Exercise walk-through

4.3
Binary Applications

GAC Mathematics

Sets, relations and functions (2)

- A **relation** pairs elements of two sets. A **function** is a **relation** in which each input has exactly one output. "Each student $\to$ their birthday" is a **function** (two students may share a birthday); "each birthday $\to$ the students born on it" is not, because one input can have several outputs.
- A **Venn diagram** , introduced by the English logician John Venn in 1880, turns a set problem into a picture. Check that the regions add up to the total.

$$N_{neither} = |U| - |F \cup G| = 30 - 26 = 4$$

Counting systems

- A **number base** b uses the digits 0 to $b-1$, and each place is worth a power of b. **Decimal** uses ten, **binary** two, **hexadecimal** sixteen (digits 0–9 then A–F).
- Every digit's value is its **place value**: in binary the places are 1, 2, 4, 8, 16 and so on.
- **Hexadecimal** is shorthand for binary: one hex digit is exactly four bits. Web colours use it: '#FF8000' means red 255, green 128, blue 0, a bright orange.

$$V = \sum d_i 2^i$$

Graphs and networks

- A **graph** is a set of **vertices** joined by **edges**. It models anything with connections: roads, friendships, dependencies.
- The **degree** of a vertex is the number of **edges** that meet it. Every edge has two ends, so the degrees always add up to twice the number of edges. This is the fact behind Euler's Königsberg proof.
- A **tree** is a connected **graph** with no cycles; a **tree** with n vertices has $n - 1$ edges. A family tree and a computer's folder structure are trees.

Exercises 2.1·2.2

2.1 For A true and B false, find A AND B and A OR B, using inclusive OR. [2]

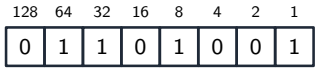
128	64	32	16	8	4	2	1
0	1	1	0	1	0	0	1

2.2 Invert the four-bit pattern 1010 using bitwise NOT, and find its unsigned decimal value. [2]

128	64	32	16	8	4	2	1
0	1	1	0	1	0	0	1

Exercises 2.1·2.2

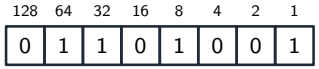
2.1 For A true and B false, find A AND B and A OR B, using inclusive OR. [2]



2.1

A AND B is false because both inputs are not true. A OR B is true because at least one input is true.

2.2 Invert the four-bit pattern 1010 using bitwise NOT, and find its unsigned decimal value. [2]

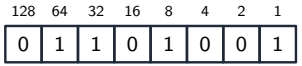


2.2

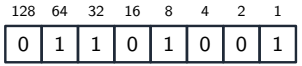
The inverted four-bit pattern is 0101. Unsigned value is 5. The width remains four bits.

Exercises 2.3·2.4

2.3 List the outputs of $A \wedge \neg B$ in row order (0,0), (0,1), (1,0), (1,1). State which input condition makes the output true. [3]

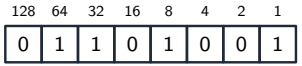


2.4 Use De Morgan to rewrite $\neg(A \wedge B)$. Evaluate both forms at A true and B true. [3]



Exercises 2.3·2.4

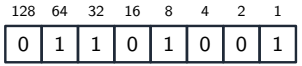
2.3 List the outputs of $A \wedge \neg B$ in row order (0,0), (0,1), (1,0), (1,1). State which input condition makes the output true. [3]



2.3

In order the outputs are 0, 0, 1, 0. Applying NOT before AND earns M1; correct output column earns A1. Output is true exactly when A is true and B false.

2.4 Use De Morgan to rewrite $\neg(A \wedge B)$. Evaluate both forms at A true and B true. [3]

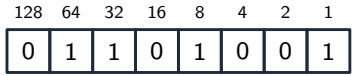


2.4

Equivalent form $(\neg A) \vee (\neg B)$. Original form is NOT true, hence false. Rewritten form is false OR false, also false. A single row check alone is not a proof of full equivalence.

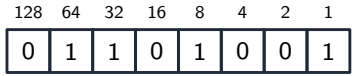
Exercise 3.1

3.1 For four-bit patterns X = 1100 and Y = 1010, calculate (a) bitwise AND, (b) bitwise inclusive OR, and (c) bitwise NOT of X. Give each result also as an unsigned decimal value. [6]



Exercise 3.1

3.1 For four-bit patterns X = 1100 and Y = 1010, calculate (a) bitwise AND, (b) bitwise inclusive OR, and (c) bitwise NOT of X. Give each result also as an unsigned decimal value. [6]



3.1

AND gives 1000. Decimal 8. OR gives 1110. Decimal 14. Four-bit NOT of X gives 0011. Decimal 3. NOT does not create an unrestricted or signed value here.

Exercise 3.2

3.2 A simplified access rule is $Y = (A \wedge B) \vee (A \wedge \neg B)$, where A and B are Boolean inputs. (a) Construct its complete truth table with intermediate columns. (b) Simplify the rule and justify equivalence. (c) Explain whether this mathematical simplification establishes that a real access system is secure. [8]

128	64	32	16	8	4	2	1
0	1	1	0	1	0	0	1

Exercise 3.2

3.2 A simplified access rule is $Y = (A \wedge B) \vee (A \wedge \neg B)$, where A and B are Boolean inputs. (a) Construct its complete truth table with intermediate columns. (b) Simplify the rule and justify equivalence. (c) Explain whether this mathematical simplification establishes that a real access system is secure. [8]

128	64	32	16	8	4	2	1
0	1	1	0	1	0	0	1

3.2

(a) Use row order 00, 01, 10, 11. Column NOT B is 1, 0, 1, 0. Column A AND B is 0, 0, 0, 1. Column A AND NOT B is 0, 0, 1, 0. Final Y is 0, 0, 1, 1.

(b) Factor A using distributivity.

$$Y = A \wedge (B \vee \neg B)$$

The bracket is always true, so $Y = A$. This agrees with the A column in every truth-table row, proving Boolean equivalence.

(c) The truth table tests this abstract rule only; actual authentication, implementation and policy are not established by its simplification.