MODULE M PROGRAMMING · M1

Variables, constants, input and output: mark scheme

22 marks · Cambridge 0478 syllabus (exams 2026-2028) 8.1.1 (declare and use variables and constants), 8.1.3 (input and output), 8.1.4(a) (sequence), with 7.7 (trace tables) and 7.8 (identifying and correcting errors)

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. Credit any other valid answer that shows the same understanding. Written algorithms must be in pseudocode; ignore minor syntax slips that do not change the logic.

QUESTION	ANSWER	MARKS	GUIDANCE
1 0478 8.1.1, 8.1.4(a)	One mark for each correct term: A named value that is set once and cannot change ...: Constant Statements are carried out one after another ...: Sequence A named memory location that stores a value that can change ...: Variable	**3**	Input and Output are not used. More than one term in a box scores 0 for that box.
2(a) 0478 8.1.1	`TicketPrice`	**1**	Ignore capitals. Do not credit 7.50 (the value, not the identifier).
2(b) 0478 8.1.1	`Adults` // `TotalCost`	**1**	Any one. Do not credit `TicketPrice`, `INTEGER` or `REAL`.
2(c) 0478 8.1.3	Total to pay: 30.0	**1**	Message and value both needed. Accept 30 // 30.00 for the value. Ignore a missing colon or space. Do not credit quotation marks around the output, or `TotalCost` in place of the value.
2(d) 0478 8.1.1	Any two from: If the price changes it only needs to be changed once / in line 01 // easier to update (the algorithm) The identifier shows what the value means // makes the algorithm easier to read / understand The value cannot be changed (by mistake) while the program is running The same value is used everywhere the price is needed // avoids typing errors in the value	**2**	max 2. Do not credit "faster", "uses less memory" or "more secure". "Easier to change" is credited only as updating the value in the code, not as changing it while the program runs.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>3(a)
0478 8.1.1, 8.1.3, 8.1.4(a) (7.7)</td><td>
<table>
<tr><th>POINTS</th><th>BONUS</th><th>OUTPUT</th></tr>
<tr><td>3</td><td></td><td></td></tr>
<tr><td></td><td>6</td><td></td></tr>
<tr><td>9</td><td></td><td></td></tr>
<tr><td></td><td>18</td><td></td></tr>
<tr><td>1</td><td></td><td>1 and 18</td></tr>
</table>
One mark for each correct column:
<code>Points</code>: 3, 9, 1 (in this order)
<code>Bonus</code>: 6, 18 (in this order)
OUTPUT: 1 and 18 (only)
</td><td>3</td><td>Ignore the row a value is written on and values repeated on later rows. OUTPUT: do not credit quotation marks around the output, “9 and 18” or more than one output. Ignore spacing.</td></tr>
<tr><td>3(b)
0478 8.1.1, 8.1.4(a)</td><td>The statements are executed in sequence // one after another, in the order written // line 07 is executed after line 05 / before line 08
Line 07 assigns 1 to <code>Points</code>, which overwrites / replaces the 9 // a variable stores only one value at a time</td><td>2</td><td>Do not credit “Points was changed” alone for the second mark: the answer must show the old value is replaced. Do not credit “Points is a constant”.</td></tr>
<tr><td>4
0478 8.1.1, 8.1.3 (7.8)</td><td>One mark for each line number and one mark for each matching correction:
Line 05
... <code>INPUT Stored</code> // INPUT does not take a prompt / message (the prompt is output on line 04)
Line 06
... <code>FreeSpace ← ShelfLimit - Stored</code> // a constant cannot be assigned a new value, so store the result in <code>FreeSpace</code></td><td>4</td><td>A correction mark is only awarded with the correct line number. Allow <code>FreeSpace ← 60 - Stored</code>. Allow = for ←. Do not credit line 07 as an error, or changing line 01 so that <code>ShelfLimit</code> is a variable (<code>FreeSpace</code> would still have no value). Do not credit removing line 04 instead of correcting line 05.</td></tr>
</table>

5
0478 8.1.1, 8.1.3, 8.1.4(a)
	One mark per bullet point. Max 5.
	Constant declared with the value 4.25, e.g. `CONSTANT BagPrice ← 4.25`
	Variables declared with suitable data types: number of bags INTEGER and total cost REAL
	Prompt output before the input, e.g. `OUTPUT "How many bags?"`
	Number of bags input into a variable, e.g. `INPUT Bags`
	Total cost calculated using the constant and stored, e.g. `TotalCost ← Bags * BagPrice`
	Total cost output with a message, e.g. `OUTPUT "Total cost: ", TotalCost`
	Example answer:
	`CONSTANT BagPrice ← 4.25`
	`DECLARE Bags : INTEGER`
	`DECLARE TotalCost : REAL`
	`OUTPUT "How many bags of compost do you want?"`
	`INPUT Bags`
	`TotalCost ← Bags * BagPrice`
	`OUTPUT "Total cost: ", TotalCost`
		5	max 5. Constant: must be a literal value; do not credit a constant given a variable or an expression. Declarations: both variables needed for the mark; accept REAL for the number of bags; do not credit INTEGER for the total cost, or TEXT as a data type. Allow = or <- for ←. Do not credit the prompt written inside the INPUT statement for the prompt mark. Calculation: do not credit if 4.25 is written in place of the constant. Output: message and value both needed; accept the calculation inside the OUTPUT statement for the calculation mark only if the constant is used. Program code is not credited.

Total: 22 marks