: This will view the functions history or use two variables for a function	2
3	Corresponds to 3: This will exit the function or use three variables for a function	3

1-24	Corresponds to how many variables the user wants in the function	Type the corresponding number on the number pad (ex: 10 uses 1 and 0)

3. Getting started

To get started using our program, you will be introduced to a menu page. You will have an option to enter an expression or exit the program.

If Users select 1. New Expression, users will be asked to enter how many variables they want. For example, if 6 variables are entered, you will be given 6 variables to use in the expression and how to format AND, OR, NOT, XOR, NAND. The following snippet is what the user will see.

How many variables will be in your expression?: 6
Your variables are: U V W X Y Z
When defining your expression below, please use the following
symbols to represent the following operations:
AND = &
OR = |
NOT = !
XOR = $
NAND = @
Use parentheses to separate 2-bit blocks in your expression
Write your expression using variables U, V, W, X, Y, Z below.

The user will then be asked to write the equation with the given variables.
Expression: (U&V)|X(Y|Z)

The program will then ask the user for the value of the variables with a T, for TRUE, and F, for FALSE.
What is the truth value of U(Please enter T or F): T
What is the truth value of V(Please enter T or F): F
What is the truth value of W(Please enter T or F): T
What is the truth value of X(Please enter T or F): F
What is the truth value of Y(Please enter T or F): F
What is the truth value of Z(Please enter T or F): T

The user will then be displayed with the truth table and the final value of the Boolean expression.

Truth Table:
Variables:
W | X | Y | Z |
T | T | F | F |
Subexpressions:
W&X = T
Y|Z = F
Expression: (W&X)|(Y|Z) = T

Finally, the user will be able to do a new expression or exit the program.

4. Advanced features

An advanced feature that our program offers is if the Boolean expression entered does not follow the correct format (example: (T&F)). If the user enters a lower-case letter (t) rather than an upper-case letter (T) our program will check for lower-case letters and the user will be asked to type the upper-case letters. We will not be accounting for whitespace so users will have to enter the expression with no spaces. The backspace and enter keys may be used like other programs to delete a symbol or calculate a function. We will provide some examples and instructions on how the format of the Boolean expression should look like. Having these capabilities ensures an efficient way to solve expressions without using extra symbols. Another advanced feature we will have is a 'view history'. The user will be able to view the history after the first pass-through. No history will be able to be viewed if the user does not enter a Boolean expression.

5. Troubleshooting

If a user types in an invalid expression when converting from an arbitrary letter to a truth value, they will be prompted to either type in a T or an F for the calculation to be solved. Another issue may be putting in an extra parenthesis where it is not needed, and this will throw an error to the user that the calculator detected an unmatched parenthesis. The calculator will first generate arbitrary variables to create a function and if someone accidentally typed W and X for the function when it asked for Y and Z, then that is invalid because the user needs to input the values they are given. If a user types a function with only one variable in it, then it wouldn't return a result since it is missing an operand.

6. Examples

To use our calculator to solve any Boolean expression, please refer to the *Getting Started* section in this document to see how this calculator is run and the symbols used. Here are a couple examples that may be helpful for solving basic expressions. Note that you will need to have parentheses to close each function off so it can be solved.

AND:
To find the solution using an AND operator, first type in how many variables you want, then type in a function using arbitrary letters with the AND (&) operator.
EXAMPLE: T&F
1. Enter how many variables you want: 2
2. The calculator will give some arbitrary letters (Y, Z). Type in your function using these given letters starting
3. The calculator will ask the user to enter a truth value in place of the letters: either a T or an F based on user choice.
4. Press enter: The output should return False (F) and will display a truth table for what Y and Z are

OR:
To calculate an OR function, first type in the first variable, followed by the | button and then the next variable. Once the expression is done, press the enter button on the keyboard.
EXAMPLE: T|F
1. Enter how many variables you will use in the expression: 2
2. The calculator will give two letters (Y and Z) to input. Enter the function using Y, | and Z respectively.
3. Enter T or an F for the letters.
4. Press enter and it will give the output to T
    OUTPUT: T

NOT
To calculate a function containing a NOT, first enter "!", then the variable that it is being NOT with and then the rest of the function.
EXAMPLE: !T|!F
1. Enter how many variables that will be used: 2
2. Enter the function using the NOT (!) operator followed by the variable given the letters Y and Z. Type in !, Y,|,!,Z respectively
3. Convert Y and Z into either a T or an F
4. Press the enter key for OUTPUT: T

NAND:
To calculate a NAND function, first type the first variable, followed by the @ key and then the second variable.
EXAMPLE: T@F
1. Enter how many variables you want: 2
2. Enter function in terms of Y and Z and press the buttons for Y, @, and Z respectively.
3. Convert Y and Z into a T or an F.
4. Press enter button for an OUTPUT: T

XOR:
To calculate an XOR function, first type a variable followed by the $ symbol and the second variable until all parts of function are entered.
EXAMPLE: T$T
1. Enter how many variables you want: 2
2. Enter function in terms of Y and Z and the buttons for Y, $, Z respectively.
3. Convert Y and Z into either a T or F
4. Press the enter button and the output will result in F.
   OUTPUT: F

Order of operations:
To calculate a multivariable function using order of operations, first type in an open parenthesis and then the function you want inside of the parenthesis and then a closed parenthesis to close off the function. You may put multiple functions in an expression if they are closed properly.
EXAMPLE: (T@F)|(T&F)
1. Enter how many variables you want: 4
2. Type in function using the generated letters; ex: W, X, Y Z. Press buttons for (, W, @, X,), |,(Y,&,Z,) respectively
3. Enter either a T or F for the generated letters.
4. Press enter and the result will output to be T.
   OUTPUT: T

All examples listed above will have a truth table and subexpressions for solving certain parts of the expression and determine what T and F were initialized to.

7. Glossary of terms

Interface: an interface is the layout of information between components in an application. Here, the interface is the calculator's communication between the user and the C++ code the calculator is based on.
Boolean expression: uses AND, NOT, OR, NAND or XOR to evaluate an equation that will return either a True or False value.
IDE (Integrated Development Environment): An Integrated Development Environment is what most C++ programs use to run a program, debugging and testing. Here, it is used to run the calculator program and its code files.
Repository: On GitHub, there are repositories to store program files and code files for a program. Here, the repository is used to download the code files used for the project, collaborate with other developers with revision history and here, it is available for anyone to build using the files provided.

8. FAQ

What are the requirements of this software? What devices can this be used on?

This program will require an IDE to run the program and needs a computer that can run C++ files. Refer to the *Introduction* section on how to set up and get started on this calculator.

What are some limitations to this calculator?

This is a Boolean calculator, hence there can be limitations that may occur such as the output doesn't always match as expected. As stated in the troubleshooting section, this calculator can only handle functions that are not spaced out and will only be able to support valid functions and integers based on our code. Another thing to consider is that this may seem confusing at first, but using letters that aren't a T or an F may cause confusion for the user, and it should be noted that the different letters will eventually turn into a T and F based on user choice before the function is executed.