MODULE M PROGRAMMING · M19

Two-dimensional arrays and looping through them: mark scheme

21 marks · Cambridge 0478 syllabus 8.2.1 (declare and use 1D and 2D arrays), 8.2.2 (understand the use of arrays), 8.2.3 (write values into, and read values from, an array using iteration, including nested iteration), 7.7 (dry-run an algorithm), 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. 2D indexes must be in the order given in the question. 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 8.2.1	One mark for each: • `DECLARE Rainfall : ARRAY … OF REAL` • `[1:12, 1:5]` Example: `DECLARE Rainfall : ARRAY[1:12, 1:5] OF REAL`	**2**	The data type must be REAL (values have one decimal place); INTEGER or STRING loses the first mark. `[1:5, 1:12]` (bounds in the wrong order) loses the second mark. Do not accept `[12, 5]` or `[0:12, 0:5]` (the question asks for a lower bound of 1).
1(b) 0478 8.2.2	Any **two** from: • only one identifier is needed for all the values // fewer variables to declare and name • related data is kept together in one structure // data is organised as a table of months and reservoirs • the values can be processed using (nested) loops, with variables as the indexes • less code needs to be written // the program is easier to write / maintain / understand	**2**	Do not accept "faster" or "uses less memory" alone. Do not accept "more data can be added later" (arrays have a fixed size). The two benefits must be different points.
2(a) 0478 8.2.1	`'E'` // E // empty	**1**	Row 4 is F E F F F, so column 2 holds 'E'.
2(b) 0478 8.2.1	`Lockers[1, 4] ← 'F'`	**1**	Indexes must be 1 then 4. Accept `"F"`. Do not accept the value on the left, or `Lockers[4, 1]`.
2(c) 0478 8.2.1	There is no row 5 // the upper bound of the first index is 4 // index 5 is outside the bounds / range of the array	**1**	Do not accept "there is no column 2" or "the locker is full". "The array is too small" alone: do not credit.

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) 0478 8.2.3 7.7	One mark for each correct line, in this order: • `Row 1: 2 empty` • `Row 2: 0 empty` • `Row 3: 4 empty` • `Row 4: 1 empty`	**4**	Ignore spacing and quotation marks. The words must be included (a line of numbers only, e.g. `1  2`, is not the output: max 2 for the whole part if every line is numbers only). Lines in the wrong order: max 3.
3(b) 0478 8.2.3	So the count is reset to 0 for each new row // so each row is counted separately // otherwise the count from earlier rows would be added on (giving a running total)	**1**	Accept "otherwise row 3 would show 6 and row 4 would show 7". "To initialise Count" alone: do not credit (it does not say why inside the loop).
4(a) 0478 7.8 8.2.3	One mark for each error identified **and** corrected: • Line 01: `FOR Day ← 1 TO 7` • Line 04: `DayTotal ← DayTotal + Sales[Kiosk, Day]` • Line 05: `NEXT Kiosk`	**3**	The line number alone, or a correction without its line number: 0 for that error. Line 04: the indexes must be swapped and the addition kept. Do not credit changes to lines 02, 03, 06 or 07 (they are correct). Do not credit "line 03 should be 1 TO 7" (there are only 4 kiosks).

QUESTION	ANSWER	MARKS	GUIDANCE

5(a)
0478 7.9
8.2.3
	One mark per bullet point. Max 6:
	• input of the tank number inside a loop that repeats until it is from 1 to 6 (e.g. `REPEAT … UNTIL Tank >= 1 AND Tank <= 6`)
	• total set to 0 before the loop
	• loop for the 8 tests: `FOR Test ← 1 TO 8` … `NEXT Test`
	• adds each result for that tank to the total: `Total ← Total + TankPH[Tank, Test]`
	• calculates total / 8 and outputs it rounded to one place, with a message: `ROUND(Average, 1)`
	• inside the loop, checks each result `< 6.5` and sets a flag / adds to a count
	• after the loop, outputs `Check water` once, only if the flag is set / count > 0
	Example:
	`REPEAT`
	`  OUTPUT "Enter a tank number from 1 to 6"`
	`  INPUT Tank`
	`UNTIL Tank >= 1 AND Tank <= 6`
	`Total ← 0.0`
	`LowPH ← FALSE`
	`FOR Test ← 1 TO 8`
	`  Total ← Total + TankPH[Tank, Test]`
	`  IF TankPH[Tank, Test] < 6.5`
	`  THEN`
	`  LowPH ← TRUE`
	`  ENDIF`
	`NEXT Test`
	`Average ← Total / 8`
	`OUTPUT "Average pH for tank ", Tank, " is ", ROUND(Average, 1)`
	`IF LowPH = TRUE`
	`  THEN`
	`  OUTPUT "Check water"`
	`ENDIF`
		6	Accept a pre-condition loop (`INPUT Tank` then `WHILE Tank < 1 OR Tank > 6 DO … INPUT Tank … ENDWHILE`). A single IF re-input (only one retry) does not get the first mark. `TankPH[Test, Tank]` (indexes reversed): lose the fourth mark. Dividing by 6, or rounding to 2 places: lose the fifth mark. `<= 6.5` loses the sixth mark. `Check water` output inside the loop (may be output many times): lose the last mark. Accept a nested loop over all tanks only if it restricts the results to the input tank. Do not accept program code (e.g. Python) in place of pseudocode. No declarations needed.

Total: 21 marks