MODULE M PROGRAMMING · M12

Post-condition loops, and choosing the right loop: mark scheme

22 marks · OCR J277 spec 2.2.1 (iteration: count- and condition-controlled loops), 2.1.2 (trace tables; refine algorithms; OCR Exam Reference Language)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip (e.g. = for comparison) is penalised once then follow through. Capitals in keywords are ignored. Do not accept maths symbols (≥, ≤, ≠) or English ("is more than") in place of operators. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.1	Condition checked before the code inside runs: `while` Code inside always runs at least once: `do ... until` It is a condition-controlled loop: Both	**3**	1 mark per correct row. More than one tick in a row: 0 for that row.
2(a) J277 2.1.2, 2.2.1	1 mark each: `temp`: 78, 60, 42, 24 (after the given 96) `minutes`: 1, 2, 3, 4 (after the given 0) Outputs 78, 60, 42, 24 (in this order) Final output `Minutes to cool: 4` (as the last output, once only)	**4**	Each mark is for the whole sequence of values in that column, in order. Values may be repeated on later rows, and changes may be shown on separate rows. Ignore quotation marks round outputs. Do not award the output marks if `print(...)` is written round the values. Extra values after 24 (e.g. a further 6 or another pass of the loop) lose the mark for that column. Accept `Minutes to cool:4` (spacing).
2(b) J277 2.2.1	`17` `Minutes to cool: 1`	**2**	1 mark each. Working: the loop body runs once before the condition is checked, so `temp` becomes 35 − 18 = 17 and `minutes` becomes 1; then 17 < 40 is True, so the loop ends. "Nothing is output" / only `Minutes to cool: 0` = 0 (treats the loop as checked first).

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) J277 2.2.1	Lighthouse: `for` (count-controlled) Quiz machine: `do ... until` Robot vacuum cleaner: `while`	**3**	1 mark per row. Row 1: accept "count-controlled". Rows 2 and 3: "condition-controlled" alone = 0 (it does not say which loop). Row 2: do not accept `while` (the task must run at least once). Row 3: do not accept `do ... until`.
3(b) J277 2.2.1	1 mark each to max 2: A `do ... until` loop runs the code inside at least once / checks its condition only after the code has run ... ... so the robot would move forward (at least once) before the bumper is checked The bumper may already be pressed / the robot may already be against a wall, so it should not move at all / the loop needs to be able to run zero times A `while` loop checks the bumper first, so the robot does not move if it is already touching the wall	**2**	Max 2. The second bullet depends on the first being stated or clearly implied. "A while loop is better" alone (no reason) = 0. Accept "it would crash / push into the wall" as the consequence for the second bullet.

4(a)
J277 2.2.1, 2.1.2
	1 mark each:
	`level` initialised before the loop (to 1, or 0 if incremented before the output)
	`do ... until` structure with the output and `level = level + 1` both inside the loop
	Correct `until` condition so the loop stops after `Level 4` is output, e.g. `until level > 4` / `until level == 5` / `until level >= 5`
	Example:
	`level = 1`
	`do`
	`  print("Level " + str(level))`
	`  level = level + 1`
	`until level > 4`
	Alternative: `level = 0`, then inside the loop `level = level + 1` before the print, `until level == 4`.
		3	The output must be exactly `Level 1` to `Level 4`. `until level == 4` with the increment after the print outputs only Levels 1 to 3: loses the third mark. Do not award the second mark for a `while` or `for` loop (the question asks for `do ... until`); the first and third marks can still be given if the logic is the same. Accept `level += 1`. Ignore a missing `str()`.

QUESTION	ANSWER	MARKS	GUIDANCE

5(a)
J277 2.2.1 (2.3.1 input validation)
	1 mark each to max 5:
	Inputs the target temperature and stores it in a variable
	Attempt at a condition-controlled loop (`do ... until` or `while ... endwhile`) ...
	... with the input inside the loop, so the user is asked again each time
	Correct range test with both boundaries included: `target >= 10 AND target <= 30` (or, for invalid, `target < 10 OR target > 30`)
	Loop condition the right way round for the loop used (a `do ... until` ends when the target is valid; a `while` repeats while it is invalid) and `Target set to` with the target output once, after the loop
	Example:
	`do`
	`  target = int(input("Enter target temperature (10 to 30)"))`
	`until target >= 10 AND target <= 30`
	`print("Target set to " + str(target))`
	Example with a `while` loop:
	`target = int(input("Enter target temperature (10 to 30)"))`
	`while target < 10 OR target > 30`
	`  target = int(input("Must be 10 to 30. Enter again"))`
	`endwhile`
	`print("Target set to " + str(target))`
		5	Allow any logically valid method. A `while` answer that inputs before the loop and again inside it gets the "input inside the loop" mark. Accept `target > 9 AND target < 31`. Each side of a compound condition must be a full comparison: `target >= 10 AND <= 30` loses the range mark. OR joining the two valid checks / AND joining the two invalid checks loses the range mark. A single `if` with no loop (only one re-entry): max 2 (input mark and range mark). Accept a Python `while True:` loop with `break` when valid. Do not penalise a missing `int()`. Do not accept `input` as a variable name. Do not accept flowcharts, structured English or bullet points.

Total: 22 marks