

Cambridge IGCSE · Computer Science 0478 · Notes

Boolean logic

Read online at pastopic.com/cambridge/computer-science-0478/notes/boolean-logic

Last checked 30 September 2026

© 2026 Pastopic. Free for personal study. Please share the link, not the file.

What you need to know

By the end of this topic you should be able to:

1. **Recognise and draw the six gate symbols** the syllabus uses: NOT, AND, OR, NAND, NOR and XOR (XOR is sometimes written EOR). Draw them the way the syllabus shows them, because an unclear shape can't earn the mark.
2. **Say what each gate does** and write its truth table, which lists what comes out for each way the inputs can be set. Five of the gates have two inputs; NOT is the odd one out with just one.
3. **Build a logic circuit** from any of three starting points: a written description of a situation, a logic expression, or a truth table.
4. **Fill in a truth table** from any of three starting points: a written description, a logic expression, or a drawn circuit.
5. **Write a logic expression** from any of three starting points: a written description, a drawn circuit, or a truth table.

No exam circuit uses more than three input letters, and each ends in a single output line, so a full truth table never needs more than 8 rows. When you draw the circuit for a printed expression, keep every gate it names, even if a shorter circuit would behave the same.

Understand it

Ones and zeros as yes and no

Inside a processor every signal is either high or low voltage, which we write as 1 or 0. It helps to read 1 as "yes" (true, on) and 0 as "no" (false, off). A **logic gate** is a tiny electronic switch arrangement that looks at one or two of these bits and produces one new bit. Chains of gates, called **logic circuits**, are how a computer adds numbers, compares values and makes decisions.

The three building blocks

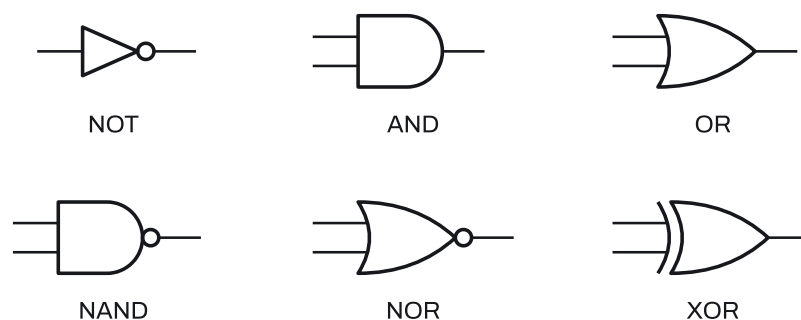
- **NOT** has one input and simply flips it: a 1 comes out as 0 and a 0 comes out as 1. It is also called an inverter.

- **AND** gives 1 only if the first input is 1 *and* the second input is 1. Think of a drawbridge that lowers only when the guard has pressed her button and the bridge-keeper has pressed his. One "no" is enough to stop it.
- **OR** gives 1 if at least one input is 1, including when both are. Think of a doorbell wired to a front button and a back button: either press rings it, and pressing both still just rings it.

Three more gates made from those

- **NAND** means "NOT AND": work out AND, then flip it. So NAND is 0 in exactly one case, when both inputs are 1, and 1 otherwise.
- **NOR** means "NOT OR": work out OR, then flip it. So NOR is 1 in exactly one case, when neither input is 1.
- **XOR** ("exclusive OR") gives 1 when the inputs disagree: one is 1 and the other is 0. If the two inputs match, whether both 0 or both 1, it gives 0. It is OR with the "both" case taken away.

On a symbol, a small circle on the output always means "then flip the answer". That is the only difference between AND and NAND, and between OR and NOR.



Here are all six at once. For the two-input gates, read each column downwards against the inputs on the left.

First input	Second input	AND	OR	NAND	NOR	XOR
0	0	0	0	1	1	0
0	1	0	1	1	0	1
1	0	0	1	1	0	1
1	1	1	1	0	0	0

NOT: input 0 gives 1, input 1 gives 0.

A useful way to remember the columns is to count the 1s: AND and NOR each have a single 1 (at opposite ends), OR and NAND each have three 1s, and XOR has two, in the middle.

Truth tables

A **truth table** lists every possible set of inputs, one per row, next to the output each set produces. With two inputs there are $2 \times 2 = 4$ rows; with three there are $2 \times 2 \times 2 = 8$. Write the input rows as the binary numbers 0 to 7 counting upwards: the first input column is four 0s then four 1s, the second goes in pairs (0 0 1 1 0 0 1 1), and the third alternates (0 1 0 1 0 1 0 1). Exam tables are always printed in this order.

For anything with more than one gate, add **working columns**: one column for the output of each gate, filled in from the inputs towards the final output. The exam gives you a "working space" for exactly this.

Logic expressions

A **logic expression** writes a circuit in words, using the gate names as operators, for example $Y = (D \text{ OR } E) \text{ AND NOT } F$. Three rules make it unambiguous:

- Brackets are worked out first, innermost first.
- NOT applies only to the single letter or bracket written straight after it. NOT $F \text{ AND } G$ flips only F and then ANDs the result with G , but NOT $(F \text{ AND } G)$ does the AND first and flips its answer. These have different truth tables.
- Where brackets group two gates, keep them exactly as printed. The exam sets expressions with brackets around every pair, so you never have to guess the order.

To find one row's output, substitute the 0s and 1s and work outwards. For $(D \text{ OR } E) \text{ AND NOT } F$ with $D = 0$, $E = 1$, $F = 1$: the bracket is $0 \text{ OR } 1 = 1$, NOT F is 0, and $1 \text{ AND } 0 = 0$.

Logic circuits

A circuit is the same thing drawn as gates joined by wires, with inputs on the left and the single output on the right.

- **Every operator in the expression is one gate**, including every NOT. Count them before you start: that number is the number of gates you should draw, and each one usually earns a mark.
- **Each gate other than NOT has exactly two inputs**. So "D AND E AND F" needs two AND gates: one for D AND E, and a second that combines that result with F.
- **One input can feed several gates**. Where an input wire splits, mark the split with a dot. Where two wires cross without joining, make that obvious, for example with a little hump in one wire.
- **Build from the inside of the brackets outwards**, so the gate for the outermost operator sits at the far right, producing the output.

From a description of a real situation

In a "problem statement" question you get a table that says what 1 and 0 mean for each input, and a sentence saying when the output is 1.

1. Read what **1** stands for on each input. Very often 1 is the helpful condition, but sometimes the table sets 1 as the *unwanted* one (for example "1 = sensor reports a fault"). When the rule needs the opposite of what 1 means, that input needs a NOT.
2. Turn the joining words into gates: "and", "both" and "as well as" are AND; "or" and "either ... or both" are OR; "not", "unless", "is off" and "is absent" usually mean NOT; "exactly one of" is XOR.
3. Put brackets where the sentence groups ideas. "U, or else both V and W" is $U \text{ OR } (V \text{ AND } W)$, which is different from $(U \text{ OR } V) \text{ AND } W$.
4. Check your expression on two or three rows against the sentence before completing the table.

From a truth table back to an expression

This works for any truth table.

1. Find each row where the output is 1.
2. For each such row, write one AND chain that is true for that row only: use the letter where the input is 1 and NOT with the letter where it is 0.
3. Join the chains with OR.

For example, if the output is 1 only in the rows $D, E, F = 0, 1, 1$ and $1, 1, 0$, the expression is $(\text{NOT } D \text{ AND } E \text{ AND } F) \text{ OR } (D \text{ AND } E \text{ AND NOT } F)$. Any expression with the same truth table is correct, so a shorter equivalent is fine here; drawing the circuit then follows the rules above.

Key facts and formulas

No formula list is given in this exam, so everything below is **Learn it**.

Formula	Status
NOT: output is the opposite of the single input	Learn it
AND: 0 unless every input is 1	Learn it
OR: 0 only for the input pair 0, 0	Learn it
NAND = NOT (AND): 0 for the input pair 1, 1 and 1 for the other three	Learn it
NOR = NOT (OR): 0 unless neither input is 1	Learn it
XOR: 1 when exactly one input is 1	Learn it
The six symbols, as drawn in the syllabus; a circle on the output means "invert"	Learn it
n inputs give 2^n rows: 2 inputs $\rightarrow$ 4 rows, 3 inputs $\rightarrow$ 8 rows	Learn it
Row order for three inputs: 000, 001, 010, 011, 100, 101, 110, 111	Learn it
One gate per operator (each NOT is a gate); non-NOT gates have exactly two inputs	Learn it

How to do it

In the questions we have tagged for this topic (38 questions, 2021–2025), the types came up this often. 28 of the 38 questions mix two or more types (typically "draw the circuit, then complete its table"), so the counts add up to more than 38.

Question type	Questions
Complete a truth table from an expression or a circuit	24
Draw a circuit from a logic expression	21
Name gates, match symbols, write single-gate truth tables	16
Turn a problem statement into an expression, truth table or circuit	6
Write an expression from a drawn circuit	2

1. Complete a truth table from an expression or a circuit (24 questions)

1. **Split the job into gates.** From an expression, each bracket and each NOT is a stage; from a circuit, each gate is a stage.
2. **Make a working column for each stage**, in the order you would calculate them, using the printed working space.
3. **Fill each column top to bottom**, using only the input columns or earlier working columns and the gate's rule.
4. **Last column = output.** Copy it into the answer column and look for a sanity check: a final NAND should give mostly 1s, a final NOR or AND mostly 0s.

Variations you will meet:

- **The circuit is drawn, not written.** Write the expression beside the diagram first (see type 5), then work from that. It is easier to check than tracing wires eight times.
- **"There are four errors in the output column; identify the rows."** Work out the full correct column yourself, compare row by row and give the row numbers that differ. Don't try to spot the errors by eye.
- **Conditions written with "= 1" and "= 0"**, such as "Y is 1 if (D = 1 AND F = 0) ...", are an expression in disguise: "F = 0" (or "F = NOT 1") is simply NOT F.

2. Draw a circuit from a logic expression (21 questions)

1. **Count the operators** (including every NOT). That is the number of gates.
2. **Draw the input letters** down the left side, spaced out, each with a line going right.
3. **Draw the innermost pieces first:** each NOT on the letter it belongs to, then the gate for each bracket, fed by the right two wires.
4. **Join the bracket outputs** into the gate for the operator between the brackets, and label its output with the output letter.

5. **Check it:** every gate has the right number of inputs, no gate is left unconnected, and NOT sits exactly where the expression puts it (before a gate or after it).

Variations you will meet:

- **A long nested statement** such as $((F \text{ NOR } G) \text{ AND } H) \text{ OR NOT } F) \text{ NAND } G$: start with the innermost bracket (here the NOR) and work outwards one bracket at a time; the circuit becomes a long chain from left to right.
- **An input used twice:** split its wire with a dot and send a branch to each gate that needs it. Don't draw the same input letter twice unless the question's layout does.
- **"Do not simplify":** a NOT in front of an OR is drawn as an OR gate followed by a NOT gate, exactly as written, even though a single NOR gate would behave the same.

3. Name gates, match symbols, write single-gate truth tables (16 questions)

1. Learn the six symbols until you can draw each from memory, including the circle on NAND and NOR and the extra back line on XOR.
2. For a description or a truth table, count the 1s in the output column: one 1 in the bottom row is AND, one 1 in the top row is NOR, three 1s with a 0 at the bottom is NAND, three 1s with a 0 at the top is OR, two 1s in the middle is XOR.
3. Write any single-gate truth table in the standard order 00, 01, 10, 11.

Variations you will meet:

- **Match-with-lines**, four names and five symbols: one symbol is left over, often an XOR drawn without its back line or a gate with a wrong circle.
- **Tick which statements are true for which gate**, or which gates give 1 for a given input pair: check each gate against its truth table row, one gate at a time.
- **"Name a gate that is not in this expression (and draw its symbol)"**: list the gates that are in it, then pick any of the six that isn't.
- **"Identify the gates used in this circuit"**: read each shape; don't forget that a circle on the output changes the name.
- **"Which of these shows inputs and outputs for every combination?"** The answer is a truth table (not a trace table, which is for algorithms).

4. Turn a problem statement into an expression, truth table or circuit (6 questions)

1. **Label the inputs** with the letters given and note what 1 means for each.
2. **Decide for each input whether the rule wants it to be 1 or 0.** Wanting 0 means NOT that letter.
3. **Write the expression** from the sentence, with brackets where the sentence groups conditions.
4. **Complete the truth table** from your expression, with working columns, then read two rows back against the sentence as a check.
5. If asked for a circuit, draw it from your expression as in type 2.

Variations you will meet:

- **"Either one or the other, or both"** is plain OR. Adding the "both" case as a separate AND term is also accepted, but it is longer and gives more chances to slip.
- **Every condition must hold** ("only if it is ... and ... and not ..."): chain ANDs, with NOT on any input whose 1 means the wrong thing. You can write the chain without brackets, but each extra AND is still a separate gate if you draw it.
- **Only the truth table is asked:** you still write the expression in the working space first.

5. Write an expression from a drawn circuit (2 questions)

1. **Start at the gates nearest the inputs** and write what each produces, for example "NOT X" or "Y XOR Z".
2. **Move right**, wrapping each earlier result in brackets as it goes into the next gate.
3. **Stop at the output gate** and write `output = ...`. Don't simplify; the mark scheme gives a mark for each correct piece.

From a truth table instead (in the syllabus, not yet in our tagged questions): use the "rows with a 1" method from "Understand it".

Worked examples

The bold codes show where marks are earned, as in Cambridge mark schemes. The number is how many marks.

- **B1** mark for a correct result on its own, with no method needed.
-

Example 1

(a) Two of the six logic gates give an output of 1 in exactly three of the four rows of their truth tables. Name both gates and complete a truth table for each. **[4]**

(b) Name the only gate whose output is 1 in exactly two rows. **[1]**

(a) OR and NAND. **B1** for each gate named.

First input	Second input	OR	NAND
0	0	0	1
0	1	1	1
1	0	1	1
1	1	1	0

B1 for each correct truth table.

(b) XOR (1 when the inputs differ, which happens in the rows 0 1 and 1 0). **B1**

Example 2

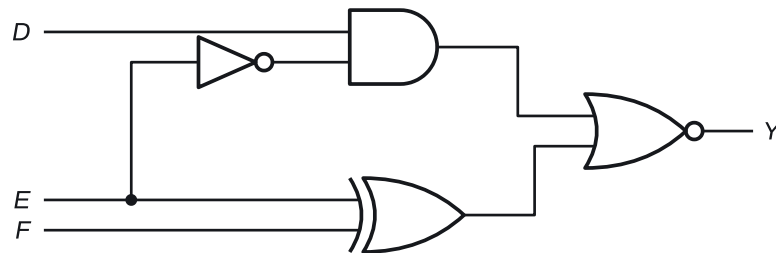
Consider the logic expression

$$Y = (D \text{ AND NOT } E) \text{ NOR } (E \text{ XOR } F)$$

(a) Draw a logic circuit for this expression. Each logic gate must have a maximum of two inputs. Do not simplify the expression. **[4]**

(b) Complete the truth table for the expression. **[4]**

(a) There are four operators (NOT, AND, XOR, NOR), so four gates:



B1 for the NOT gate on E, **B1** for the AND gate with D and NOT E, **B1** for the XOR gate with E and F, **B1** for the NOR gate combining those two outputs.

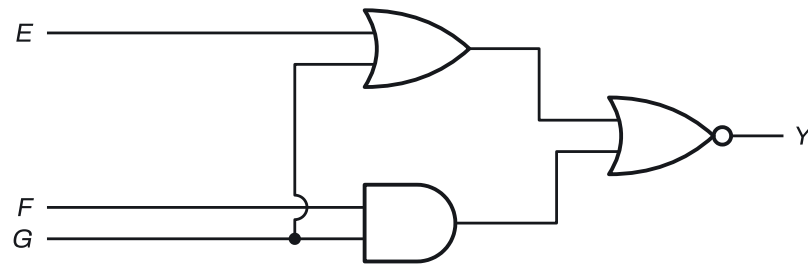
(b) Working columns first, then the output:

D	E	F	D AND NOT E	E XOR F	Y
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	0	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	0	1	0
1	1	1	0	0	1

NOR gives 1 only where both working columns are 0: rows 1, 4 and 8. **B1 B1 B1 B1** for all eight outputs correct; six or seven correct would still earn three of the four.

Example 3

Consider this logic circuit.



(a) Write a logic expression for this circuit. Do not simplify it. **[3]**

(b) Complete the truth table for the circuit. **[4]**

(a) Gates nearest the inputs: the OR combines E and G; the AND combines F and G. Both feed the last gate, a NOR.

$$Y = (E \text{ OR } G) \text{ NOR } (F \text{ AND } G)$$

B1 for E OR G, **B1** for F AND G, **B1** for joining them with NOR, brackets correct.

(b)

E	F	G	E OR G	F AND G	Y
0	0	0	0	0	1
0	0	1	1	0	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	0	0
1	1	1	1	1	0

NOR is 1 only where both working columns are 0: rows 1 and 3. **B1 B1 B1 B1** for all eight outputs.

Example 4

A car has a warning chime, C. The inputs are:

Input	Binary value	Meaning
Engine, E	0	engine off
	1	engine running
Driver's belt, B	0	belt not fastened
	1	belt fastened
Door, D	0	all doors shut
	1	a door is open

The chime sounds ($C = 1$) if the engine is running and the driver's belt is not fastened, or if a door is open.

(a) Write the logic expression for the chime. [2]

(b) Complete the truth table for the chime. [4]

(c) Draw the logic circuit for your expression. [3]

(a) "Belt not fastened" is the opposite of what 1 means for B, so it needs NOT B.

$$C = (E \text{ AND NOT } B) \text{ OR } D$$

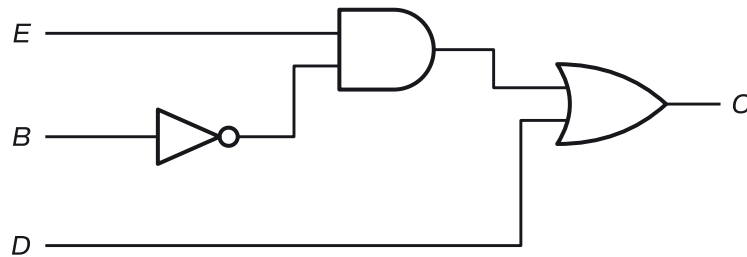
B1 for E AND NOT B, **B1** for OR D.

(b)

E	B	D	E AND NOT B	C
0	0	0	0	0
0	0	1	0	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	0	1

Check against the sentence: row 5 (engine on, belt off, doors shut) chimes; row 7 (engine on, belt on, doors shut) is silent. **B1 B1 B1 B1** for all eight outputs.

(c) Three gates: NOT, AND, OR.



B1 for NOT on B, **B1** for the AND gate with E and NOT B, **B1** for the OR gate with D.

Common mistakes

- **Mixing up NOR and XOR.** Both are "OR with something extra", and examiners have seen each written in place of the other when naming gates. NOR has a circle on the output and a single 1 (top row); XOR has a second back line and two 1s (middle rows).
- **Putting NOT on the wrong side of a gate.** NOT (G NAND H) means the NAND happens first and the NOT acts on its output; putting the NOT on an input wire before the NAND builds a different circuit and loses the mark. Any gate inside a NOT bracket works the same way.
- **Reading E AND NOT F as NOT (E AND F).** In the first, only F is inverted. Look at where the bracket is before placing the NOT.
- **Simplifying or "tidying" the expression.** Rearranging the printed expression is a reliable way to lose marks: swapping an OR-then-NOT for a NOR, or cancelling two NOTs, leaves gates the mark scheme is looking for missing.
- **Every output flipped.** If all eight of your answers are the wrong way round, it usually means the final NAND or NOR was worked as AND or OR (the flip was forgotten). Check what the last gate is before copying out the answer.
- **Shapes that look alike.** A lopsided AND can look like an OR. Draw AND with a straight back and OR with a curved back and a point, as in the syllabus, and draw the XOR back line clearly separate.
- **Crossing wires that could be joins.** If a line goes past another, show a hop or leave a clear gap; use a dot only where a wire really splits. If a junction is ambiguous, the gate it feeds may not be credited.
- **Knowing the gate names but not their tables.** Learning "NAND" as a word isn't enough: practise writing all six truth tables from memory.
- **Half an expression for a problem statement.** Many answers catch one condition and miss the other, or forget a NOT on an input whose 1 means the unwanted case. Test your expression on a row before moving on.

How to get the marks

- **Truth tables are marked on the output column.** Roughly every two correct outputs earn a mark, up to full marks for all eight, so partly correct work still scores. Working columns aren't marked, but they are how you get the output right.
- **Circuits are marked gate by gate.** Each gate scores on its own when it is the right kind and is fed by the right wires, so one slip early on doesn't wipe out the gates after it.
- **"Do not simplify"** means one gate per operator, in the order written. Don't combine or cancel anything.
- **"Each gate must have a maximum of two inputs":** a three-way AND or OR must be drawn as two gates.
- **Expressions from a circuit** are marked piece by piece (each inner gate, then the whole), so write every part even if unsure of the last step. Keep the brackets.
- **Expressions from a problem statement** are accepted whether you write them straight from the sentence or in a shorter equivalent form, as long as the truth table is the same.
- **"Draw the symbol"** needs the syllabus shape with the right number of inputs and one output; for a gate plus its truth table, each is usually a separate mark.
- **"Identify" or "State"** a gate means just the name; spell it plainly (XOR or EOR are both fine).

Quick check

1. A two-input gate gives output 1 when both inputs are 0, and output 0 when both inputs are 1. Which two gates could it be, and which input row would tell them apart?
2. A circuit has inputs D, E and F. Its output is 1 only in the rows $D, E, F = 0, 1, 0$ and $1, 0, 0$, and 0 in every other row. Write a logic expression for it.
3. For $Y = (F \text{ NOR } H) \text{ OR } (G \text{ AND } H)$, a student gave the output column (rows 1 to 8, in the usual order $F \text{ } G \text{ } H = 000$ to 111) as 1, 0, 0, 1, 0, 1, 0, 1. Two outputs are wrong. Which rows?
4. A stair light has a switch at the bottom, U, and one at the top, V (1 = switch up). Flipping either switch changes the light, so it should be on when exactly one switch is up, but only while the main power switch P is on (1). Write an expression for the light, L, using an XOR gate, and say how many of the 8 rows give $L = 1$.
5. What is the output of $(D \text{ XOR } E) \text{ NOR } F$ when $D = 1, E = 1$ and $F = 0$?

Answers

1. NAND or NOR: both give 1 for inputs 0, 0 and 0 for inputs 1, 1. Try inputs 0, 1 (or 1, 0): NAND gives 1, NOR gives 0.
2. Rows with a 1, each as an AND chain: $(\text{NOT } D \text{ AND } E \text{ AND NOT } F) \text{ OR } (D \text{ AND NOT } E \text{ AND NOT } F)$. This is the same as $(D \text{ XOR } E) \text{ AND NOT } F$, which is also correct.
3. Correct column: working $F \text{ NOR } H$ is 1, 0, 1, 0, 0, 0, 0, 0 and $G \text{ AND } H$ is 0, 0, 0, 1, 0, 0, 0, 1, so Y is 1, 0, 1, 1, 0, 0, 0, 1. The student's rows **3** and **6** are wrong.
4. $L = (U \text{ XOR } V) \text{ AND } P$. Two rows give $L = 1$: $U, V, P = 0, 1, 1$ and $1, 0, 1$.
5. $1 \text{ XOR } 1 = 0$, then $0 \text{ NOR } 0 = 1$. The output is **1**.

Past questions to try

- [0478/21/M/J/25 Q7](#): draw a circuit from an expression with XOR, then complete its truth table.
- [0478/23/O/N/24 Q8](#): a truth table from a three-condition problem statement.
- [0478/23/O/N/23 Q6](#): name gates from descriptions, write their tables, then draw a circuit.
- [0478/13/M/J/21 Q8](#): identify the gates in a circuit and find the wrong rows in its truth table.