Exercise

A. Multiple-Choice Questions (MCQs)

Tick (✓) the correct answer.

1. **Which of the following Boolean expressions represents the OR operation?**
   (a) A · B (b) **A + B** (c) A (d) A ⊕ B
2. **What is the dual of the Boolean expression A · 0 = 0?**
   (a) **A + 1 = 1** (b) A + 0 = A (c) A · 1 = A (d) A · 0 = 0
3. **Which logic gate gives output "true" only when both inputs are true?**
   (a) OR gate (b) **AND gate** (c) XOR gate (d) NOT gate
4. **In a half-adder circuit, the carry is generated by which operation?**
   (a) XOR operation (b) **AND operation** (c) OR operation (d) NOT operation
5. **What is the decimal equivalent of binary number 1101?**
   (a) 11 (b) 12 (c) **13** (d) 14

B. Short Questions

1. **Define a Boolean function and give an example.**
   A Boolean function is an expression made using Boolean variables and logical operations like AND, OR, and NOT.
   Example: F = A + B
2. **What is the significance of the truth table in digital logic?**
   A truth table shows all possible input values and their corresponding output. It helps understand how a logic gate or circuit behaves.
3. **Explain the difference between analog and digital signals.**
   Analog signals are continuous and vary smoothly.
   Digital signals are in discrete steps, usually only 0 and 1.
4. **Describe the function of a NOT gate with its truth table.**
   A NOT gate inverts the input. If input is 1, output is 0 and vice versa.

Input	Output
0	1
1	0
5. **What is the purpose of a Karnaugh map in simplifying Boolean expressions?**
   A Karnaugh Map (K-Map) is used to simplify Boolean expressions easily by grouping adjacent 1s in a truth table format.

C. Long Questions (Detailed Answers)

1. **Explain the usage of Boolean functions in computers.**
   Boolean functions are used in computers for decision-making, controlling circuits, building logic gates, and simplifying complex logic in programming and hardware.
2. **Describe how to construct a truth table for a Boolean expression with an example.**
   List all possible inputs (like A, B), then calculate the output for each.
   Example for F = A + B:

A	B	F = A + B
0	0	0
0	1	1
1	0	1
1	1	1

3. **Describe the concept of duality in Boolean algebra and give an example.**
   Duality means replacing + with · and 0 with 1 (and vice versa) in a Boolean expression.
   Example: $A \cdot 0 = 0 \Rightarrow$ Dual: $A + 1 = 1$
4. **Compare half-adders and full-adders with truth tables and Boolean expressions.**

Half-Adder	Full-Adder
Adds 2 bits	Adds 3 bits (including carry)
Sum = $A \oplus B$	Sum = $A \oplus B \oplus Cin$
Carry = $A \cdot B$	Carry = $(A \cdot B) + (Cin \cdot (A \oplus B))$

Truth table (Half-Adder):

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

5. **How do Karnaugh maps simplify Boolean expressions? Explain with steps.**
   K-Maps simplify expressions by grouping 1s in adjacent cells. This removes redundant variables and gives the simplest form of the Boolean expression.
   Steps:

- Draw K-map with required variables.
- Fill it using the truth table.
- Group adjacent 1s in sizes of 1, 2, 4, or 8.
- Write the simplified expression from the groups.

6. **Design a 4-bit binary adder using half and full adders.**

- Use 1 half-adder for the first bit.
- Use 3 full-adders for the remaining bits.
- Each full-adder takes a carry input from the previous stage.
- Draw circuit with connections and write Boolean expressions for each stage.

7. **Simplify the Boolean function using Boolean algebra rules:**
   F(A, B) = A · B + A · B
   Solution:
   A · B + A · B = A (B + B) = A (1) = A
8. **Use De Morgan's laws to simplify the function:**
   F(A, B, C) = A + B + AC
   No negations here. But applying laws if needed:
   If the expression were ¬(A + B + AC), it becomes:
   ¬A · ¬B · ¬A + ¬C
9. **Simplify the following expressions:**

(a) **A + B · (A + B)**
Apply distributive:
A + A·B + B·B = A + A·B + B = A + B

(b) **(A + B) · (A + B)**
Same expression:
= A + B

(c) **A + A · (B + C)**
Apply Absorption Law:
A + A·X = A ⇒ A + A·(B + C) = A

(d) **A · B + A · B**
= A · (B + B) = A · 1 = A

(e) **(A · B) + (A · B)**
= A · (B + B) = A · 1 = A