MODULE M PROGRAMMING · M16

String handling: length, substring, upper and lower case: mark scheme

22 marks · Cambridge 0478 syllabus 8.1.4(e) (string handling: length, substring, upper, lower), 8.1.2 (data types), 7.5(a) (validation), 7.7 (trace tables), 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. Quotation marks around string values are not needed. Algorithm answers must be in pseudocode; marks are for the features listed, not for one exact algorithm. Ignore minor slips in case and spacing of keywords and identifiers. Accept `<-` or `=` for assignment. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 8.1.4(e)	One mark for each correct row: • `LENGTH(Place)`: 13 • `SUBSTRING(Place, 8, 3)`: Val • `UCASE(SUBSTRING(Place, 1, 3))`: COP • `LCASE(Place)`: copper valley	**4**	Case must be exact (`val`, `VAL`, `Cop` = 0). Row 1: the space is a character, so 12 = 0. Row 2: `all` (counting from 0) = 0. Row 4: the space between the words must be kept.
2(a) 0478 8.1.4(e)	`Size ← LENGTH(Product)`	**1**	Do not accept program code (e.g. `len(Product)`, `Product.length`). Do not accept the statement written the wrong way round (`LENGTH(Product) ← Size`).
2(b) 0478 8.1.4(e)	• `SUBSTRING` used on `Product` with a length of 1, assigned to `LastChar` • start position `LENGTH(Product)` // `Size` (from part (a)) Example: `LastChar ← SUBSTRING(Product, LENGTH(Product), 1)`	**2**	A fixed start position (e.g. `SUBSTRING(Product, 9, 1)`) does not work for any length: max 1. `LENGTH(Product) - 1` as the start (wrong for position-1 strings) loses the second mark. Program code (e.g. `Product[-1]`) = 0.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr>
<td>3(a)
0478 7.7, 8.1.4(e)</td>
<td>One mark for each correct column:
• <code>Index</code>: 1, 2, 3, 4, 5
• <code>Letter</code>: t, r, u, s, t (all lower case)
• <code>Count</code>: 0, 1, 2
• OUTPUT: Number found: 2
<table>
<tr><th>INDEX</th><th>LETTER</th><th>COUNT</th><th>OUTPUT</th></tr>
<tr><td></td><td></td><td>0</td><td></td></tr>
<tr><td>1</td><td>t</td><td>1</td><td></td></tr>
<tr><td>2</td><td>r</td><td></td><td></td></tr>
<tr><td>3</td><td>u</td><td></td><td></td></tr>
<tr><td>4</td><td>s</td><td></td><td></td></tr>
<tr><td>5</td><td>t</td><td>2</td><td></td></tr>
<tr><td></td><td></td><td></td><td>Number found: 2</td></tr>
</table>
</td>
<td>4</td>
<td>Values must be in the correct order. Repeated unchanged values of <code>Count</code> (1, 1, 1) are fine. A capital <code>T</code> in the <code>Letter</code> column loses that mark. A final <code>Count</code> of 1 loses the <code>Count</code> mark; follow through to an OUTPUT of <code>Number found: 1</code> only if it matches the candidate's final Count. Quotation marks around the output are ignored; the space between <code>:</code> and 2 is not required. <code>Index</code> of 6 or 0 loses the <code>Index</code> mark.</td>
</tr>
<tr>
<td>3(b)
0478 8.1.4(e)</td>
<td>Any two from:
• the comparison on line 09 is with a lower case <code>"t"</code>
• the comparison is case-sensitive // <code>"T"</code> is not equal to <code>"t"</code>
• <code>LCASE</code> converts each extracted character to lower case // changes <code>"T"</code> to <code>"t"</code>
• so the letter is counted whether it is upper case or lower case // so the capital T at the start of <code>Trust</code> is counted // without it the count would be 1 instead of 2</td>
<td>2</td>
<td>"To make it lower case" alone = 1 (third point). Do not credit answers that say LCASE changes the value of <code>Word</code> (it does not; only <code>Letter</code> is stored in lower case).</td>
</tr>
<tr>
<td>4(a)
0478 7.8, 8.1.4(e), 8.1.2</td>
<td>One mark for each error identified and corrected:
• Line 02: <code>DECLARE Prefix : STRING</code> // CHAR should be STRING (Prefix holds two characters)
• Line 05: <code>Prefix ← UCASE(SUBSTRING(Ref, 1, 2))</code> // LCASE should be UCASE
• Line 06: <code>IF LENGTH(Ref) = 8 AND Prefix = "BK"</code> // <> should be =</td>
<td>3</td>
<td>Line number and correction are both needed for each mark. For the case error, accept keeping LCASE on line 05 and comparing with <code>"bk"</code> on line 06 instead; if a candidate makes both line 06 changes in one corrected line, award both the case and the length marks. Brackets in the condition are not required. Do not accept changes to <code>SUBSTRING(Ref, 1, 2)</code>, to lines 03, 04 or the outputs.</td>
</tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE

5(a)
0478 7.9, 8.1.4(e), 7.5(a)
	One mark per bullet point. Max 6:
	• a loop (`REPEAT ... UNTIL` // `WHILE ... DO ... ENDWHILE`) with `INPUT` of the plant name inside it
	• correct loop condition using `LENGTH`: `UNTIL LENGTH(PlantName) >= 3` // `WHILE LENGTH(PlantName) < 3`
	• a prompt before the input // an error message when the name is too short
	• `UCASE` used on the plant name (or on the extracted characters)
	• `SUBSTRING(PlantName, 1, 3)` used to extract the first three characters
	• `LENGTH(PlantName)` used for the number of characters
	• `OUTPUT` of both values with a suitable message, after the loop
	Example:
	`REPEAT`
	`  OUTPUT "Enter the plant name (at least 3 characters)"`
	`  INPUT PlantName`
	`UNTIL LENGTH(PlantName) >= 3`
	`PlantName ← UCASE(PlantName)`
	`Code ← SUBSTRING(PlantName, 1, 3)`
	`Size ← LENGTH(PlantName)`
	`OUTPUT "Label code: ", Code, " ", Size`
		6	Accept any sensible identifiers. A single IF in place of a loop (only one re-entry) does not get the loop mark but can get the condition mark. `UNTIL LENGTH(PlantName) > 3` // `WHILE LENGTH(PlantName) <= 3` (rejects a 3-character name) loses the condition mark. A WHILE loop needs an input before the loop as well as inside it for the loop mark. `SUBSTRING(PlantName, 0, 3)` loses the SUBSTRING mark (position 1 is first on this sheet). Accept a concatenation operator (e.g. &) or commas to join the output values. Program code (e.g. Python) in place of pseudocode: 0.

Total: 22 marks