ANSWERS

F9.2 Logic circuits and expressions

Logic and computer systems · GCSE · OCR J277 2.4.1, AQA 8525 3.4.2, Edexcel 1CP2 1.3.1 · about 15 min

What this lesson is about

Combining gates, Boolean expressions, XOR, and the half adder.

- A circuit and its expression
- XOR
- Adding two bits
- From a problem to a circuit

Questions

6 marks in all

1. In Q = (A AND B) OR (NOT C), A = 0, B = 1 and C = 1. What is Q? [1 mark]

> **Answer:** 0. A AND B is 0, NOT C is 0, and 0 OR 0 is 0.

2. In Q = (A AND B) OR (NOT C), A = 0, B = 0 and C = 0. What is Q? [1 mark]

> **Answer:** 1. NOT C is 1, so the OR gives 1.

3. When does XOR output 1? [1 mark]

- ☐ **A** When the inputs are different
- ☐ **B** When both inputs are 1
- ☐ **C** When at least one input is 1
- ☐ **D** When both inputs are 0

> **Answer:** A. Exclusive OR: one or the other, but not both.

4. In a half adder, which gate gives the carry bit? [1 mark]

- ☐ **A** AND
- ☐ **B** XOR
- ☐ **C** OR
- ☐ **D** NOT

> **Answer:** A. The carry is 1 only when both bits are 1.

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

```
def XOR(a, b): return 1 if a != b else 0
for a, b in [(1, 1), (1, 0)]:
    print(a & b, XOR(a, b))
```

> **Answer:**
>
> ```
> 1 0
> 0 1
> ```
>
> 1 + 1 is carry 1, sum 0; 1 + 0 is carry 0, sum 1.

6. "Beep if the robot is close and moving, or it has bumped." Which expression matches? [1 mark]

- ☐ **A** (close AND moving) OR bumped
- ☐ **B** close AND (moving OR bumped)
- ☐ **C** (close OR moving) AND bumped
- ☐ **D** NOT close AND moving AND bumped

> **Answer:** A. The and joins close and moving; the or adds bumped on its own.

The task: a half adder

Write `AND(a, b)` and `XOR(a, b)` as functions of 0s and 1s. XOR must be built from AND, OR and NOT functions you also write, as `(A OR B) AND NOT (A AND B)`, not with `!=`. Print the half adder's truth table as four lines in the form `A=1 B=1 carry=1 sum=0`.

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

def AND(a, b): return 1 if a and b else 0
def OR(a, b): return 1 if a or b else 0
def NOT(a): return 1 - a
```

The hint students can ask for: XOR is true when the inputs differ, which is the same as saying at least one is true but not both: build it from OR, AND and NOT. The carry is the plain AND of the inputs.

A solution

```
from bugbot import *
connect()
def AND(a, b):
    return 1 if a == 1 and b == 1 else 0

def OR(a, b):
    return 1 if a == 1 or b == 1 else 0

def NOT(a):
    return 1 - a

def XOR(a, b):
    return AND(OR(a, b), NOT(AND(a, b)))

for a in [0, 1]:
    for b in [0, 1]:
        print(f"A={a} B={b} carry={AND(a, b)} sum={XOR(a, b)}")
```

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