Exercise 4

Module 4: Logical Operations

1. Complete the truth table for the following expression:

```
result = (A and not B) or (not A and C)
```

Fill in all 8 rows. Show your working in the space below each row.

A	B	C	not B	not A	A and not B	not A and C	result
False	False	False					
False	False	True					
False	True	False					
False	True	True					
True	False	False					
True	False	True					
True	True	False					
True	True	True					

2. **Find and Fix the Logic Bug**

Each code block below contains a logical error. Identify the bug, explain what it causes, and write the corrected version.

(a)

```
# Should alert if temp is between -5 and 2
if temp > -5 and < 2:
    send_alert()
```

(b)

```
# Should irrigate only when NEITHER condition is true
if not frost_risk or not heavy_rain:
    irrigate()
```

(c)

```
# Should check if any of three sensors triggered
if sensor_a or sensor_b or sensor_c = True:
    log_event()
```

3. Design Your Own Decision System

You are building the logic for a greenhouse monitoring system. Design a Python if/elif/else block that handles the following requirements:

1. If the CO_2 level is above 1,500 ppm AND the fan is off → turn the fan on
2. If the temperature is above 35°C OR the humidity is above 90% → open the roof vent
3. If it is night-time (time_of_day = 'night') AND temperature drops below 12°C → turn heating on
4. Otherwise → log that all conditions are normal

(a) Define the variables you need, with sensible example values.

(b) Write the full if/elif/else block.

(c) Trace through your code with your example values and state which block runs.

4.

Write the following code to decide whether to apply fertiliser:

```
apply = ndvi < 0.4 or soil_n < 50 and not recently_fertilised
```

(a) Explain in what order Python evaluates this expression. Use the precedence rules.

(b) The student intended: 'apply fertiliser if (NDVI is low OR nitrogen is low) AND the field has not been recently fertilised'. Does the code match this intention? If not, rewrite it correctly.

(c) Explain why using parentheses is considered good practice in complex conditions, even when technically unnecessary.