MODULE L STANDARD ALGORITHMS · L7

Totalling, counting, maximum, minimum and average: mark scheme

21 marks · OCR J277 spec 2.1.2 (creating algorithms, common errors, trace tables), 2.2.1 (variables, operators, inputs, outputs, assignment, iteration, selection), 2.2.3 (arrays)

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: mark the logic, not the language. Allow OCR Exam Reference Language or any high-level language, and allow any logically valid method. Do not penalise minor syntax slips (missing casting, capitals, a missing colon). A recurring slip: penalise once, then follow through. Do not accept maths symbols such as ×, ≥, ≤ in code.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1(a)
J277 2.2.1, 2.2.3</td><td>Line 01: <code>0</code>
Line 03: +
Line 05: <code>sales.length / 12</code></td><td>3</td><td>Line 05: allow <code>len(sales)</code> or any expression that gives 12. Do not accept <code>i</code>, <code>11</code> or <code>sales.length - 1</code>.</td></tr>
<tr><td>2(a)
J277 2.1.2, 2.2.1</td><td>day: 1, 2, 3, 4
<code>wettest</code>: 4, 12, 15
<code>dryDays</code>: 0, 1
Output: 15 then 1
<table>
<tr><th>DAY</th><th>WETTEST</th><th>DRYDAYS</th><th>OUTPUT</th></tr>
<tr><td></td><td>4</td><td>0</td><td></td></tr>
<tr><td>1</td><td>12</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td>1</td><td></td></tr>
<tr><td>3</td><td>15</td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td>15</td></tr>
<tr><td></td><td></td><td></td><td>1</td></tr>
</table></td><td>4</td><td>1 mark for each column with the correct values in the correct order. Ignore the row a value is on, and allow values to be repeated on later rows. Do not award the day mark if 0 or 5 appears. Outputs written as <code>print(15)</code> or in quotation marks: do not award the Output mark.</td></tr>
<tr><td>2(b)
J277 2.1.2</td><td>The loop starts at index 1 / <code>rain[0]</code> is never checked (to see if it is 0) / the first day is missed (and it was a dry day)
<code>for day = 0 to 4</code></td><td>2</td><td>Allow <code>for day = 0 to rain.length - 1</code>. Credit any other refinement that also checks <code>rain[0]</code> for the second mark. Do not accept "the loop is wrong" alone for the first mark.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE

3(a)
J277 2.1.2, 2.2.1, 2.2.3
	1 mark each to max 4:
	Sets a variable for the smallest to `heights[0]` / the first value / a value larger than any height (e.g. 1000) before the loop
	Loop that visits every item (index 0 to 29 / `heights.length - 1`; starting at 1 is allowed only if `smallest` was set to `heights[0]`)
	Compares each item with the smallest so far (`heights[i] < smallest`)
	... and replaces the smallest with that item when it is smaller
	Outputs the smallest after the loop
	`smallest = heights[0]`
	`for i = 1 to 29`
	`    if heights[i] < smallest then`
	`        smallest = heights[i]`
	`    endif`
	`next i`
	`print(smallest)`
		4	Do not award the first bullet for starting at 0 (no height is smaller than 0). A loop of 0 to 29 is fine. Start value of 1000 (or similar) with a loop starting at 1: do not award the loop bullet, max 3. Output inside the loop: do not award the last bullet. Use of `min()` or a built-in sort: max 1 (for the output). Allow <= for the comparison.

4(a) J277 2.1.2, 2.2.1	0 is higher than every temperature (in the array) so `highest` is never replaced / the program outputs 0, which is not one of the temperatures	**1**	Do not accept "0 is the wrong number" or "it is a logic error" alone.
4(b) J277 2.1.2, 2.2.1	`highest = temps[0]`	**1**	Allow a starting value lower than any temperature the freezer could record, e.g. `highest = -1000`. Do not accept 0, or a value that could be higher than every reading (e.g. -20).

QUESTION	ANSWER	MARKS	GUIDANCE

5(a)
J277 2.1.2, 2.2.1
	1 mark each to max 6:
	Inputs the number of repairs and stores it in a variable
	Loop that repeats once for each repair (uses the number input)
	Inputs a time inside the loop
	Total set to 0 before the loop and the time added to it inside the loop
	Counter set to 0 before the loop and increased by 1 inside the loop when time > 45
	Mean calculated as total / number of repairs (after the loop)
	Outputs the total, the count and the mean (after the loop)
	`repairs = int(input("How many repairs today?"))`
	`total = 0`
	`over45 = 0`
	`for r = 1 to repairs`
	`    time = int(input("Time in minutes"))`
	`    total = total + time`
	`    if time > 45 then`
	`        over45 = over45 + 1`
	`    endif`
	`next r`
	`print("Total time: " + str(total))`
	`print("Over 45 minutes: " + str(over45))`
	`print("Mean time: " + str(total / repairs))`
		6	Count bullet: do not accept >= 45. Allow a while loop with its own counter. Allow the mean worked out with a separate count of repairs. Allow uninitialised total/counter to be taken as 0 if the language allows it. Not required: checking for 0 repairs, rounding, messages with the outputs. Loop with a fixed number of repeats (e.g. 10): do not award the loop bullet.

Total: 21 marks