WORKSHEET

A8.4 De Morgan's laws

Boolean algebra and logic circuits · A level · OCR H446 1.4.3, AQA 7517 4.6.5.1, Eduqas A500QS 1.2 · about 20 min

BugBotLab

Name _____ Class _____ Date _____

What this lesson is about

Breaking the bar and changing the sign, NAND and NOR, and rewriting the robot's loop condition.

- The two laws
- Break the bar, change the sign
- More than two inputs, and nested brackets
- Why NAND and NOR are universal
- De Morgan in your programs

Questions

6 marks in all

1. By De Morgan's law, ¬(A ∧ B) equals which expression? [1 mark]

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

2. By De Morgan's law, ¬(A ∨ B ∨ C) equals which expression? [1 mark]

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

3. What does ¬(¬A ∨ ¬B) simplify to? [1 mark]

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

4. Which Python condition means the same as not (close or hit)? [1 mark]

- ☐ **A** not close and not hit
- ☐ **B** not close or not hit
- ☐ **C** not close and hit
- ☐ **D** close and not hit

5. In AQA notation, how do $\overline{A} \cdot \overline{B}$ (a bar over each letter) and a single bar over the whole of $A \cdot B$ differ? [1 mark]

- ☐ **A** The first is NOR and the second is NAND
- ☐ **B** They are the same function
- ☐ **C** The first is NAND and the second is NOR
- ☐ **D** The first is XOR and the second is AND

6. What does this program print? [1 mark]

```
result = []
for a in [0, 1]:
    for b in [0, 1]:
        result.append((1 - (a & b)) == ((1 - a) | (1 - b)))
print(all(result))
```

The task: De Morgan at the wall

The robot starts facing a wall. First print four check lines, one for each combination of `close` and `hit` (each 0 or 1, `close` in the outer loop), in exactly this form: `close=0 hit=1 original=0 rewritten=0` where `original` is NOT (close OR hit) and `rewritten` is your De Morgan version, (NOT close) AND (NOT hit), each worked out as 0 or 1 by a function of your own. Then drive the robot towards the wall and stop in the band before it. The loop must be a `while` loop whose condition reads `distance()`, stops when the distance is under 20 cm or `bumped()` is true, and is written using De Morgan so that it joins its two parts with `and not`, with no `not` in front of a bracket.

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

while not (distance() < 20 or bumped()):
    forward(50)
    wait(0.1)
stop()
```

Plan your program here, then type it in and press Run.

Do it on the robot

www.bugbotlab.com/learn/a8-4-de-morgans-laws/

The simulator checks it and tells you when it passes. Nothing to install, no account.

Challenges

1. Rewrite `if not (battery() > 20 and distance() > 30):` without a `not` in front of the bracket.
2. Simplify ¬(¬A ∨ ¬B) in two lines. Which single gate is left?
3. Draw NOR as an AND gate with bubbles. Build NOT, OR and AND from NOR functions alone, and prove each by printing its truth table.