Practical 6: To Simplify the Given Boolean Expression and Realize It Using Basic Gates and Universal Gates

Boolean algebra is used to simplify digital expressions so that a circuit becomes easier, faster, and uses fewer gates.
After simplification, the expression is implemented using:

1. **Basic Gates** (AND, OR, NOT, XOR)

2. **Universal Gates** (NAND and NOR) — because they can build any circuit.

In this practical, we simplify the given Boolean expression and design its logic circuit.

> **Note:** Since your teacher will give any expression like
> **F = A'B + AB'** or **F = A + BC**,
> main yahan ek universal format likh rahi hoon jisse tum *koi bhi* expression apply kar sakte ho.
> Ye pura practical EXACT FORMAT me hota hai.

Aim

To simplify the given Boolean expression using Boolean laws and realize it using
Basic gates and **Universal gates** (NAND/NOR).

Boolean Simplification (General Method)

Common Boolean laws used for simplification:

1. Identity Law

$A + 0 = A$
$A \cdot 1 = A$

2. Null Law

A + 1 = 1
A · 0 = 0

3. Idempotent Law

A + A = A
A · A = A

4. Complement Law

A + A’ = 1
A · A’ = 0

5. Absorption Law

A + AB = A
A(A + B) = A

6. Distributive Law

A(B + C) = AB + AC

7. De Morgan’s Law

(A · B)’ = A’ + B’
(A + B)’ = A’ · B’

Example Simplification (Standard for Practical Write-Up)

Let the expression be:

F = A’B + AB’

This is a very common expression used in labs.

Step-by-step Simplification:

F = A’B + AB’
This is in XOR form.

So, simplified form:

F = A ⊕ B

Ye simplest form hoti hai.

Realization Using Basic Gates

F = A ⊕ B

To realize XOR using basic gates:

XOR = A'B + AB'

So the circuit uses:

- 2 NOT gates (A', B')
- 2 AND gates
- 1 OR gate

This creates the basic XOR circuit.

Realization Using Universal Gates (NAND or NOR)

Universal gates can build any logic circuit.

XOR using only NAND Gates:

- 4 NANDs to create XOR
- No other gates needed
- Inputs are passed through NAND combinations to generate SUM output

XOR using NOR Gates:

- 5 NOR gates required

- Both input signals inverted using NOR
- Then combined to create XOR behavior

Universal implementations are used because:

- Cheaper
- Faster
- More reliable in IC design

Truth Table (General Table for Expression F)

A	**B**	**A’**	**B’**	**A’B**	**AB’**	**F = A ⊕ B**
0	0	1	1	0	0	0
0	1	1	0	1	0	1
1	0	0	1	0	1	1
1	1	0	0	0	0	0

You can modify this table depending on your given expression.