A1.2.3 : Describe the Purpose and Use of Logic Gates **2**
1. What are Logic Gates? 2
2. Importance of Logic Gates 2
3. Common Logic Gates and Their Uses 2
4. How Logic Gates Work 3
5. Real-World Applications 4
6. Combining Logic Gates 4
7. Summary Table 4
Questions for Revision 6
A1.2.4 : Construct and Analyse Truth Tables **6**
1. What is a Truth Table? 6
3. Truth Tables for Basic Logic Gates 7
4. Analysing a Truth Table 9
6. Real-World Example: Automatic Door 11
7. Common Mistakes to Avoid 12
8. Summary Table 12
Questions for Revision 13
A1.2.5 — Construct Logic Diagrams **14**
1. Logic Diagrams: Overview 14
2. Standard Logic Gates and Symbols 14
3. Boolean Algebra 15
Key Boolean Laws 15
4. Simplifying Boolean Expressions 16
Example 1 – Using Absorption 16
Example 2 – Using Distributive and Inverse 16
Example 3 – Using De Morgan's Law 17
Example 4 – Complex Simplification 17
5. Simplifying Logic Diagrams Using Boolean Laws 17
6. De Morgan's Law in Circuit Form 17
7. Simplifying Using Karnaugh Maps (K-maps) 18
Steps 18
8. 2-Variable K-map Example 18
9. 3-Variable K-map Example 19
10. Grouping Rules 21
11. Example Using De Morgan + K-map 21
12. Real-World Relevance 21
13. Summary Table 22
Questions for Revision 23

A1.2.3 : Describe the Purpose and Use of Logic Gates

1. What are Logic Gates?

- **Definition:**
  Logic gates are **electronic components** that perform logical operations on binary inputs (0s and 1s) to produce a single binary output.
  Each gate follows a specific logical rule (such as AND, OR, or NOT).

- **Purpose:**
  Logic gates are the **basic building blocks** of digital circuits found in all computers and electronic systems.

- **Example:**
  A **security door** opens only when both the password and keycard are verified — this uses an **AND** gate.

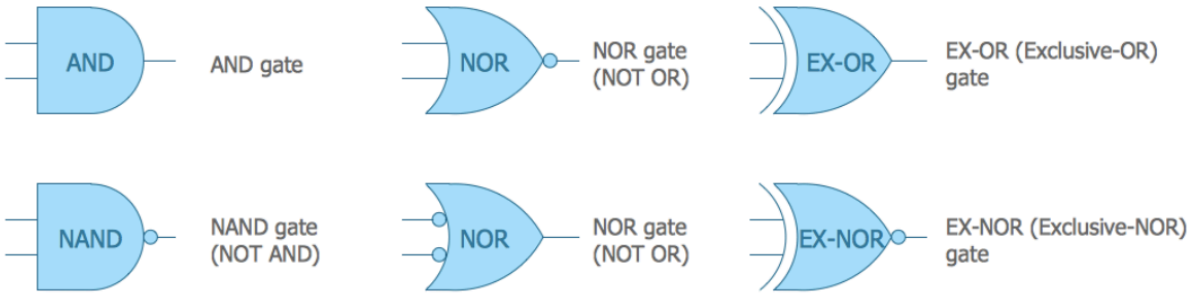
2. Importance of Logic Gates

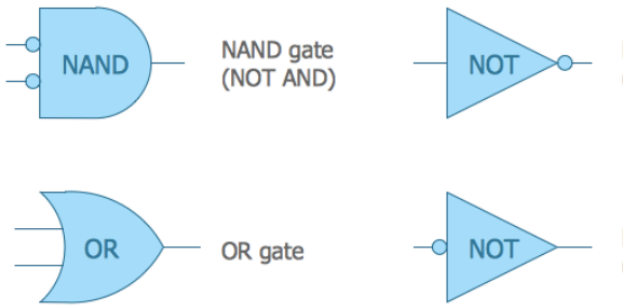
- **Decision-making:** Gates control whether an output should be ON or OFF based on conditions.

- **Foundation of computation:** Every microprocessor and memory circuit operates using millions of logic gates.

- **Used in control systems:** From washing machines to elevators, gates decide when actions should happen.

3. Common Logic Gates and Their Uses





Logic Gate	Symbol / Name	Operation / Description	Real-World Example
AND		Output is 1 only if **both inputs are 1**	A microwave starts only if the door is closed **and the timer** is set.
OR		Output is 1 if **any input is 1**	A room light turns on if **either** of two switches is pressed.
NOT		Inverts the input (1 → 0, 0 → 1)	A motion detector turns **off** lights when **no motion** is detected.
NAND		Opposite of AND (output 0 only if both inputs are 1)	Used in **alarm systems** to check for faulty signals.
NOR		Opposite of OR (output 1 only if all inputs are 0)	Used in **reset systems** to stop an operation.
XOR (Exclusive OR)		Output is 1 if **inputs differ**	Used in **parity checking** to detect errors.

XNOR (Exclusive NOR)		Output is 1 if **inputs are the same**	Used in **digital comparators** to compare signals.

4. How Logic Gates Work

Gate	Input A	Input B	Output	Explanation
AND	1	1	1	Both inputs ON gives ON
AND	1	0	0	One input OFF → result OFF
OR	1	0	1	At least one ON gives ON
NOT	1	—	0	Inverts the input
XOR	1	0	1	Inputs are different
XOR	1	1	0	Inputs same → OFF

5. Real-World Applications

Application	Gate Used	Purpose
Security alarm	AND	Triggers when two sensors detect movement
Automatic door	OR	Opens if button **or** sensor is activated
Safety lock	NAND	Ensures door won't open under unsafe conditions
Voting circuit	XOR	Detects when only one button is pressed
Thermostat	NOT	Turns cooling ON when temperature is **not low**

6. Combining Logic Gates

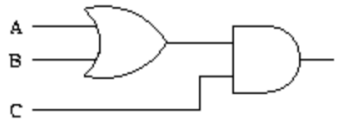
- Logic gates can be **combined** to perform more complex decisions.
  Example 1:
  Output = (Temperature High **AND** Fan ON) **OR** (Humidity High)
  → The fan runs if it's hot and turned on, or if humidity is high.
  Example 2:
  Output = **NOT** (Door Open **OR** Motion Detected)
  → Alarm only when there is **no motion** and **door closed**.

7. Summary Table

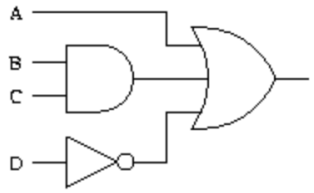
Gate	Inputs	Output Rule	Boolean Expression

AND	2	1 if both inputs are 1	$A \cdot B$
OR	2	1 if either input is 1	A + B
NOT	1	Inverts input	¬A or A’
NAND	2	Opposite of AND	$\neg(A \cdot B)$
NOR	2	Opposite of OR	¬(A + B)
XOR	2	1 if inputs differ	$A \oplus B$
XNOR	2	1 if inputs same	$\neg(A \oplus B)$

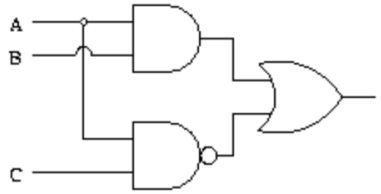
- Draw a logic circuit for $(A + B)C$.



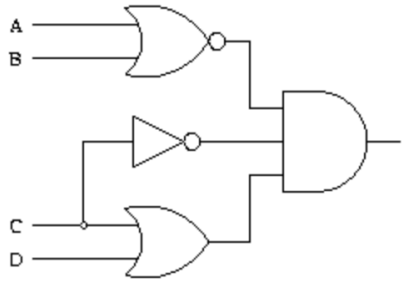
- Draw a logic circuit for $A + BC + \overline{D}$.



- Draw a logic circuit for $AB + \overline{AC}$.



- Draw a logic circuit for $\overline{(A + B)}(C + D)\overline{C}$.



Questions for Revision

1. **Define** a logic gate.
2. **Describe** the function of an AND gate using a real-world example.
3. **Identify** which logic gate outputs 1 only when both inputs are 0.
4. **Distinguish** between XOR and XNOR gates.
5. **Explain** the purpose of combining multiple logic gates in one circuit.
6. **State** the Boolean expression for a NOR gate.
7. **Describe** how an OR gate can be used in a smart home lighting system.

8. **Explain** why NAND gates are considered universal gates.
9. **Identify** one use of a NOT gate in an automatic system.
10. **Draw** or describe a simple example where multiple gates control an outcome.

A1.2.4 : Construct and Analyse Truth Tables

1. What is a Truth Table?

- **Definition:**
  A **truth table** is a table used to show **all possible input combinations** for a logic circuit and their corresponding **outputs**.
  It helps us **predict and analyse** how logic gates or circuits behave for every possible input.

- **Purpose:**
  Truth tables are used to:

  - Check if a logic circuit performs as intended
  - Understand relationships between inputs and outputs
  - Simplify complex logical expressions

- **Example:**
  If a security system requires **both** a keycard (A) and a PIN (B) to unlock, we can use a truth table to verify that it only opens when **A = 1** and **B = 1**.

2. Structure of a Truth Table

A truth table lists **every possible combination of inputs** (each can be 0 or 1) and shows the corresponding output.

Number of Inputs	**Number of Possible Combinations**
1	2 (0, 1)

2	4 (00, 01, 10, 11)
3	8 (000 to 111)
4	16 combinations

Each input combination shows how the output changes depending on the logic gate.

---

3. Truth Tables for Basic Logic Gates

Gate	A	B	Output	Explanation
AND	0	0	0	Both OFF → output OFF
	0	1	0	One OFF → output OFF
	1	0	0	One OFF → output OFF
	1	1	1	Both ON → output ON
OR	0	0	0	Both OFF → output OFF
	0	1	1	At least one ON → output ON
	1	0	1	At least one ON → output ON

	1	1	1	Both ON → output ON
NOT	0	—	1	Inverts input
	1	—	0	Inverts input
NAND	0	0	1	Opposite of AND
	0	1	1	Opposite of AND
	1	0	1	Opposite of AND
	1	1	0	Opposite of AND
NOR	0	0	1	Opposite of OR
	0	1	0	Opposite of OR
	1	0	0	Opposite of OR
	1	1	0	Opposite of OR
XOR	0	0	0	Same inputs → OFF

	0	1	1	Inputs differ → ON
	1	0	1	Inputs differ → ON
	1	1	0	Same inputs → OFF
XNOR	0	0	1	Same inputs → ON
	0	1	0	Inputs differ → OFF
	1	0	0	Inputs differ → OFF
	1	1	1	Same inputs → ON

4. Analysing a Truth Table

To **analyse** a truth table:

1. **List all input combinations** (binary patterns).
2. **Apply the logic rule** of the gate or circuit for each combination.
3. **Observe output patterns** to understand how the circuit behaves.

Example:
Expression: $Q=A \cdot B+C$ $\overline{Q} = A \cdot B + \overline{C}$ $Q=A \cdot B+C$
This means:

- $A \cdot B$ $A \cdot B$ → both A and B must be 1

- C¬\overline{C}C → C must be 0
- The OR (+) means **either condition** gives output 1

A	B	C	A · B	¬C	Q = (A · B + ¬C)
0	0	0	0	1	1
0	0	1	0	0	0
0	1	0	0	1	1
1	1	1	1	0	1
1	1	0	1	1	1

Interpretation:
The output Q = 1 for most cases except when all conditions fail (A · B = 0 and ¬C = 0).

5. Combining Gates in a Truth Table

You can use truth tables to test **combinations of logic gates**:

- Step 1: Identify **sub-expressions** (e.g., A · B, ¬C).
- Step 2: Calculate intermediate outputs.
- Step 3: Determine final output.

Example:
Expression: Q=(A+B) · ¬CQ = (A + B) \cdot ¬CQ=(A+B) · ¬C

A	B	C	A + B	¬C	Q = (A + B) · ¬C
0	0	0	0	1	0
0	1	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	1

Interpretation:
The output Q = 1 when **either A or B is 1**, and **C is 0**.

6. Real-World Example: Automatic Door

A door opens if:

- The **sensor detects motion (A = 1)**
- The **door is not locked (B = 0)**

Expression: Q=A · ¬BQ = A \cdot ¬BQ=A · ¬B

A (Motion)	B (Locked)	¬B	Q (Open Door)
0	0	1	0
0	1	0	0

1	0	1	1
1	1	0	0

Interpretation:
The door opens only when motion is detected and the lock is off — exactly what's expected.

7. Common Mistakes to Avoid

- Forgetting to include **all input combinations**.
- Mixing up the order (always list binary in order: 00, 01, 10, 11).
- Confusing **AND (·)** and **OR (+)** operations.
- Forgetting to apply **NOT (¬)** to the correct variable.

8. Summary Table

Concept	Description
Purpose	Show all possible input-output combinations
Inputs	Binary (0 or 1)
Output	Depends on logical rule
Used for	Testing circuits, verifying expressions

Key operations	AND, OR, NOT, NAND, NOR, XOR, XNOR

Questions for Revision

1. **Define** a truth table.
2. **State** how many rows are needed in a truth table with 3 inputs.
3. **Construct** a truth table for the expression Q=A+BQ = A + BQ=A+B.
4. **Analyse** the output of Q=¬A⋅BQ = ¬A \cdot BQ=¬A⋅B when A = 0 and B = 1.
5. **Explain** why truth tables are useful in designing digital circuits.
6. **Identify** which gate gives output 1 only when inputs differ.
7. **Construct** a truth table for a NOR gate.
8. **Describe** how a truth table helps detect logical errors in a circuit.
9. **Complete** the truth table for Q=(A⋅B)+¬CQ = (A \cdot B) + ¬CQ=(A⋅B)+¬C.
10. **Discuss** one real-world example where a truth table can be used to verify circuit output.

A1.2.5 — Construct Logic Diagrams

1. Logic Diagrams: Overview

A **logic diagram** is a **graphical representation** of a digital circuit showing how **logic gates** are connected to process inputs and produce an output.
They are used to:

- Visualize logical relationships.
- Simplify design before hardware implementation.
- Verify Boolean expressions using truth tables.

Example:
A burglar alarm activates (output = 1) only if the window is open (A = 1) **and** motion is detected (B = 1).
→ Expression: $Q=A \cdot BQ = A \cdot BQ=A \cdot B$
→ Logic diagram: two inputs to an AND gate.

---

2. Standard Logic Gates and Symbols

Gate	Symbol Description	Boolean Expression	Logic Rule	Common Use
AND	D-shaped symbol	$Q = A \cdot B$	1 only if both inputs = 1	Safety switches
OR	Curved input	$Q = A + B$	1 if at least one input = 1	Multi-switch circuits
NOT	Triangle + small circle	$Q = \neg A$	Output inverts input	Control signals

NAND	AND + small circle	$Q = \neg(A \cdot B)$	Opposite of AND	Fail-safe designs
NOR	OR + small circle	$Q = \neg(A + B)$	Opposite of OR	Reset systems
XOR	OR with extra line	$Q = A \oplus B$	1 if inputs differ	Parity check
XNOR	XOR + circle	$Q = \neg(A \oplus B)$	1 if inputs same	Equality check

---

3. Boolean Algebra

Boolean algebra uses symbols ($\cdot$, +, ¬) to describe logic operations and apply **laws** for simplification.

Key Boolean Laws

Category	**Law**	**Expression**	**Meaning / Example**
Identity	$A + 0 = A$ $A \cdot 1 = A$	Adding 0 or ANDing with 1 changes nothing	
Null (Domination)	$A + 1 = 1$ $A \cdot 0 = 0$	OR with 1 = 1; AND with 0 = 0	
Idempotent	$A + A = A$ $A \cdot A = A$	Repeating same input has no effect	
Inverse	$A + \neg A = 1$ $A \cdot \neg A = 0$	A or not-A is always true; A and not-A is false	

Commutative	$A + B = B + A$ $A \cdot B = B \cdot A$	Order doesn't matter	
Associative	$(A + B) + C = A + (B + C)$ $(A \cdot B) \cdot C = A \cdot (B \cdot C)$	Grouping doesn't matter	
Distributive	$A \cdot (B + C) = A \cdot B + A \cdot C$	AND distributes over OR	
Absorption	$A + A \cdot B = A$ $A \cdot (A + B) = A$	Redundant terms removed	
Double Negation	$\neg(\neg A) = A$	Two NOTs cancel	
De Morgan's Laws	$\neg(A + B) = \neg A \cdot \neg B$ $\neg(A \cdot B) = \neg A + \neg B$	Inverts group operations	

---

4. Simplifying Boolean Expressions

Simplification reduces complex logic into fewer gates.

Example 1 – Using Absorption

$Q=A+A \cdot BQ = A + A \cdot BQ=A+A \cdot B$
→ Apply absorption rule → $Q=AQ = AQ=A$
✅ Simplified circuit: one wire from A to Q.

Example 2 – Using Distributive and Inverse

$Q=A \cdot B+A \cdot \neg BQ = A \cdot B + A \cdot \neg BQ=A \cdot B+A \cdot \neg B$
→ Factor A → $Q=A(B+\neg B)Q = A(B + \neg B)Q=A(B+\neg B)$
→ $(B+\neg B)=1(B + \neg B) = 1(B+\neg B)=1$
→ $Q=AQ = AQ=A$
✅ Only one input needed.

Example 3 – Using De Morgan's Law

$Q=\neg(A+B)Q = \neg(A + B)Q=\neg(A+B)$
$\rightarrow Q=\neg A \cdot \neg BQ = \neg A \cdot \neg BQ=\neg A \cdot \neg B$
✅ Equivalent to NOR gate replaced by two NOTs and one AND.

Example 4 – Complex Simplification

$Q=\neg(A \cdot B)+AQ = \neg(A \cdot B) + AQ=\neg(A \cdot B)+A$
$\rightarrow$ Apply De Morgan: $Q=(\neg A+\neg B)+AQ = (\neg A + \neg B) + AQ=(\neg A+\neg B)+A$
$\rightarrow$ Rearrange: $(\neg A+A)+\neg B(\neg A + A) + \neg B(\neg A+A)+\neg B$
$\rightarrow 1+\neg B=11 + \neg B = 11+\neg B=1$
✅ Q = 1 (always true).

---

5. Simplifying Logic Diagrams Using Boolean Laws

1. **Write** the Boolean expression.
2. **Apply** appropriate laws step by step.
3. **Combine** identical terms.
4. **Remove** redundant gates from the logic diagram.
5. **Verify** with a truth table.

Example:
$Q=A \cdot B+A \cdot \neg B+B \cdot CQ = A \cdot B + A \cdot \neg B + B \cdot CQ=A \cdot B+A \cdot \neg B+B \cdot C$
$\rightarrow A \cdot (B+\neg B)+B \cdot CA \cdot (B + \neg B) + B \cdot CA \cdot (B+\neg B)+B \cdot C$
$\rightarrow A+B \cdot CA + B \cdot CA+B \cdot C$
✅ From 3 AND + 1 OR → 1 AND + 1 OR gate.

---

6. De Morgan's Law in Circuit Form

Original	**Equivalent (De Morgan)**	**Gate Conversion**
$\neg(A + B)$	$\neg A \cdot \neg B$	OR → AND + inverters

¬(A · B)	¬A + ¬B	AND → OR + inverters

Example:
A NOR gate (¬(A + B)) can be made from **two NOTs** feeding into an **AND**.
Similarly, a NAND gate equals **two NOTs** into an **OR**.

---

7. Simplifying Using Karnaugh Maps (K-maps)

Link

Purpose:
K-maps provide a visual way to simplify Boolean expressions by grouping 1s from the truth table.

Steps

1. Draw grid using Gray code order.
2. Plot 1s from truth table.
3. Form groups of 1, 2, 4, 8… (power of 2).
4. Each group → one product term.
5. Combine all groups with OR (+).
6. Verify simplification.

---

8. 2-Variable K-map Example

Expression: $Q=A \cdot B+A \cdot \neg BQ = A \cdot B + A \cdot \neg BQ=A \cdot B+A \cdot \neg B$

A	**B**	**Q**

0	0	0
0	1	0
1	0	1
1	1	1

K-map

A＼B	**0**	**1**
0	0	0
1	1	1

→ One horizontal group (A = 1)
✅ **Q = A**

---

9. 3-Variable K-map Example

Expression: $Q=A \cdot B \cdot C+A \cdot B \cdot \neg C+A \cdot \neg B \cdot C$Q = A · B · C + A · B · ¬C + A · ¬B · CQ=A · B · C+A · B · ¬C+A · ¬B · C

Truth Table:

A	**B**	**C**	**Q**

0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

K-map

A＼BC	**00**	**01**	**11**	**10**
0	0	0	0	0
1	0	1	1	1

→ Group (B = 1, A = 1) → $A \cdot B$
→ Single 1 (A = 1, B = 0, C = 1) → $A \cdot \neg B \cdot C$
Simplify → **Q = A(B + C)**

---

10. Grouping Rules

Group Size	Effect	Variables Removed
1	No simplification	0
2	Adjacent cells	1
4	Square/rectangle	2
8	Whole map	3

Wrap-around grouping allowed (edges connect).

---

11. Example Using De Morgan + K-map

Expression: $Q=\neg(A \cdot B)+A \cdot C$ $Q = \neg(A \cdot B) + A \cdot C$ $Q=\neg(A \cdot B)+A \cdot C$
→ Apply De Morgan: $\neg A+\neg B+A \cdot C$ $\neg A + \neg B + A \cdot C$ $\neg A+\neg B+A \cdot C$
→ On K-map, group cells for ¬A and ¬B = 1 → redundant → simplifies to **$Q = \neg B + A \cdot C$**

---

12. Real-World Relevance

System	Logic	Simplified Function
Smart lamp	(LightSensor = 0 AND Motion = 1) OR Override	Automatic activation

Car safety lock	(Brake = 1 AND Gear = P) → Unlock	Ensures safety conditions
CPU control	Boolean rules applied to optimize instruction decoding	Faster execution

---

13. Summary Table

Concept	**Key Idea**
Logic diagrams	Show gate connections and interactions
Boolean laws	Algebraic rules for simplification
De Morgan’s laws	Invert group operations
K-maps	Visual simplification tool
Simplification goal	Fewer gates → efficient circuits
Verification	Always confirm with truth table

Questions for Revision

1. **Define** a logic diagram.
2. **State** the symbols for all seven standard logic gates.
3. **List** four fundamental Boolean laws.
4. **Apply** the absorption law to simplify A+A⋅BA + A \cdot BA+A⋅BA+A⋅B.
5. **Use** De Morgan's law to simplify ¬(A⋅B)¬(A ⋅ B)¬(A⋅B).
6. **Simplify** Q=A⋅B+A⋅¬BQ = A \cdot B + A \cdot \neg BQ=A⋅B+A⋅¬BQ=A⋅B+A⋅¬B using Boolean laws.
7. **Construct** the K-map for Q=A⋅B+A⋅¬BQ = A \cdot B + A \cdot \neg BQ=A⋅B+A⋅¬BQ=A⋅B+A⋅¬B.
8. **Group** adjacent 1s in a 3-variable K-map and write the simplified result.
9. **Explain** why Gray code ordering is used in K-maps.
10. **Discuss** how Boolean simplification reduces circuit complexity.

1. Expression: A + A·B

Laws that can be used:

- **Identity Law:** X = X·1
- **Distributive Law:** A·(B + C) = A·B + A·C
- **Absorption Law:** A + A·B = A

Step	Working	Law Used
1	A + A·B	Given Expression
2	A·1 + A·B	Identity Law

3	A·(1 + B)	Distributive Law
4	A·1	Null Law (1 + B = 1)
5	**A**	Identity Law

✅ **Simplified Expression:** A

---

2. Expression: A·(A + B)

Laws that can be used:

- **Distributive Law:** A·(B + C) = A·B + A·C
- **Idempotent Law:** A + A = A
- **Absorption Law:** A·(A + B) = A

Step	Working	Law Used
1	A·(A + B)	Given Expression
2	A·A + A·B	Distributive Law
3	A + A·B	Idempotent Law (A·A = A)
4	**A**	Absorption Law

✅ **Simplified Expression:** A

---

3. Expression: A + Ā·B

Laws that can be used:

- **Distributive Law:** A + B·C = (A + B)·(A + C)
- **Complement Law:** A + Ā = 1
- **Identity Law:** 1·X = X

Step	Working	Law Used
1	A + Ā·B	Given Expression
2	(A + Ā)·(A + B)	Distributive Law
3	1·(A + B)	Complement Law (A + Ā = 1)
4	**A + B**	Identity Law (1·X = X)

✅ **Simplified Expression:** A + B

---

4. Expression: (A + B)·(A + Ā)

Laws that can be used:

- **Complement Law:** A + Ā = 1
- **Identity Law:** X·1 = X

Step	Working	Law Used
1	(A + B)·(A + Ā)	Given Expression
2	(A + B)·1	Complement Law (A + Ā = 1)
3	**A + B**	Identity Law

✅ **Simplified Expression:** A + B

---

5. Expression: (A + B)·(A + C)

Laws that can be used:

- **Distributive Law:** (A + B)·(A + C) = A + B·C

Step	Working	Law Used
1	(A + B)·(A + C)	Given Expression
2	**A + B·C**	Distributive Law

✅ **Simplified Expression:** A + B·C

---

6. Expression: (A·B) + (A·B̄)

Laws that can be used:

- **Distributive Law:** A·B + A·C = A·(B + C)
- **Complement Law:** B + $\bar{B}$ = 1
- **Identity Law:** A·1 = A

Step	Working	Law Used
1	A·B + A·$\bar{B}$	Given Expression
2	A·(B + $\bar{B}$)	Distributive Law
3	A·1	Complement Law
4	**A**	Identity Law

✅ **Simplified Expression:** A

---

7. Expression: $\bar{A}$·B + A·B

Laws that can be used:

- **Distributive Law:** A·B + $\bar{A}$·B = (A + $\bar{A}$)·B
- **Complement Law:** A + $\bar{A}$ = 1
- **Identity Law:** 1·X = X

Step	Working	Law Used

1	Ā·B + A·B	Given Expression
2	(Ā + A)·B	Distributive Law
3	1·B	Complement Law
4	**B**	Identity Law

✅ **Simplified Expression:** B

---

8. Expression: (A·B)’

Laws that can be used:

- **De Morgan’s Theorem:** (A·B)’ = A’ + B’

Step	Working	Law Used
1	(A·B)’	Given Expression
2	**A’ + B’**	De Morgan’s Theorem

✅ **Simplified Expression:** A’ + B’

---

9. Expression: (A + B)’

Laws that can be used:

- **De Morgan's Theorem:** (A + B)' = A'·B'

Step	Working	Law Used
1	(A + B)'	Given Expression
2	**A'·B'**	De Morgan's Theorem

✅ **Simplified Expression:** A'·B'

---

10. Expression: (A + B)' + A

Laws that can be used:

- **De Morgan's Theorem:** (A + B)' = A'·B'
- **Distributive Law:** X + X·Y = X
- **Absorption Law:** A + A·B = A

Step	Working	Law Used
1	(A + B)' + A	Given Expression
2	A'·B' + A	De Morgan's Theorem
3	**A + B'**	Simplification using Absorption Law

✅ **Simplified Expression:** A + B'

Practice questions.

Simplify the following boolean expressions.

1. Expression: A + A·B
Laws that can be used: Distributive Law, Absorption Law, Identity Law

2. Expression: A·(A + B)
Laws that can be used: Distributive Law, Idempotent Law, Absorption Law

3. Expression: A + Ā·B
Laws that can be used: Distributive Law, Complement Law, Identity Law

4. Expression: (A + B)·(A + Ā)
Laws that can be used: Complement Law, Identity Law

5. Expression: (A + B)·(A + C)
Laws that can be used: Distributive Law, Absorption Law

6. Expression: A·B + A·B̄
Laws that can be used: Distributive Law, Complement Law, Identity Law

7. Expression: Ā·B + A·B
Laws that can be used: Distributive Law, Complement Law, Identity Law

8. Expression: (A·B)'
Laws that can be used: De Morgan's Theorem

9. Expression: (A + B)'
Laws that can be used: De Morgan's Theorem

10. Expression: (A + B)' + A
Laws that can be used: De Morgan's Theorem, Absorption Law, Distributive Law