ANSWERS

# A8.5 Simplifying expressions

Boolean algebra and logic circuits · A level · OCR H446 1.4.3, AQA 7517 4.6.5.1, Eduqas A500QS 1.2 · about 25 min

## What this lesson is about

A method for simplifying by algebra, worked exam-style examples, and checking the result.

- What "simpler" means
- A method
- Worked examples
- Checking each step by machine
- Simplifying a robot's condition

## Questions

**5 marks in all**

**1.** Simplify (A ∨ B) ∧ (A ∨ ¬B). [1 mark]

- ☐ **A** A
- ☐ **B** B
- ☐ **C** A ∨ B
- ☐ **D** 1

> **Answer:** A. Distribution gives A ∨ (B ∧ ¬B) = A ∨ 0 = A.

**2.** Simplify ¬(A ∨ B) ∨ (¬A ∧ B). [1 mark]

- ☐ **A** ¬A
- ☐ **B** ¬B
- ☐ **C** ¬A ∧ ¬B
- ☐ **D** A ∨ B

> **Answer:** A. De Morgan gives (¬A ∧ ¬B) ∨ (¬A ∧ B) = ¬A ∧ (¬B ∨ B) = ¬A.

**3.** Simplify (A ∧ B ∧ C) ∨ (A ∧ B ∧ ¬C) ∨ (A ∧ ¬B). [1 mark]

- ☐ **A** A
- ☐ **B** A ∧ B
- ☐ **C** A ∧ C
- ☐ **D** B ∨ C

> **Answer:** A. The first two terms combine to A ∧ B, and (A ∧ B) ∨ (A ∧ ¬B) = A.

**4.** Put the steps simplifying $A \cdot B + A \cdot \overline{B} + \overline{A} \cdot B$ in order. [1 mark]

Number the lines 1 to 4 to put them in the right order.

___ $A + \overline{A} \cdot B$

___ $A \cdot 1 + \overline{A} \cdot B$

___ $A + B$

___ $A \cdot (B + \overline{B}) + \overline{A} \cdot B$

**Answer:**

$A \cdot (B + \overline{B}) + \overline{A} \cdot B$
$A \cdot 1 + \overline{A} \cdot B$
$A + \overline{A} \cdot B$
$A + B$

Take out A, use the complement, use AND with 1, then $X + \overline{X} \cdot Y = X + Y$.

**5.** A student writes $\neg A \vee \neg B = \neg A \wedge \neg B$ as a step. What is wrong? [1 mark]

- ☐ **A** The operator was changed without the bar over the whole that De Morgan needs
- ☐ **B** Nothing, it is De Morgan's law
- ☐ **C** Commutation was used instead of association
- ☐ **D** The NOTs should have cancelled

**Answer:** A. $\neg A \vee \neg B$ is $\neg(A \wedge B)$, not $\neg A \wedge \neg B$; the two differ when exactly one input is 1.

## The task: simplify, then prove it

The function `original(a, b, c)` computes Q = ¬(A ∨ ¬B) ∨ (A ∧ B) ∨ (A ∧ ¬B ∧ C). Its inputs are bits (0 or 1) and it returns 0 or 1. Simplify Q on paper, then write `simplified(a, b, c)`, taking and returning the same, as a single `return` line that uses at most two of `and`, `or` and `not` in total. It may not call `original` or use the bitwise operators `&`, `|`, `^` or `~`. Print eight lines, one per row in binary order (A outermost, then B, then C), in exactly this form, with every value 0 or 1: `A=0 B=1 C=0 original=1 simplified=1`

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

def original(a, b, c):
    return int((not (a or not b)) or (a and b) or (a and not b and c))

def simplified(a, b, c):
    return 0
```

**The hint students can ask for:** Work on paper first. Apply De Morgan to the bracket with NOT in front, then look for two terms that differ only in one letter being inverted, and for a letter that can absorb or cancel part of the last term. Check your answer against the truth table the program prints.

## A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

def original(a, b, c):
    return int((not (a or not b)) or (a and b) or (a and not b and c))

def simplified(a, b, c):
    return int(b or (a and c))

for a in [0, 1]:
    for b in [0, 1]:
        for c in [0, 1]:
            print(f"A={a} B={b} C={c} original={original(a, b, c)} simplified=
{simplified(a, b, c)}")
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.