FURZE PLATT SENIOR SCHOOL

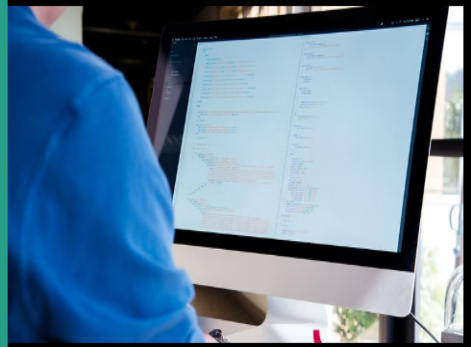
A Level Computing

Furze Platt Senior School

Course Content Paper 1

1. Hardware

1.1a - **The CPU & FDE Cycle**
1.1b - **Performance**
1.2 - **Processors**
1.3a - **Input & Output Devices**
1.3b - **Memory & Storage**

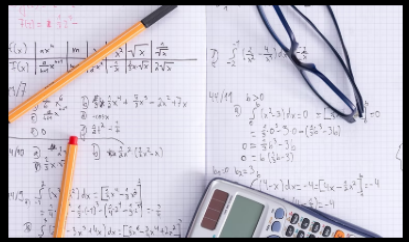


2. Software

2.1 - **Operating Systems**
2.2a - **Applications & Utilities**
2.2b - **Translators & Compilation**
2.3 - **Software Development Methodologies**
2.4a - **Programming & Pseudocode**
2.4b - **Assembly Language**
2.4c - **Object-Oriented Language**

3. Networks & Databases

3.1a - **Compression**
3.1b - **Encryption & Hashing**
3.2a - **Databases & Normalisation**
3.2b - **SQL**
3.3a - **Network Characteristics**
3.3b- **Protocols & TCP-IP Stack**
3.3c - **Network Hardware & DNS**
3.3d - **Network Security & Threats**
3.4a - **Web Technologies**
3.4b - **HTML, CSS & JavaScript**



4. Data & Logic

4.1a - **Data Types & Character Sets**
4.1b - **Denary, Binary & Hexadecimal**
4.1c - **Signed Binary & Floating Point**
4.1d - **Binary Calculations**
4.1e - **Shifts & Masks**
4.2 - **Data Structures**
4.3a - **Logical Operators & Truth Tables**
4.3b - **Flip Flops, Adders, Laws & Maps**

5. Laws

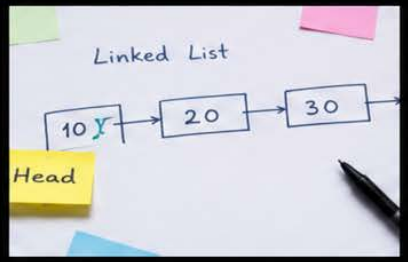
5.1 - **Computer Legislation**
5.2 - **Moral & Ethical Issues**

Course Content – Paper 2

OCR A Level Computer Science – Data Structures, Algorithms & Programming

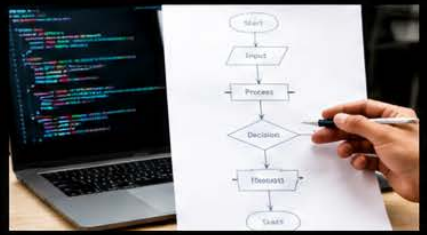
1. Algorithms

1.1a - Computational Thinking
1.1b - Problem Solving
1.2 - Designing Algorithms
1.3 - Algorithms & Programming
1.4 - Algorithm Efficiency



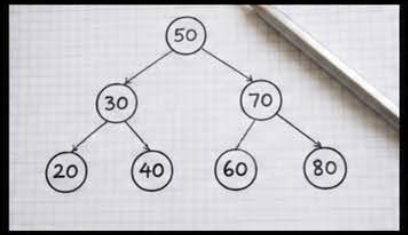
3. Programming Techniques

3.1a - Programming Fundamentals
3.1b - Producing Robust Programs
3.2a - Boolean Logic
3.2b - Programming Languages & IDEs
3.3 - Data Types & Structures
3.4a - Subprograms
3.4b - Recursion



2. Programming Fundamentals

2.1a - Programming Concepts
2.1b - Data Representation
2.2 - Programming Constructs
2.3a - Input & Output
2.3b - Files
2.4 - Defensive Design



4. Data Structures

4.1a - Basic Data Structures
4.1b - Arrays & Lists
4.1c - Records
4.2a - Stacks & Queues
4.2b - Trees
4.2c - Graphs

5. Standard Algorithms

5.1 - Searching Algorithms
5.2 - Sorting Algorithms

6. Programming Project

6.1 - Problem Analysis
6.2 - Design
6.3 - Development
6.4 - Testing & Evaluation

Recommended Course Books

- Tackling A Level Projects in Computer Science OCR H446Paperback – 30 Jan. 2020
- OCR AS and A Level Computer Science Paperback – by Heathcote
- OCR A/AS Level Computer Science Workbook by Alistair Surrall , Adam Hamflett
- A Level Computer Science Workbook OCR: isaac computer science 2020 paperback

Other Resources in Use

- OCR Website
- Craig & Dave Videos
- IsaacComputerScience.org
- PhysicandMathTutor.com
- peterhigginson.co.uk
- Microsoft Teams

Course Structure

- OCR H046, H446
- Paper 1, 150 mins (40%)
- Paper 2, 150 mins (40%)
- Programming Project – NEA (20%)

Programming Project

Students are expected to apply the principles of computational thinking to a practical coding programming project. They will analyse, design, develop, test, evaluate and document a program written in a suitable programming language.

Previous Project Idea

- Rocket Simulator
- Games
- Mobile Fitness Apps
- Machine Learning
- Satellite Simulator
- Minecraft Mod Investigation Modelling
- F1 Cornering Simulator
- Football Management Tools
- WebComics

OCR Computing Summer Project

Furze Platt Senior School wants to introduce a paid parking system.

The Headmaster wants a computer system that:

- Allows different users to park
- Calculates parking charges
- Prints receipts
- Uses a menu system

User Requirements:
The parking system should:

- Allow user to select user type
- Record vehicle or parking details
- Calculate parking cost
- Display total charge
- Print a receipt
- Use a clear menu system

Your Task : Decompose the problem and create a System that works with an interactive UI, a database and subroutines. It should also export data from the system to a .txt file. (You may add your own style and stretch. OOP Maybe?

The list below are prerequisites for this course. Each worksheet should be completed. You may Use them to guide you in completing the project. Please focus on refining your coding to acceptable A level standard. (If links do not work you may access via the Summer Work Folder Shared with you.)

Introductory Notes..... 00b-Introduction.pdf

1. The Basics.....01-TheBasics.pdf
2. Variables & Assignment Statements..... 02-Variables\u0026Assignment.pdf
3. Data Types 03-DataTypes.pdf
4. Procedures &Functions.....04-Procedures\u0026Functions.pdf
5. Structured Programming.....05-StructuredProgramming.pdf
6. Importing Modules.....06-ImportingModules.pdf
7. Scope of Variables and Functions..... 07-Scope.pdf
8. Lists (Arrays).....08-Lists.pdf
9. Tuples.....09-Tuples.pdf
10. Dictionaries..... 10-Dictionaries.pdf
11. File Handling 10-Dictionaries.pdf
12. Defensive Design.....12-DefensiveDesign.pdf
13. Testing.....13-Testing.pdf
14. Databases.....14-Databases.pdf
15. Simple GUI (Tkinter).....15-TkinterGUI.pdf

.

Lesson Breakdown

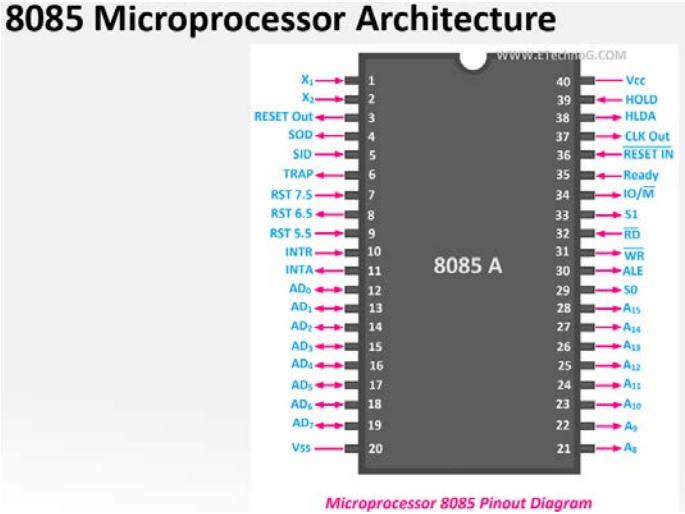
- Lesson Overview
- Recapping Basis Logic Gates, Truth Tables and Boolean Expressions
- Universal gates
- Boolean Expressions
- Breakout Activity – logic.ly

FURZE PLATT SENIOR SCHOOL

Lesson Title: Logic Gates

Boolean Logic

Purpose of the Lesson



Modern computing systems are built upon a set of fundamental principles that enable them to **process** and **store** vast amounts of information.

One of the most important of these principles is **Boolean logic** , implemented through **logic gates** , which form the foundation of all **digital circuits** and computer **processors**

Lesson Objectives

- Identify and describe the function of the basic logic gates: **AND** , **OR**, and **NOT** .
- Recognise and interpret the truth tables for **XOR**, NAND, and **NOR** gates.
- Construct and complete truth tables for simple and compound logic circuits.
- Convert Boolean expressions into logic gate circuits using BNAO (Brackets, **NOT** , **AND** , **OR**).
- Write Boolean expressions from existing logic circuits.

What are logic gates and why are they important in computer science?

- Frequently used in integrated circuits
- Logic gates are one of the fundamental electronic devices of digital circuits.
- They perform basic logical operations on one or more input digital signals to produce a single output signal.

How does the computer use logic gates?

- In computing, all information is represented using only two states: 1 (on) and 0 (off). Computers use billion of tiny electronic components called transistors to implement logic gates. These transistors act as switches that can be either on or off (binary 1 and 0).

Why is this important?

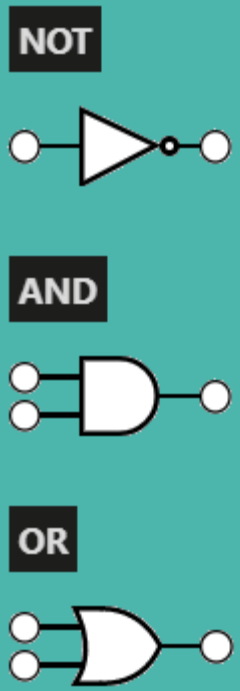
Every **calculation** , **decision** , and **memory operation** performed by a computer relies on networks of **logic gates** working together.

Over time, engineers have found ways to make these circuits smaller, simpler, and more efficient.

The result?

- Faster smartphones, more powerful gaming consoles, and increasingly capable computers.

Recapping the Basics



New Gates

NAND

NOR

XOR

Which Truth Table?

Table 1		
In A	In B	Out
0	0	0
0	1	0
1	0	0
1	1	1

Table 2	
In	Out
0	1
1	0

Table 3		
In A	In B	Out
0	0	0
0	1	1
1	0	1
1	1	1

Which Truth Table?

AND		
InA	InB	Out
0	0	0
0	1	0
1	0	0
1	1	1

NOT	
In	Out
0	1
1	0

OR		
InA	InB	Out
0	0	0
0	1	1
1	0	1
1	1	1

New Gates

NAND

NOR

XOR

Which Truth Table?

Table 1		
InA	InB	Out
0	0	0
0	1	1
1	0	1
1	1	0

Table 2		
InA	InB	Out
0	0	1
0	1	1
1	0	1
1	1	0

Table 3		
InA	InB	Out
0	0	1
0	1	0
1	0	0
1	1	0

Which Truth Table?

XOR (eXcusive OR)		
InA	InB	Out
0	0	0
0	1	1
1	0	1
1	1	0

NAND (Not AND)		
InA	InB	Out
0	0	1
0	1	1
1	0	1
1	1	0

NOR (Not OR)		
InA	InB	Out
0	0	1
0	1	0
1	0	0
1	1	0

Universal Gates

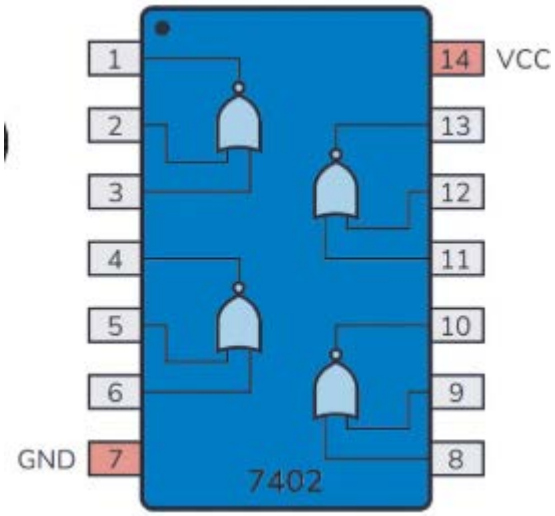
NAND & NOR

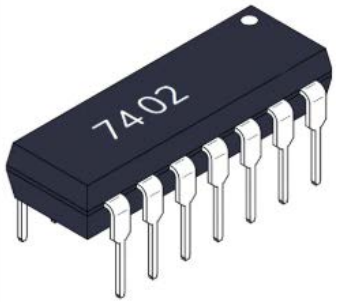
Universal Gates

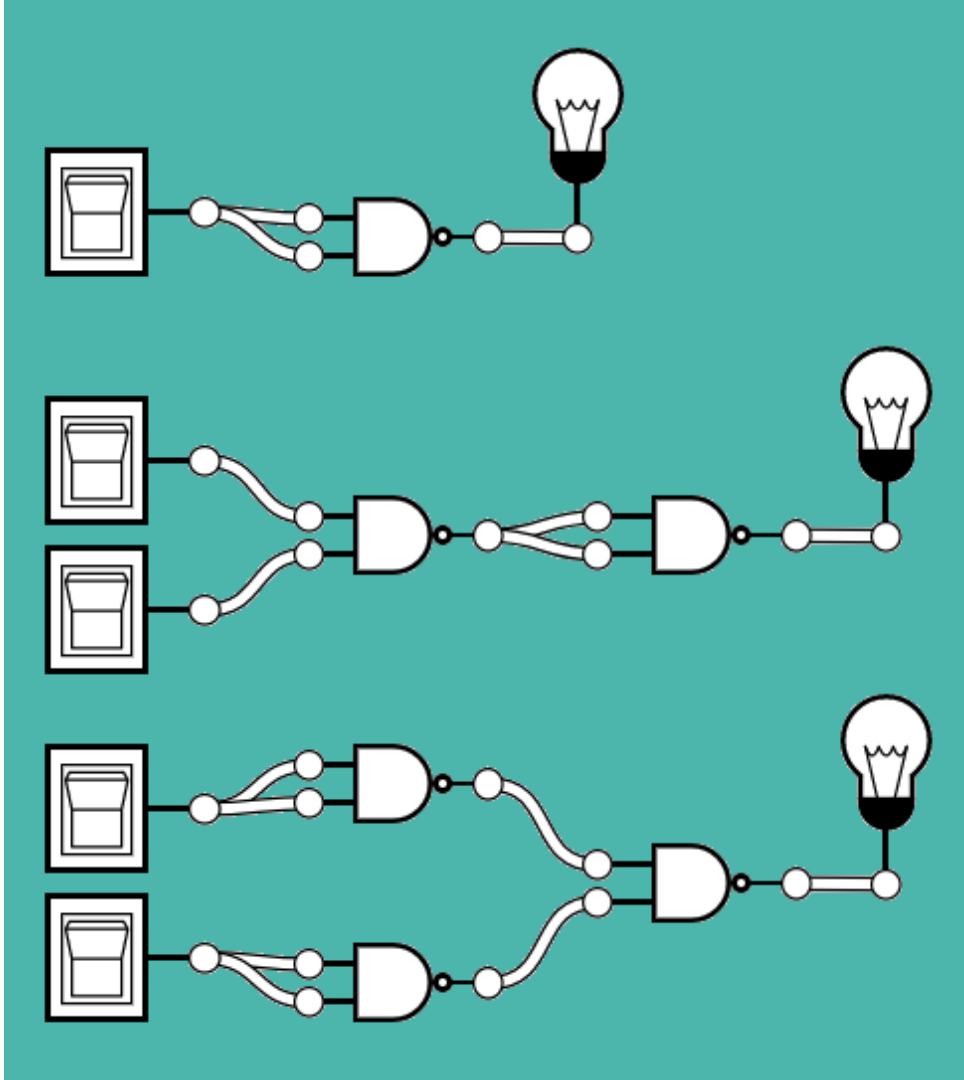
NAND & NOR

- versatile logic gates that can be used to construct *any* other Boolean logic function or gate
- simplifies the fabrication of microprocessors and integrated circuits
- reduces the number of unique components required

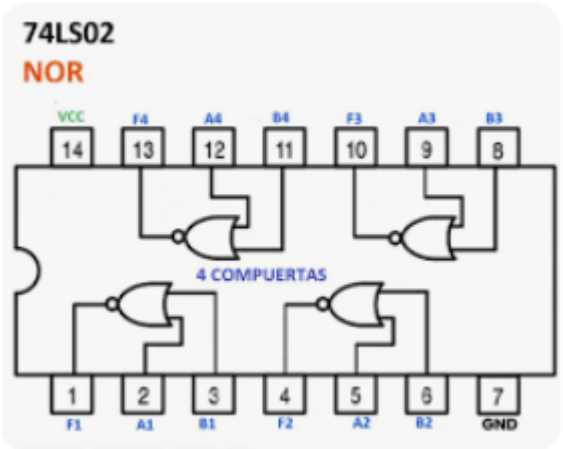
Universal Gate: NAND

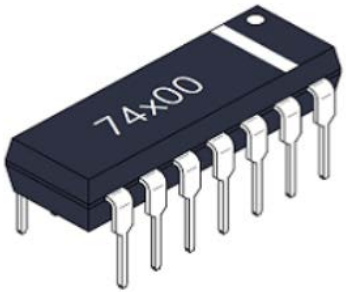


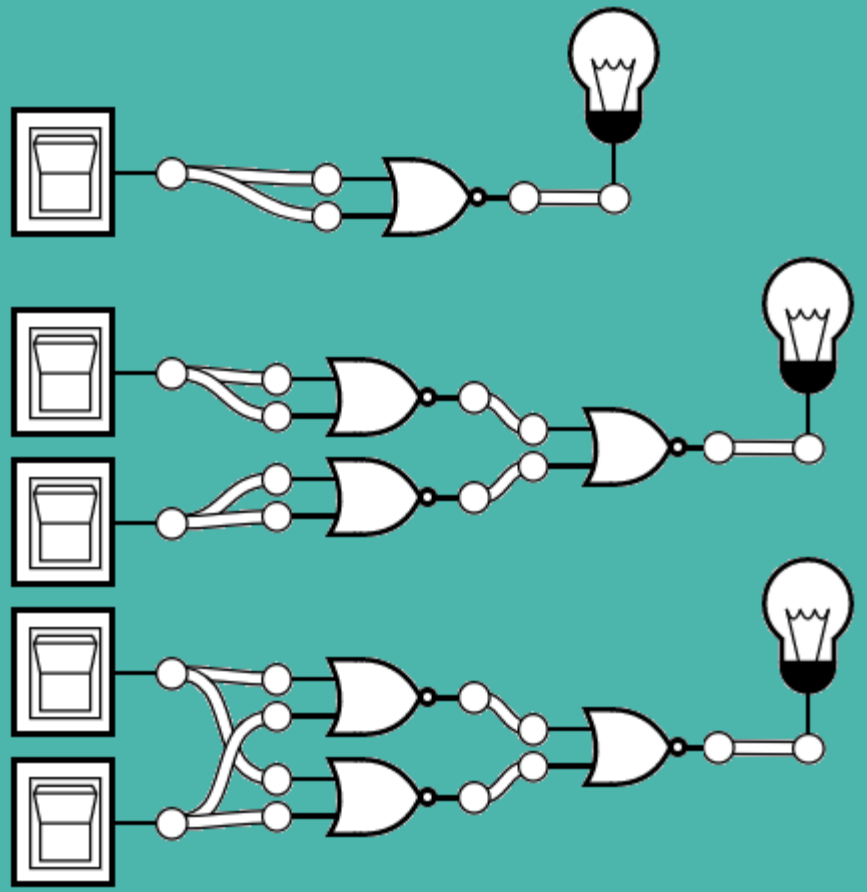




Universal Gate: N







Boolean Algebra

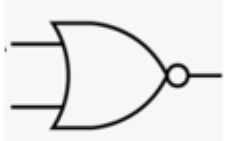
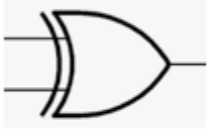
Recap Boolean Algebra

Basic Gates

Operation	Conjunction	Disjunction	Negation
Logic gate			
	AND	OR	NOT
Symbol	$\wedge$	$\vee$	$\neg$

Boolean Algebra OCR

Advanced Gates

Logic Gate	NOR	XOR	NAND
Symbol			
Boolean Expression	$\neg(A \vee B)$	$(A \wedge \neg B) \vee (\neg A \wedge B)$ OR $A \oplus B$	$\neg(A \wedge B) = \neg A \vee \neg B$

Break out Activity

Task

Logic Gates Activities

1. Use the link *https://logic.ly/demo/* to assess logic.ly from your browser
2. Complete the activities on Slide 30 and continue working through to Slide 38

(Complete as many as you can)

Expressions to Circuits

BNAΘBrackets, Not, And, Or
(Like BODMAS / BIDMAS)

Logic.ly Demo

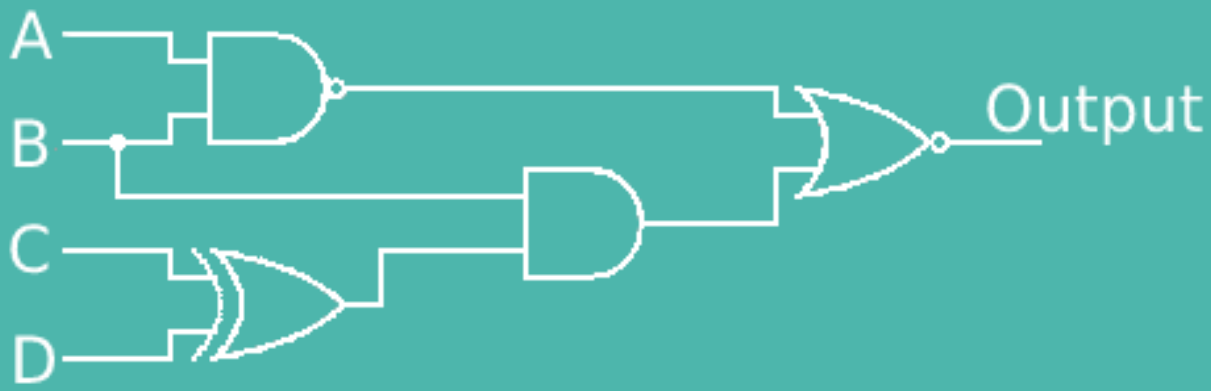
Can you create circuits for the following equations using logic.ly/demo?

$U = \neg(A \oplus B)$	$X = A \vee (B \oplus C)$
$V = \neg(A \wedge B) \vee C$	$Y = A \wedge \neg(B \vee C)$
$W = \neg(A \vee (B \wedge \neg C))$	$Z = (A \wedge B) \vee \neg(D \wedge E)$

Circuit to Expression

BNAΘBrackets, Not, And, Or
(Like BODMAS / BIDMAS)

Create the expression for the circuit below

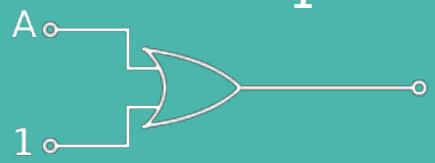


Expression

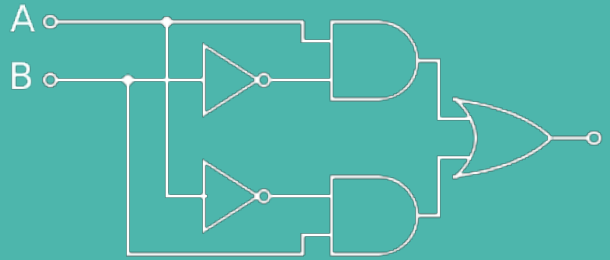
Equivalent Circuits

Each circuit has a counterpart. Can you match them up

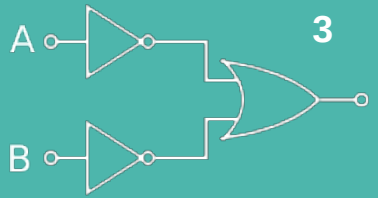
1



2

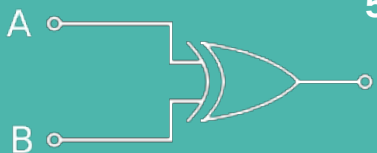


3



4

5



6

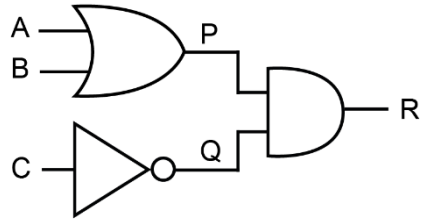
Equivalent Expressions

Each expression has a counterpart. Can you match them

You may need to do some truth tables.

1. ¬(A∨B)
2. ¬(A∧B)
3. $A \wedge B) \vee A$
4. ¬A∨¬B
5. 0
6. $\neg(\neg(A \vee B))$
7. ¬(¬(A∨B))
8. $A \wedge \neg A$
9. A∨B
10. ¬(A ∨ B)
11. $\wedge A \wedge B$
12. A
13. ¬A∧¬B
14. $A \wedge B$

Complete the truth table for the following l circuit

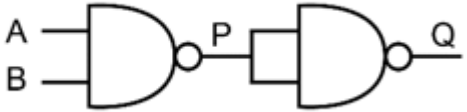


Input A	Input B	Input C	P = A OR B	Q = NOT C	Output R = (A OR B) AND (NOT C)
0	0	0			
0	0	1			
0	1	0			
0	1	1			
1	0	0			
1	0	1			
1	1	0			
1	1	1			

Complete truth tables for each of the following logic circuits to show that they are equivalent.

(i)

(ii)



Truth table (i)

Input A	Input B	Output Q	
0	0		
0	1		
1	0		
1	1		

Truth table (ii)

Input A	Input B	P = NOT (A AND B)	Q = NOT (P AND P)

Scenario 1: A security system should activate an alarm when the window sens
door sensor (D) is triggered, and the system is armed (A).

Boolean Expression: (W OR D) AND A

- Create three switches labelled W (Window), D (Door), and A (Armed)
- Connect W and D to an OR gate.
- Connect the OR gate output and A to an AND gate.
- Connect the final output to a lamp labelled 'Alarm'.
- Test all possible input combinations and complete the truth table.

W	D	A	Alarm
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Challenge Extension:

Rebuild the same circuit using only NAND gates and co designs.

W	D	A	Alarm
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Challenge Extension:

Scenario 2: users of a computer system have a unique usern
password. The computer system has implemented two-factor authentication sc
that users must respond to either an email or text message contai
code to be able to access the system.

Let:

- **A** be a Boolean value for if a user enters a valid username
- **B** be a Boolean value for if a user enters a password that matches their username
- **C** be a Boolean value for if a user is able to respond to an email containing a secret code
- **D** be a Boolean value for if a user is able to respond to a text message containing a secret code
- **Q** be a Boolean value for if entry to the computer system is allowed

Task: Create the Boolean Expression and Logic Circuit from this scenario.

Expression Match Up Answers

Here are the equivalent statements grouped together

1. ¬(A ∨ B) = **10.** ¬(A ∨ B) = **13.** ¬A ∧ ¬B

2. ¬(A ∧ B) = **4.** ¬A ∨ ¬B

3. (A ∧ B) ∨ A = **12.** A

5. 0 = **8.** A ∧ ¬A

6. ¬(¬(A ∨ B)) = **7.** ¬(¬(A ∨ B)) = **9.** A ∨ B

11. A ∧ B = **14.** A ∧ B

Plenary

You have learned

- about the functions of basic logic gates.
- how to interpret the truth tables for **XOR**, **NAND** , and **NOR** gates.
- how to convert Boolean expressions into logic gate circuits.
- how to convert Boolean expressions into logic gate circuits.
- how to write Boolean expressions from existing logic circuits.