MODULE M PROGRAMMING · M19

Two-dimensional arrays and looping through them: mark scheme

21 marks · OCR J277 spec 2.2.3 (the use of arrays when solving problems, including 1D and 2D arrays; arrays as fixed length or static structures; use of 2D arrays to emulate database tables of a collection of fields and records), with 2.2.1 (iteration)

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. 2D indexes may be written `[r,c]` or `[r][c]`, but must be in the row, column order given in the question. Minor syntax slips are not penalised; a repeated slip (e.g. = for comparison) is penalised once then follow through. Accept Python `range(0, 5)` / `range(5)` for `0 to 4`. Do not accept flowcharts, structured English or bullet points for write tasks.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.2.3	Every element must be of the same data type: True `grid[4,6]` can be used: False Extra rows can be added while running: False	**3**	1 mark per correct row. Both boxes ticked in a row: 0 for that row. Row 2: `array grid[4,6]` gives indexes 0 to 3 and 0 to 5, so `grid[4,6]` is outside the array. Row 3: arrays are fixed length / static.
2(a) J277 2.2.3	Haddad	**1**	Accept with or without quotation marks. Do not accept "Yoga" or "Lindqvist".
2(b) J277 2.2.3	`instructors[4,2] = "Yoga"`	**1**	Accept `instructors[4][2] = "Yoga"`. Indexes must be 4 then 2. Allow missing quotes (minor slip). Do not accept ==, or the value on the left (`"Yoga" = instructors[4,2]`).

QUESTION	ANSWER	MARKS	GUIDANCE

2(c)
J277 2.2.3
	1 mark each to max 3:
	Loop over every record / row 0 to 4
	Checks the activity field of the current row: `instructors[r,2] == "Climbing"`
	Outputs the surname field of the same row: `print(instructors[r,1])` (inside the IF)
	Example:
	`for r = 0 to 4`
	`  if instructors[r,2] == "Climbing" then`
	`  print(instructors[r,1])`
	`  endif`
	`next r`
		3	The check and output marks need the loop variable as the first (row) index. Accept a while loop with a counter, and `for r = 0 to instructors.length - 1` or `len(instructors)`. Loop 1 to 5 or 0 to 5: loop mark lost, FT for the other two marks. Hard-coded checks of rows 1 and 3 with no loop: 0 (iteration is required). Output of the whole record or the ID instead of the surname: lose the output mark.

3(a) J277 2.2.3	`177`	**1**	Working: 55 + 49 + 44 + 29 = 177 (row index 1, every reading).

3(b)
J277 2.2.3
2.1.2
	1 mark per correct line:
	`0 2`
	`1 3`
	`2 1`
		3	Values below 30 are 27 (bed 0, reading 2), 29 (bed 1, reading 3) and 26 (bed 2, reading 1); the outer loop works through the beds in order. Ignore spacing (accept `02`). Do not accept the moisture values (27, 29, 26) instead of the indexes. All three correct but in the wrong order: max 2. Any extra line output: max 2.

3(c)
J277 2.2.3
	1 mark each to max 3:
	Loop over the three beds (0 to 2) adding to a running total that starts at 0
	Adds the correct element each time: `moisture[bed,3]`
	Divides the total by 3 (after the loop) and outputs the result
	Example:
	`total = 0`
	`for bed = 0 to 2`
	`  total = total + moisture[bed,3]`
	`next bed`
	`print(total / 3)`
		3	The result is 100 / 3 = 33.3 (recurring); candidates do not need to state it. Accept `moisture[bed][3]`. `moisture[3,bed]` (indexes reversed): lose the second mark. Dividing inside the loop and outputting once correctly: allow. Dividing by 4: lose the third mark. No loop (three values added directly): max 1 (third mark only). Do not penalise rounding the output.

4(a)
J277 2.2.3
2.2.1
	1 mark each to max 6:
	Counter set to 0 before the loops
	Outer loop over the rows 0 to 4 ...
	... with an inner loop over the lockers 0 to 7 nested inside it
	Checks `rented[row, locker]` using both loop variables, row first, for a free locker (`== False` / `NOT`) ...
	... and adds 1 to the counter when it is free
	Outputs the counter once, after both loops have finished
	Outputs `All lockers rented` only when the count is 0
	Example:
	`free = 0`
	`for row = 0 to 4`
	`  for locker = 0 to 7`
	`  if rented[row, locker] == False then`
	`  free = free + 1`
	`  endif`
	`  next locker`
	`next row`
	`print(free)`
	`if free == 0 then`
	`  print("All lockers rented")`
	`endif`
		6	Accept `if NOT rented[row, locker]` / `!= True`. Accept counting rented lockers and outputting `40 - count` (then "All lockers rented" when 40 are rented). Accept loops that go 1 to 5 / 1 to 8 only if the index is adjusted (`row - 1`); otherwise lose the loop mark it affects. Indexes reversed (`rented[locker, row]`): lose the check mark. Output inside the loops (prints many times): lose the output mark. Accept a flag instead of the count for the last mark. Accept the words `All lockers rented` with any minor change in capitals. Do not accept `input` as a variable name.

Total: 21 marks