MODULE M PROGRAMMING · M7

Boolean operators AND, OR and NOT in code: mark scheme

22 marks · Cambridge 0478 syllabus 8.1.4(f) (arithmetic, relational and logical operators), 8.1.4(b) (selection), 7.8 (identify and correct errors), 7.9 (write algorithms)

How to use. One mark per creditworthy point up to the maximum shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Algorithm answers must be in pseudocode; marks are for the features listed, not for one exact algorithm. Ignore minor slips in case and spacing. Accept <- or = for assignment. Do not accept English ("is more than") or maths symbols (≥, ≤, ≠) in place of operators. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 8.1.4(f)	One mark for each correct row: • either one or both TRUE: OR • only when both TRUE: AND • the opposite of a single Boolean value: NOT	**3**	Do not accept NAND, NOR or XOR (gate names, not pseudocode logical operators). Accept lower case.
2(a) 0478 8.1.4(f)	One mark for each correct row: • `(Humidity > 60) AND (Level = 3)`: TRUE • `(NOT FanOn) OR (Level > 5)`: FALSE • `NOT ((Humidity >= 64) AND FanOn)`: FALSE	**3**	Accept T/F, 1/0. Working: row 1 TRUE AND TRUE = TRUE; row 2 FALSE OR FALSE = FALSE; row 3 64 >= 64 is TRUE, so NOT (TRUE AND TRUE) = FALSE.
3(a) 0478 8.1.4(f)	• both correct comparisons: `Setting >= -25` // `Setting > -26` and `Setting <= -15` // `Setting < -14` • joined with `AND` Example: `(Setting >= -25) AND (Setting <= -15)`	**2**	Brackets not required. Both boundaries must be included: `> -25` or `< -15` loses the first mark. Comparisons written the wrong way round for negatives (e.g. `Setting <= -25 AND Setting >= -15`, never TRUE) loses the first mark. Each side must be a full comparison: `Setting >= -25 AND <= -15` = 1 (AND mark only). Accept `NOT ((Setting < -25) OR (Setting > -15))` for 2. OR between the two in-range checks: max 1.

QUESTION	ANSWER	MARKS	GUIDANCE
3(b) 0478 8.1.4(f)	• both correct comparisons: `Setting < -25` // `Setting <= -26` and `Setting > -15` // `Setting >= -14` • joined with `OR` Example: `(Setting < -25) OR (Setting > -15)`	**2**	A comparison that rejects a valid boundary (`<= -25` or `>= -15`) loses the first mark. AND in place of OR (never TRUE): max 1. The question asks for OR: a correct `NOT (... AND ...)` with no OR gets max 1.
4(a) 0478 7.8, 8.1.4(f)	One mark for each line identified, one mark for each matching correction: • Line 03 • `IF Day = "Sat" OR Day = "Sun"` // AND should be OR • Line 07 • `IF Hour >= 6 AND Hour <= 9` // the second comparison needs `Hour` in front of `<= 9`	**4**	The correction mark needs the right line number. Accept `Hour > 5 AND Hour < 10` (Hour is a whole number), with or without brackets. Line 03: accept `(Day = "Sat") OR (Day = "Sun")`. Do not accept changing line 07 to OR (TRUE at every hour). Do not accept changes to lines 01, 02, 12 or 13.
4(b) 0478 8.1.4(f)	Any **two** from: • AND gives TRUE only when both conditions are TRUE • `Day` can hold only one value at a time // cannot be "Sat" and "Sun" at the same time • so (at least) one comparison is always FALSE // the condition on line 03 is always FALSE (so the ELSE branch always runs)	**2**	"It should be OR" alone (no reason) = 0. Do not credit answers about line 07.

QUESTION	ANSWER	MARKS	GUIDANCE

5(a)
0478 7.9, 8.1.4(f), 8.1.4(b)
	One mark per bullet point. Max 6:
	• `INPUT` battery charge and wind speed (into two variables)
	• correct charge check: `Charge >= 40` // `Charge > 39` // (reversed form) `Charge < 40` // `Charge <= 39`
	• correct wind check: `WindSpeed < 30` // `WindSpeed >= 30`
	• correct check that it is not raining: `NOT Raining` // `Raining = FALSE` // `Raining <> TRUE` // `Raining` // `Raining = TRUE`
	• all three checks joined with `AND` // all three failing checks joined with `OR` (outputs swapped) // correctly nested IFs, so take-off is allowed only when all three are met
	• `IF ... THEN ... ELSE ... ENDIF` with `"Take-off allowed"` and `"Take-off blocked"` output in the correct places
	Example:
	`INPUT Charge`
	`INPUT WindSpeed`
	`IF (Charge >= 40) AND (WindSpeed < 30)`
	`AND (NOT Raining)`
	`  THEN`
	`  OUTPUT "Take-off allowed"`
	`  ELSE`
	`  OUTPUT "Take-off blocked"`
	`ENDIF`
		6	Accept any sensible identifiers. Prompts before INPUT are not required. `Charge > 40` or `WindSpeed <= 30` loses that mark. A second side with no variable (e.g. `Charge >= 40 AND < 30`) loses the wind mark. OR used in a condition that tests for take-off being allowed loses the AND mark. Accept the reversed form: all three failing checks joined with OR (e.g. `(Charge < 40) OR (WindSpeed >= 30) OR Raining` // `Raining = TRUE`), with "Take-off blocked" output when it is TRUE and "Take-off allowed" in the ELSE. This scores the check marks, the joining mark and the output mark. Nested IFs: the last mark needs "Take-off blocked" output for every failing case. Do not penalise an extra INPUT of Raining. Program code (e.g. Python) in place of pseudocode: 0.

Total: 22 marks