MODULE M PROGRAMMING · M18

# One-dimensional arrays: mark scheme

22 marks · Cambridge 0478 syllabus 8.2.1 (declare and use 1D arrays), 8.2.2 (understand the use of arrays), 8.2.3 (write values into and read values from an array using iteration), 7.7 (trace tables), 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. Algorithm answers must be in pseudocode; marks are for the features listed, not for one exact algorithm. Ignore minor slips in case and spacing. Accept `<-` or `=` for assignment. Credit any other valid answer that shows the same understanding.

| QUESTION | ANSWER | MARKS | GUIDANCE |
|---|---|---|---|
| **1(a)** 0478 8.2.1, 8.2.2 | One mark for each correct tick:<br>• All the elements of the array have the same data type<br>• Each element is accessed using its index | **2** | More than two ticks: deduct 1 mark for each extra tick (minimum 0). |
| **1(b)(i)** 0478 8.2.1 | 24 | **1** | Do not accept 23 (the lower bound is 0, so indexes 0 to 23 give 24 elements). |
| **1(b)(ii)** 0478 8.2.1 | REAL | **1** | Accept real. Do not accept float, decimal, number or INTEGER. |
| **2(a)** 0478 8.2.1 | 4.8 | **1** | Do not accept 3.3 (counting from 0). |
| **2(b)** 0478 8.2.1 | `Weight[3] ← 1.5` | **1** | Index must be 3. Accept `Weight[3] ← Weight[3] + 0.3`. Do not accept `1.5 ← Weight[3]` or `Weight[2] ← 1.5`. Do not accept `INPUT Weight[3]` alone (the value 1.5 is not stored by the statement). |
| **3(a)** 0478 8.2.2 | Any **two** from:<br>• one identifier // one array name stores all 400 weights // no need to declare 400 separate variables<br>• a variable can be used as the index // a loop can be used to process each weight in turn<br>• the same few lines of code work for every weight // the algorithm is shorter / easier to write / easier to maintain<br>• all the weights are stored together as one group of related data | **2** | "Easier", "quicker" or "more efficient" alone, with no reason = 0. "Uses less memory" = 0 (not true in general). Credit any other valid point. |

| QUESTION | ANSWER | MARKS | GUIDANCE |
|---|---|---|---|
| **4(a)**<br>0478 7.7, 8.2.3 | One mark for each correct column (or group of columns):<br>• `Count`: 0, 1, 2<br>• `Shelf`: 1, 2, 3, 4<br>• `Loaves[2]` becomes 12 and `Loaves[3]` becomes 12, with no new values in `Loaves[1]` or `Loaves[4]`<br>• OUTPUT: 2 shelves refilled<br>Example completed table (L = Loaves):<br>`Count Shelf L[1] L[2] L[3] L[4]`<br>`  0  1  8  3  0  5`<br>`  1  2  12`<br>`  2  3  12`<br>`  4`<br>then OUTPUT (on a final row): 2 shelves refilled | **4** | Values must be in the correct order; ignore the row they are written on as long as the order is right. Ignore unchanged values copied down. Accept `Count` 0 on its own first row. OUTPUT: accept "2 shelves refilled" with or without the space after 2; do not accept quotation marks round it, or 2 alone. Any value of `Shelf` above 4: lose the `Shelf` mark. |
| **4(b)**<br>0478 8.2.3 | `Loaves[4]` holds 5 and 5 is not less than 5 // the condition `Loaves[Shelf] < 5` is FALSE when `Shelf` is 4 (so line 05 is not run) | **1** | Needs the value 5 or the FALSE condition. "It is not below 5" is acceptable. "It does not need refilling" alone = 0. |
| **5(a)**<br>0478 8.2.1, 8.2.3 | One mark per bullet point:<br>• `DECLARE Members : ARRAY[1:20] OF STRING`<br>• count-controlled loop from 1 to 20 with closing `NEXT` // `FOR Index ← 1 TO 20 ... NEXT Index`<br>• `INPUT Members[Index]` inside the loop, using the loop counter as the index<br>Example:<br>`DECLARE Members : ARRAY[1:20] OF STRING`<br>`FOR Index ← 1 TO 20`<br>`  OUTPUT "Enter name ", Index`<br>`  INPUT Members[Index]`<br>`NEXT Index` | **3** | Declaration needs the identifier, bounds 1:20 and STRING. Do not accept TEXT as the data type. Accept `INPUT Name` followed by `Members[Index] ← Name` inside the loop. Accept a REPEAT or WHILE loop with a counter that is correctly set, incremented and stops after 20 names. A loop of 0 to 19 (or 1 to 21) does not match the lower bound of 1: lose the loop mark. A prompt is not required. |

| QUESTION | ANSWER | MARKS | GUIDANCE |
|---|---|---|---|
| **6(a)**<br>0478 8.2.1, 8.2.3, 7.9 | One mark per bullet point. Max 6:<br>• input of the tool number and the number returned (into two variables)<br>• the input tool number used as the index // `Stock[Tool]`<br>• `Stock[Tool] ← Stock[Tool] + Returned` (new total stored back in the same element)<br>• count-controlled loop over every element // `FOR Index ← 1 TO 8 ... NEXT Index`<br>• `IF Stock[Index] = 0 THEN ... ENDIF` inside the loop, using the loop counter as the index<br>• `OUTPUT Index` (the tool number) inside the IF<br>Example:<br>`OUTPUT "Enter tool number (1 to 8)"`<br>`INPUT Tool`<br>`OUTPUT "Enter number returned"`<br>`INPUT Returned`<br>`Stock[Tool] ← Stock[Tool] + Returned`<br>`FOR Index ← 1 TO 8`<br>`  IF Stock[Index] = 0`<br>`  THEN`<br>`  OUTPUT "Out of stock: ", Index`<br>`  ENDIF`<br>`NEXT Index` | **6** | Prompts are not required. Accept any sensible identifiers. Outputting `Stock[Index]` (always 0) instead of the tool number loses the last mark. A loop with wrong bounds (e.g. 0 TO 8 or 1 TO 7) loses the loop mark. The update placed inside the loop, so it is repeated: lose the third mark. Accept `Stock[Index] < 1`. Do not accept program code in place of pseudocode. |

**Total: 22 marks**