CodeBash

codebash.co.uk

FREE SAMPLE · SHARE FREELY

Boolean Algebra

Worked Solutions

Boolean laws, simplification and proofs.

GCSE & A-Level Computer Science

This booklet is yours to keep, print and share.

Find more revision worksheets, an online code editor, lockdown assessments and full theory revision at **codebash.co.uk**.

GCSE/A-Level CS

CodeBash

Boolean Algebra

codebash.co.uk

150 practice questions - laws, truth tables, De Morgan's, simplification, proofs

SOLUTIONS

GCSE/A-Level CS

CodeBash

Boolean Algebra

codebash.co.uk

SOLUTIONS

SOLUTIONS

Section A: Apply the Law (20 marks)

Simplify each expression and state the Boolean law used.

Q	Expression	Simplified to	Law Used
1	`A AND 1`	`A`	Identity Law
2	`A OR 0`	`A`	Identity Law
3	`A AND 0`	`0`	Null Law
4	`A OR 1`	`1`	Null Law
5	`A AND A`	`A`	Idempotent Law
6	`A OR A`	`A`	Idempotent Law
7	`A AND NOT(A)`	`0`	Complement Law
8	`A OR NOT(A)`	`1`	Complement Law
9	`NOT(NOT(A))`	`A`	Double Negation
10	`A AND B`	`B AND A`	Commutative Law
11	`A OR (B OR C)`	`(A OR B) OR C`	Associative Law
12	`A AND (A OR B)`	`A`	Absorption Law
13	`A OR (A AND B)`	`A`	Absorption Law
14	`NOT(A AND B)`	`NOT(A) OR NOT(B)`	De Morgan's Law
15	`NOT(A OR B)`	`NOT(A) AND NOT(B)`	De Morgan's Law
16	`A AND (B OR C)`	`(A AND B) OR (A AND C)`	Distributive Law
17	`(A OR B)(A OR C)`	`A OR (B AND C)`	Distributive Law
18	`A AND 1 AND A`	`A`	Identity + Idempotent
19	`NOT(NOT(A) OR NOT(B))`	`A AND B`	De Morgan's + Double Negation
20	`(A AND B) OR (A AND NOT(B))`	`A`	Complement + Identity

Boolean Algebra

Section B: Truth Table Completion (20 marks)

Complete the output column for each Boolean expression.

Method - Truth Tables:
List every input combination (use binary counting order).
Evaluate the expression for each row, recording 0 or 1 in the output column.
For 2 vars: 4 rows. For 3 vars: 8 rows. For 4 vars: 16 rows.

21. `A XOR B`

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

22. `NOT(A) XOR B`

A	B	NOT(A) XOR B
0	0	1
0	1	0
1	0	0
1	1	1

23. `NOT(A XOR B)`

A	B	NOT(A XOR B)
0	0	1
0	1	0
1	0	0
1	1	1

24. `A XNOR NOT(B)`

A	B	A XNOR NOT(B)
0	0	0
0	1	1
1	0	1
1	1	0

25. `(A AND B) OR C`

A	B	C	(A AND B) OR C
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

26. `A AND (B OR C)`

A	B	C	A AND (B OR C)
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

27. `NOT(A) AND B OR C`

A	B	C	NOT(A) AND B OR C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

28. `NOT(A OR B) OR C`

A	B	C	NOT(A OR B) OR C
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1

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

29. `A XOR (B AND C)`

A	B	C	A XOR (B AND C)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

30. `NOT(A) AND NOT(B) OR C`

A	B	C	NOT(A) AND NOT(B) OR C
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

31. `(A OR B) AND (B OR C)`

A	B	C	(A OR B) AND (B OR C)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

32. `A NAND (B OR C)`

A	B	C	A NAND (B OR C)
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

33. `(A AND B) OR (C AND D)`

A	B	C	D	(A AND B) OR (C AND D
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

34. `NOT(A OR B) AND (C OR D)`

A	B	C	D	NOT(A OR B) AND (C OR
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0

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

35. `(A XOR B) AND (C XOR D)`

A	B	C	D	(A XOR B) AND (C XOR D)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

36. `A AND B AND C AND D`

A	B	C	D	A AND B AND C AND D
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0

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

37. `NOT(A AND B) OR (C AND D)`

A	B	C	D	NOT(A AND B) OR (C AND
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

38. `(A OR C) AND (B OR D)`

A	B	C	D	(A OR C) AND (B OR D)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1

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

39. `NOT(A) AND B AND NOT(C) AND D`

A	B	C	D	NOT(A) AND B AND NOT(C) A
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

40. `A XOR B XOR C XOR D`

A	B	C	D	A XOR B XOR C XOR D
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0

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

Boolean Algebra

SOLUTIONS

Section C: De Morgan's Law (20 marks)

Apply De Morgan's Law to rewrite each expression in its equivalent form.

Method - De Morgan's Laws:
NOT(A AND B) = NOT(A) OR NOT(B) (change AND to OR, negate each term)
NOT(A OR B) = NOT(A) AND NOT(B) (change OR to AND, negate each term)
Apply repeatedly for nested expressions. Double negation cancels: NOT(NOT(X))=X

Q	Expression	Equivalent Form
41	NOT(A AND B)	NOT(A) OR NOT(B)
42	NOT(A OR B)	NOT(A) AND NOT(B)
43	NOT(A AND B AND C)	NOT(A) OR NOT(B) OR NOT(C)
44	NOT(A OR B OR C)	NOT(A) AND NOT(B) AND NOT(C)
45	NOT(A) AND NOT(B)	NOT(A OR B)
46	NOT(A) OR NOT(B)	NOT(A AND B)
47	NOT((A AND B) OR C)	(NOT(A) OR NOT(B)) AND NOT(C)
48	NOT(A AND (B OR C))	NOT(A) OR (NOT(B) AND NOT(C))
49	NOT((A OR B) AND (C OR D))	(NOT(A) AND NOT(B)) OR (NOT(C) AND NOT(D))
50	NOT(NOT(A) OR NOT(B))	A AND B
51	NOT(NOT(A) AND NOT(B))	A OR B
52	NOT(NOT(A) AND NOT(B) AND NOT(C))	A OR B OR C
53	NOT(A OR B) AND NOT(C OR D)	NOT(A) AND NOT(B) AND NOT(C) AND NOT(D)
54	NOT((A XOR B) AND C)	NOT(A XOR B) OR NOT(C)
55	NOT(A NAND B)	A AND B
56	NOT(A NOR B)	A OR B
57	NOT(A OR B OR C OR D)	NOT(A) AND NOT(B) AND NOT(C) AND NOT(D)
58	NOT((A AND B) AND (C AND D))	NOT(A AND B) OR NOT(C AND D)
59	NOT(A) AND NOT(B) AND NOT(C)	NOT(A OR B OR C)
60	NOT(A) AND NOT(B) AND NOT(C) AND NOT(D)	NOT(A OR B OR C OR D)

Boolean Algebra

Section D: Full Simplification (30 marks)

Fully simplify each Boolean expression. Show each step and state the law used.

Method - Simplification Strategy:
Identify patterns: double negation, complement pairs (A AND NOT(A)=0), idempotent (A AND A=A).
Try factoring before distributing. Look for absorption: A AND (A OR B) = A.
Apply De Morgan's to move NOT inside brackets.

Q	Expression	Result	Step-by-Step Working
61	`A AND A AND B`	`A AND B`	*1. A AND A = A (Idempotent Law)* *2. Result: A AND B*
62	`A OR A OR B`	`A OR B`	*1. A OR A = A (Idempotent Law)* *2. Result: A OR B*
63	`A AND (A OR B)`	`A`	*1. Absorption Law: A AND (A OR B) = A directly*
64	`A OR (A AND B)`	`A`	*1. Absorption Law: A OR (A AND B) = A directly*
65	`(A AND B) OR (A AND NOT(B))`	`A`	*1. Factor: A AND (B OR NOT(B))* *2. B OR NOT(B) = 1 (Complement Law)* *3. A AND 1 = A (Identity Law)*
66	`A AND 1 OR A AND 0`	`A`	*1. A AND 1 = A (Identity Law)* *2. A AND 0 = 0 (Null Law)* *3. A OR 0 = A (Identity Law)*
67	`NOT(NOT(A AND B))`	`A AND B`	*1. Double Negation: NOT(NOT(X)) = X where X = A AND B* *2. Result: A AND B*
68	`(A OR B) AND (A OR NOT(B))`	`A`	*1. Distributive: A OR (B AND NOT(B))* *2. B AND NOT(B) = 0 (Complement Law)* *3. A OR 0 = A (Identity Law)*
69	`A AND B AND 1 AND A`	`A AND B`	*1. A AND 1 = A (Identity Law)* *2. A AND A = A (Idempotent Law)* *3. Result: A AND B*
70	`(A AND NOT(A)) OR B`	`B`	*1. A AND NOT(A) = 0 (Complement Law)* *2. 0 OR B = B (Identity Law)*
71	`NOT(A) AND (A OR B)`	`NOT(A) AND B`	*1. Distribute: (NOT(A) AND A) OR (NOT(A) AND B)* *2. NOT(A) AND A = 0 (Complement Law)* *3. 0 OR (NOT(A) AND B) = NOT(A) AND B*
72	`A OR (NOT(A) AND B)`	`A OR B`	*1. Distribute OR: (A OR NOT(A)) AND (A OR B)* *2. A OR NOT(A) = 1 (Complement Law)* *3. 1 AND (A OR B) = A OR B*

73	(A OR B) AND (NOT(A) OR B)	B	*1. Distribute: B OR (A AND NOT(A))* *2. A AND NOT(A) = 0* *3. B OR 0 = B*
74	NOT(NOT(A) OR NOT(B))	A AND B	*1. De Morgan's on NOT(A) OR NOT(B): NOT(NOT(A) OR NOT(B)) = NOT(NOT* *2. Double Negation: A AND B*
75	(A AND B) OR (NOT(A) AND B)	B	*1. Factor B: B AND (A OR NOT(A))* *2. A OR NOT(A) = 1* *3. B AND 1 = B*
76	NOT(A AND B) AND B	NOT(A) AND B	*1. NOT(A AND B) = NOT(A) OR NOT(B) (De Morgan's)* *2. (NOT(A) OR NOT(B)) AND B* *3. Distribute: (NOT(A) AND B) OR (NOT(B) AND B)* *4. NOT(B) AND B = 0 -> NOT(A) AND B*
77	A AND (NOT(A) OR B)	A AND B	*1. Distribute: (A AND NOT(A)) OR (A AND B)* *2. A AND NOT(A) = 0* *3. 0 OR (A AND B) = A AND B*
78	(A AND B) OR (A AND B AND C)	A AND B	*1. Factor: A AND B AND (1 OR C)* *2. 1 OR C = 1 (Null Law)* *3. A AND B AND 1 = A AND B*
79	NOT(A OR B) OR (NOT(A) AND B)	NOT(A)	*1. NOT(A OR B) = NOT(A) AND NOT(B) (De Morgan's)* *2. NOT(A) AND NOT(B) OR NOT(A) AND B* *3. Factor NOT(A): NOT(A) AND (NOT(B) OR B)* *4. NOT(B) OR B = 1 -> NOT(A)*
80	A AND B OR NOT(A) AND B	B	*1. Factor B: B AND (A OR NOT(A))* *2. A OR NOT(A) = 1 -> B*
81	(A OR NOT(A)) AND B	B	*1. A OR NOT(A) = 1 (Complement Law)* *2. 1 AND B = B (Identity Law)*
82	NOT(A) OR A	1	*1. Complement Law: A OR NOT(A) = 1 directly*
83	A AND B AND NOT(A OR B)	0	*1. NOT(A OR B) = NOT(A) AND NOT(B) (De Morgan's)* *2. A AND B AND NOT(A) AND NOT(B)* *3. A AND NOT(A) = 0 -> result is 0*
84	(A AND B) OR NOT(A AND B)	1	*1. X OR NOT(X) = 1 (Complement Law) where X = A AND B* *2. Result: 1*
85	NOT(A) AND NOT(B) AND (A OR B)	0	*1. NOT(A) AND NOT(B) = NOT(A OR B) (De Morgan's)* *2. NOT(A OR B) AND (A OR B) = 0 (Complement Law)*
86	(A OR B) AND NOT(A) AND NOT(B)	0	*1. NOT(A) AND NOT(B) = NOT(A OR B) (De Morgan's)* *2. (A OR B) AND NOT(A OR B) = 0 (Complement Law)*
87	A AND B OR NOT(A) AND NOT(B)	NOT(A XOR B) (XNOR)	*1. This is the definition of XNOR: output 1 when A=B* *2. Also written as NOT(A XOR B)*
88	NOT(A) AND (A OR B OR C)	NOT(A) AND (B OR C)	*1. Distribute: (NOT(A) AND A) OR (NOT(A) AND B) OR (NOT(A) AND C)* *2. NOT(A) AND A = 0 (Complement Law)* *3. 0 OR NOT(A) AND B OR NOT(A) AND C = NOT(A) AND (B OR C)*
89	NOT(A) AND B OR A AND NOT(B)	A XOR B	*1. NOT(A) AND B: output 1 when A=0, B=1* *2. A AND NOT(B): output 1 when A=1, B=0* *3. Together: output 1 when A differs from B = A XOR B*

90	`NOT(A XOR B)`	`A XNOR B = (A AND B) OR (NOT(A) AND NOT(B))`	*1. XOR is true when A != B, NOT(XOR) is true when A = B* *2. Expanded: (A AND B) OR (NOT(A) AND NOT(B))*

GCSE/A-Level CS

CodeBash

Boolean Algebra

codebash.co.uk

SOLUTIONS

SOLUTIONS

Section E: Prove the Equivalence (20 marks)

Show step-by-step that each pair of expressions is equivalent.

Method - Proof Strategy:
Start from one side (usually the more complex) and transform it step by step.
State the law applied at each step.
End with the expression on the other side of the equation.

Q	Statement to Prove	Proof / Your Working
91	Show A AND (B OR C) = (A AND B) OR (A AND C)	*Distributive Law applied directly. LHS = RHS. QED*
92	Show NOT(NOT(A)) = A	*Double Negation Law: negating twice restores the original. QED*
93	Show (A AND B) OR A = A	*Reorder: A OR (A AND B) = A by Absorption Law. QED*
94	Show A AND (A OR B) = A	*Absorption Law applied directly. QED*
95	Show NOT(A) AND (A OR B) = NOT(A) AND B	*Distribute: (NOT(A) AND A) OR (NOT(A) AND B) -> 0 OR NOT(A)B -> NOT(A) AND B*
96	Show (A OR B) AND (A OR NOT(B)) = A	*Distribute: A OR (B AND NOT(B)) = A OR 0 = A*
97	Show NOT(A AND B) = NOT(A) OR NOT(B)	*De Morgan's First Law - this is the definition. QED*
98	Show NOT(A OR B) = NOT(A) AND NOT(B)	*De Morgan's Second Law - this is the definition. QED*
99	Show A OR (NOT(A) AND B) = A OR B	*(A OR NOT(A)) AND (A OR B) = 1 AND (A OR B) = A OR B [Distributive + Complement]*
100	Show (A XOR B) = (A OR B) AND NOT(A AND B)	*A XOR B = (A AND NOT(B)) OR (NOT(A) AND B) = same as (A OR B) AND NOT(A AND B). QED*
101	Show NOT(A) AND NOT(B) = NOT(A OR B)	*De Morgan's: NOT(A OR B) = NOT(A) AND NOT(B). QED*
102	Show (A AND B) OR (A AND B AND C) = A AND B	*Factor: A AND B AND (1 OR C) = A AND B AND 1 = A AND B*

103	Show A OR (A AND NOT(B)) = A	*Factor: A AND (1 OR NOT(B)) = A AND 1 = A*
104	Show (A AND NOT(B)) OR (NOT(A) AND B) OR (A AND B) = A OR B	*First two = A XOR B. Adding A AND B: XOR OR AND = OR*
105	Show NOT(A) AND (A OR B) AND (A OR C) = NOT(A) AND B AND C	*NOT(A) AND (A OR B) = NOT(A) AND B. Then AND (A OR C): NOT(A) AND B AND C*
106	Show A AND B AND C = NOT(NOT(A) OR NOT(B) OR NOT(C))	*De Morgan's: NOT(NOT(A) OR NOT(B) OR NOT(C)) = A AND B AND C*
107	Show (A OR B) AND (NOT(A) OR C) AND (B OR C) = (A OR B) AND (NOT(A) OR C)	*Consensus Theorem: the third factor is redundant.*
108	Show NOT(A AND B) AND A = NOT(B) AND A	*(NOT(A) OR NOT(B)) AND A = (NOT(B) AND A) since NOT(A) AND A = 0*
109	Show A NAND A = NOT(A)	*A NAND A = NOT(A AND A) = NOT(A) by Idempotent Law*
110	Show A NOR A = NOT(A)	*A NOR A = NOT(A OR A) = NOT(A) by Idempotent Law*

GCSE/A-Level CS

CodeBash

Boolean Algebra

codebash.co.uk

SOLUTIONS

SOLUTIONS

Section F: Derive the Expression (20 marks)

Write a Boolean expression that satisfies each description.

Method - Derive Expression:
Identify which inputs must be 1, which must be 0.
AND for 'all must be true'. OR for 'any must be true'.
NOT to invert an input. XOR for 'exactly one different'.

Q	Description	Boolean Expression
111	Output is 1 only when A=1 AND B=1	`A AND B`
112	Output is 1 when A=0 AND B=0	`NOT(A) AND NOT(B)`
113	Output is 0 only when A=0 AND B=0	`A OR B`
114	Output is 0 only when A=1 AND B=1	`NOT(A AND B) = NAND`
115	Output is 1 when A and B are different	`A XOR B`
116	Output is 1 when A and B are the same	`NOT(A XOR B) = XNOR`
117	Output is 1 when at least one of A,B,C is 1	`A OR B OR C`
118	Output is 1 only when all of A,B,C are 1	`A AND B AND C`
119	Output is 1 when exactly one of A,B is 1	`A XOR B`
120	Output is 1 when A=1 and B=0	`A AND NOT(B)`
121	Output is 1 when the number of 1s in A,B,C is odd	`A XOR B XOR C`

122	Output is 0 when any input is 1	`NOT(A) AND NOT(B) AND NOT(C) = NOR(A,B,C)`
123	Output is 1 unless all inputs are 1	`NOT(A AND B AND C) = NAND(A,B,C)`
124	Output is 1 when A=1 OR when B=1 AND C=1	`A OR (B AND C)`
125	Output is 1 when A=0 AND B=0 AND C=0	`NOT(A) AND NOT(B) AND NOT(C)`
126	Output is 1 when A differs from B, and C=1	`(A XOR B) AND C`
127	Output equals A when B=1; 0 when B=0	`A AND B`
128	Output is 1 when A=B AND C=1	`NOT(A XOR B) AND C`
129	Output is NOT(A) when B=1; 1 when B=0	`NOT(A) OR NOT(B)`
130	Output is 1 when at least two of A,B,C are 1	`(A AND B) OR (A AND C) OR (B AND C)`

GCSE/A-Level CS

CodeBash

Boolean Algebra

codebash.co.uk

SOLUTIONS

SOLUTIONS

Section G: Mixed Challenge (20 marks)

Fully simplify each expression. Multiple laws may be required.

Method - Mixed Challenge Tips:
Look for De Morgan's opportunities first (NOT over AND/OR).
Use XOR identities: A XOR 0=A, A XOR 1=NOT(A), A XOR A=0.
Factor common sub-expressions then apply absorption or complement.

Q	Expression	Result	Step-by-Step Working
131	`NOT(A OR B) OR (A AND NOT(B))`	`NOT(B)`	*1. NOT(A OR B) = NOT(A) AND NOT(B) (De Morgan's)* *2. NOT(A) AND NOT(B) OR A AND NOT(B) -> NOT(B) AND (NOT(A) OR A)* *3. NOT(A) OR A = 1 -> NOT(B)*
132	`(A OR B) AND (NOT(A) OR NOT(B))`	`A XOR B`	*1. NOT(A) OR NOT(B) = NOT(A AND B) (De Morgan's)* *2. (A OR B) AND NOT(A AND B) = XOR by definition*
133	`A AND B OR NOT(A) AND NOT(B)`	`NOT(A XOR B) (XNOR)`	*1. Output 1 when A=B (both 0 or both 1) - this is XNOR by definition*
134	`NOT(NOT(A) AND B) AND NOT(A AND NOT(B))`	`NOT(A XOR B)`	*1. NOT(NOT(A) AND B) = A OR NOT(B) (De Morgan's)* *2. NOT(A AND NOT(B)) = NOT(A) OR B (De Morgan's)* *3. (A OR NOT(B)) AND (NOT(A) OR B) = NOT(A XOR B)*
135	`(A AND B AND C) OR (NOT(A) AND B AND C) OR NOT(B AND C)`	`1`	*1. First two terms: B AND C AND (A OR NOT(A)) = B AND C* *2. B AND C OR NOT(B AND C) = 1 (Complement Law)*
136	`NOT(A NAND B)`	`A AND B`	*1. NOT(A NAND B) = NOT(NOT(A AND B)) = A AND B (Double Negation)*
137	`NOT(A NOR B)`	`A OR B`	*1. NOT(A NOR B) = NOT(NOT(A OR B)) = A OR B (Double Negation)*
138	`A XOR 0`	`A`	*1. XOR with 0: if A=0, 0 XOR 0=0; if A=1, 1 XOR 0=1 -> output = A*
139	`A XOR 1`	`NOT(A)`	*1. XOR with 1: if A=0, 0 XOR 1=1; if A=1, 1 XOR 1=0 -> output = NOT(A)*
140	`A XOR A`	`0`	*1. Same input XORed: always 0 (Complement-like identity for XOR)*
141	`A XNOR A`	`1`	*1. Same input XNORed: always 1 (if A=A they are always equal)*

142	(A AND B) XOR (A AND B)	0	*1. X XOR X = 0 where X = A AND B*
143	A AND NOT(A XOR B)	A AND B	*1. NOT(A XOR B) = XNOR = 1 when A=B* *2. A AND XNOR: only 1 when A=1 AND A=B -> A AND B*
144	NOT(A AND NOT(B)) AND NOT(B)	NOT(A) AND NOT(B)	*1. NOT(A AND NOT(B)) = NOT(A) OR B (De Morgan's)* *2. (NOT(A) OR B) AND NOT(B) = (NOT(A) AND NOT(B)) OR (B AND NOT(B))* *3. B AND NOT(B) = 0 -> NOT(A) AND NOT(B)*
145	(A AND B) OR (NOT(A) AND B) OR (A AND NOT(B))	A OR B	*1. (A AND B) OR (NOT(A) AND B) = B (factor B)* *2. B OR (A AND NOT(B)) = B OR A (absorption variant) = A OR B*
146	NOT(A) OR (A AND B)	NOT(A) OR B	*1. NOT(A) OR (A AND B) = (NOT(A) OR A) AND (NOT(A) OR B) = 1 AND (NOT(A) OR B)*
147	(A OR B) AND NOT(NOT(A) AND NOT(B))	A OR B	*1. NOT(NOT(A) AND NOT(B)) = NOT(NOT(A OR B)) = A OR B (De Morgan's + Double Negation)* *2. (A OR B) AND (A OR B) = A OR B (Idempotent)*
148	NOT(A AND B) AND NOT(NOT(A) AND NOT(B))	A XOR B	*1. NOT(A AND B) = NAND. NOT(NOT(A) AND NOT(B)) = A OR B (De Morgan's x2)* *2. NAND AND (A OR B) = NOT(A AND B) AND (A OR B) = XOR definition*
149	A XOR (A AND B)	A AND NOT(B)	*1. XOR expansion: (A AND NOT(A AND B)) OR (NOT(A) AND A AND B)* *2. NOT(A) AND A AND B = 0 -> A AND NOT(A AND B)* *3. NOT(A AND B) applied: A AND (NOT(A) OR NOT(B)) = A AND NOT(B)*
150	(A OR B OR C) AND NOT(A OR B)	NOT(A) AND NOT(B) AND C	*1. NOT(A OR B) = NOT(A) AND NOT(B) (De Morgan's)* *2. NOT(A) AND NOT(B) AND (A OR B OR C) -> distribute; A,B terms cancel* *3. Result: NOT(A) AND NOT(B) AND C*

Want more like this?

CodeBash has the rest of the GCSE and A-Level CS toolkit.