CAMBRIDGE IGCSE (0478)

Module L: Standard algorithms

Workbooklet mark scheme · 3 parts · 62 marks

L2, L4, L5 and L6 are GCSE (OCR) only and not on 0478.

Contents

Part	Topic	Marks
L1	Linear search: mark scheme	21
L3	Bubble sort: mark scheme	20
L7	Totalling, counting, maximum, minimum and average: mark scheme	21
	Total	**62**

MODULE L STANDARD ALGORITHMS · L1

Linear search: mark scheme

21 marks · Cambridge 0478 syllabus 7.4 (standard methods: linear search), 7.7 (trace tables), 7.9 (write and amend 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. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.4	Any **three** from: • starts at the first element // index 1 • compares each element in turn with the value being searched for • if the element matches, the value is found // the search stops // the position is output • if it does not match, moves on to the next element • repeats until the value is found // until the end of the array is reached • if the end of the array is reached with no match, the value is not in the array // "not found" is output	**3**	"It goes through the array" alone: 0. Do not credit swapping elements (bubble sort) or checking the middle element. Credit any other valid step of a linear search.
1(b) 0478 7.4	40	**1**	Accept "all 40". Do not accept "all of them" with no number.
2(a) 0478 7.4	507, 213, 880 (in this order) ... ... then 145, and no further values	**2**	Order must be correct. Any value after 145 listed (e.g. the second 213): max 1.
2(b) 0478 7.4	4	**1**	Do not accept 3 (the array starts at index 1).
3(a) 0478 7.7, 7.4	One mark for each correct column: • Index: 1, 2, 3, 4, 5, 6 • Position: 0, 5 • OUTPUT: Found at 5	**3**	Values must be in the order shown. Ignore repeated values on later rows and the exact row each value is on. Index stops at 6: do not credit a 7 in the Index column. OUTPUT: ignore quotation marks; must be the only output.
3(b) 0478 7.7	Not found	**1**	Ignore quotation marks.
3(c) 0478 7.7	6	**1**	Every element is compared once before the loop ends with Index > 6.

QUESTION	ANSWER	MARKS	GUIDANCE

4(a)
0478 7.4, 7.9
	One mark per bullet point. Max 6:
	• input of a product code into a variable (prompt not needed)
	• index set to 1 // flag set to FALSE before the loop
	• loop through the array from 1 to 200 with the index increased inside the loop
	• comparison of `Code[Index]` with the input code
	• loop stops as soon as the code is found // condition-controlled loop using a flag or the comparison
	• output of `Price[Index]` for the matching index when found
	• suitable message output only when the code is not found (after the whole array has been checked)
	Example answer:
	`OUTPUT "Enter the product code"`
	`INPUT SearchCode`
	`Found ← FALSE`
	`Index ← 1`
	`WHILE Index <= 200 AND Found = FALSE DO`
	`  IF Code[Index] = SearchCode`
	`  THEN`
	`  Found ← TRUE`
	`  ELSE`
	`  Index ← Index + 1`
	`  ENDIF`
	`ENDWHILE`
	`IF Found = TRUE`
	`  THEN`
	`  OUTPUT "Price: ", Price[Index]`
	`  ELSE`
	`  OUTPUT "Product code not found"`
	`ENDIF`
		6	Allow REPEAT ... UNTIL or WHILE ... ENDWHILE. A FOR loop that always runs to 200 cannot get the "stops as soon as found" mark. Not-found message output inside the loop for every non-matching element: no mark for that bullet. Output of the code or the index instead of the price: no mark for that bullet. Program code (not pseudocode): 0. Ignore missing DECLARE statements.

5(a) 0478 7.9, 7.4	Any **three** from: • change the loop so that it always runs to the end of the array // remove `Position <> 0 OR` from line 10 so it is `UNTIL Index > 6` // replace the REPEAT loop with `FOR Index ← 1 TO 6` • output `Index` inside the IF (after line 07), each time a match is found • keep `Position` (or a flag / count) set when any match is found, so "Not found" is output after the loop only if there was no match • remove the ELSE output on line 15, so an index is not output twice	**3**	Each change must be explained, not just written as code. Outputting Index after line 09 (when it has already been increased): do not credit that bullet. Credit any other valid change that meets the requirement.

Total: 21 marks

MODULE L STANDARD ALGORITHMS · L3

Bubble sort: mark scheme

20 marks · Cambridge 0478 syllabus 7.4 (standard methods: bubble sort), 7.7 (trace tables), 7.8 (errors in algorithms), 7.9 (write and amend 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. Credit any other valid answer that shows the same understanding. Code answers must be in pseudocode; ignore minor syntax slips (e.g. <− or = for ←).

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.4	Any four from: Compares each pair of adjacent values // compares the first value with the second (then the second with the third, ...) If the first value is larger than the second // they are in the wrong order, they are swapped (A temporary variable is used to) swap the values Moves along the array until the end is reached // this is one pass After each pass the largest (remaining) value is at the end // in its correct position Passes are repeated ... ... until a pass makes no swaps // until the array is sorted // for (number of values − 1) passes	**4**	max 4. Credit a correct pseudocode answer that is clearly explained. Do not accept "it sorts the values" or "it puts them in order" (outcome only). Do not accept "swaps the values if they are in the right order".
2(a) 0478 7.4	After pass 1: 21 34 15 28 19 40 After pass 2: 21 15 28 19 34 40 After pass 3: 15 21 19 28 34 40	**3**	One mark per correct row. Follow through for pass 2 and pass 3 from the candidate's own previous row if it is one correct bubble sort pass of that row. The array is not sorted after pass 3 (19 and 21 are still out of order).
2(b) 0478 7.4	4	**1**	Swaps in pass 1: 34/21, 40/15, 40/28, 40/19.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>3(a)
0478 7.7</td><td>One mark for each correct column: Swapped, Index, Temp
One mark for columns Stock[1] and Stock[2] both correct
One mark for columns Stock[3] and Stock[4] both correct
<table>
<tr><th>Swapped</th><th>Index</th><th>Temp</th><th>Stock[1]</th><th>Stock[2]</th><th>Stock[3]</th><th>Stock[4]</th></tr>
<tr><td></td><td></td><td></td><td>4</td><td>9</td><td>2</td><td>7</td></tr>
<tr><td>FALSE</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>TRUE</td><td>2</td><td>9</td><td></td><td>2</td><td>9</td><td></td></tr>
<tr><td></td><td>3</td><td>9</td><td></td><td></td><td>7</td><td>9</td></tr>
<tr><td>FALSE</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>TRUE</td><td>1</td><td>4</td><td>2</td><td>4</td><td></td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>FALSE</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>2</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
Column sequences: Swapped FALSE, TRUE, FALSE, TRUE, FALSE // Index 1, 2, 3, 1, 2, 3, 1, 2, 3 // Temp 9, 9, 4 // Stock[1] 4, 2 // Stock[2] 9, 2, 4 // Stock[3] 2, 9, 7 // Stock[4] 7, 9</td><td>5</td><td>Mark each column as a sequence of values; rows may be laid out differently. Accept TRUE written again when the second swap of pass 1 is made. Accept Temp 9, 4 (second 9 not rewritten). Ignore a 4 in the Index column at the end of a pass. No further rows after the third pass: the loop stops because no swap was made. Do not credit Temp values 2 or 7.</td></tr>
<tr><td>3(b)
0478 7.9</td><td><code>IF Stock[Index] < Stock[Index + 1]</code></td><td>1</td><td>Accept <code>IF Stock[Index + 1] > Stock[Index]</code>. Do not accept <= (equal values would be swapped on every pass, so the loop would not stop if two trays held the same number). Do not accept "change > to <" with no line written.</td></tr>
<tr><td>4(a)
0478 7.8</td><td>One mark for each error identified and corrected:
Line 05: <code>FOR Position ← 1 TO 4</code> // <code>FOR Position ← 1 TO 5 - Pass</code>
Line 06: <code>IF Lengths[Position] > Lengths[Position + 1]</code>
Line 09: <code>Lengths[Position] ← Lengths[Position + 1]</code></td><td>3</td><td>Line number and correction are both needed for each mark. Line 05: <code>TO 5</code> reads <code>Lengths[6]</code>, outside the array. Line 06: < gives descending order. Line 09: stores <code>Hold</code> back in the same place so the value in <code>Lengths[Position + 1]</code> is lost. Accept <code>IF Lengths[Position + 1] < Lengths[Position]</code> for line 06. Do not credit changes to lines 04, 08 or 10 (these are correct).</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE

5(a)
0478 7.9
	One mark per bullet point:
	A new temporary variable stores the name: `TempName ← Plant[Index]`
	`Plant[Index] ← Plant[Index + 1]` **and** `Plant[Index + 1] ← TempName` (in this order, after the name is stored)
	Placed inside the IF ... THEN // between lines 08 and 13 // next to the swap of `Stock` (so names are swapped only when the numbers are swapped)
	`TempName ← Plant[Index]`
	`Plant[Index] ← Plant[Index + 1]`
	`Plant[Index + 1] ← TempName`
		3	Accept any identifier for the new variable. Reusing `Temp`: no first mark (Temp is declared INTEGER at line 02 and a plant name is a STRING), but marks 2 and 3 can still be given.
			Statements placed outside the IF (e.g. after line 13 or before line 07): no third mark. Ignore any DECLARE statement.

Total: 20 marks

MODULE L STANDARD ALGORITHMS · L7

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

21 marks · Cambridge 0478 syllabus 7.4 (standard methods: totalling, counting, maximum, minimum, average), 7.7 (trace tables), 7.8 (errors), 7.9 (write algorithms), 8.1.4(c)/(d) (iteration; totalling and counting), 8.1.7 (ROUND), 8.2 (arrays)

How to use. One mark per bullet point up to the max shown. **/** separates alternative words; **//** separates alternative answers. Words in (brackets) are not needed for the mark. Algorithms must be in pseudocode: program code is not credited. Ignore minor syntax slips (e.g. <- or = for ←, a missing THEN) if the meaning is clear. Credit any other valid method that meets the requirement.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.4, 8.1.4(d)	One mark for each correct line: `TotalWeight ← TotalWeight + Weight` → totalling `IF Weight > Heaviest ...` → finding the maximum `Parcels ← Parcels + 1` → counting `MeanWeight ← TotalWeight / Parcels` → finding the average	**4**	More than one line from a statement: 0 for that statement.
2(a) 0478 7.7, 7.4	One mark for each of: `Day` 1, 2, 3, 4, 5 and `Reading` 24, 31, 18, 30, 27 `Total` 0, 24, 55, 73, 103, 130 `Lowest` 100, 24, 18 `Count` 0, 1, 2 `Average` 26 and OUTPUT: Lowest: 18 // Hot days: 2 // Average: 26	**5**	Mark by column (or pair of columns); ignore the row a value is on if the order is correct. Initial values 0, 100, 0 must be shown for the Total, Lowest and Count marks. Count must reach 2 (30 is counted because of >=). Accept 26.0. OUTPUT: accept the values 18, 2, 26 without the message text; do not accept quotation marks around outputs.

2(a) completed table

DAY	READING	TOTAL	LOWEST	COUNT	AVERAGE	OUTPUT
		0	100	0		
1	24	24	24			
2	31	55		1		
3	18	73	18			
4	30	103		2		
5	27	130				
					26	Lowest: 18
						Hot days: 2
						Average: 26

QUESTION	ANSWER	MARKS	GUIDANCE
2(b) 0478 7.4, 7.3	Any two from: 100 is higher than any (likely) temperature reading // every reading will be lower than 100 ... so the first reading (24) replaces it // so `Lowest` is updated to a real reading If `Lowest` started at 0, no reading would be lower // `Lowest` would never change ... so the algorithm would output 0, which is not the lowest reading // the wrong answer	**2**	max 2. Credit "Lowest could instead start as the first reading input" as a valid point. Do not accept "100 is a big number" alone.
3(a) 0478 7.8, 7.4, 8.2	One mark for each error identified **and** corrected: Line 02: `Count ← 0` Line 03: `FOR Trip ← 1 TO 20` Line 04: `IF Distance[Trip] > Longest` Line 10: `Count ← Count + 1`	**4**	Line number and correction both needed for each mark. Line 04: accept >=. Line 04 also accept `IF Longest < Distance[Trip]`. Line 01 is not an error (all distances are greater than 0).

4(a)
0478 7.9, 7.4, 8.1.4(c), 8.1.4(d), 8.1.7
	One mark per bullet point, max 6:
	Initialise `Total` and `Count` to 0 (before the loop)
	Condition-controlled loop (REPEAT ... UNTIL // WHILE ... DO ... ENDWHILE) ...
	... that stops when 0 is input
	Input of the price inside the loop
	Price added to `Total` inside the loop
	`Count` increased by 1 for each price, not counting the 0
	Average calculated as `Total / Count` after the loop
	Output of the total and of the average using `ROUND(..., 2)`, with messages
	`Total ← 0`
	`Count ← 0`
	`REPEAT`
	`  OUTPUT "Enter the price, or 0 to finish"`
	`  INPUT Price`
	`  IF Price <> 0`
	`  THEN`
	`  Total ← Total + Price`
	`  Count ← Count + 1`
	`  ENDIF`
	`UNTIL Price = 0`
	`Average ← Total / Count`
	`OUTPUT "Total of prices: ", Total`
	`OUTPUT "Average price: ", ROUND(Average, 2)`
		6	Accept a WHILE loop with an input before the loop and again at the end of the loop. Accept counting the 0 and then subtracting 1 after the loop. Adding the 0 to Total is not an error. Average worked out inside the loop: do not award that bullet unless it is also correct after the last price. Not required: checking that Count is not 0. A FOR loop does not gain the loop bullets.

Total: 21 marks