MODULE M PROGRAMMING · M22

# Procedures: mark scheme

22 marks · Cambridge 0478 syllabus 8.1.6(a) (what is meant by procedures and functions), 8.1.6(b) (define and use procedures), 8.1.8 (maintainable programs), 7.7 (dry-run an algorithm), 7.8 (identify and correct errors), 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. 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)**<br>0478 8.1.6(a), 8.1.6(b) | One mark for each correct row:<br>• returns a single value to the point where it was called: FALSE<br>• can be called many times in one program: TRUE<br>• no parameters, run using CALL followed by its identifier: TRUE | **3** | Two ticks in a row: no mark for that row. Row 1 describes a function, not a procedure. |
| **2(a)**<br>0478 8.1.6(b), 7.7 | One mark for each:<br>• `Parcel labels` then `----------` as the first two lines, in that order<br>• `Parcel 1` then `Parcel 2`, with no line of dashes between them<br>• `----------` after `Parcel 2`, then `Parcel 3`<br>• `----------` as the last line (after `Parcel 3`)<br>Full output:<br>`Parcel labels`<br>`----------`<br>`Parcel 1`<br>`Parcel 2`<br>`----------`<br>`Parcel 3`<br>`----------` | **4** | Ignore the number of dashes. Ignore quotation marks round the output. Accept `Parcel1` (no space). Do not award a mark if the lines it depends on are in the wrong order or a line of dashes is added where there is none. |
| **2(b)**<br>0478 8.1.6(b) | Any **two** from:<br>• control is passed to the procedure `ShowTitle` // the procedure `ShowTitle` is called<br>• the statements in `ShowTitle` are executed // it outputs `Parcel labels` and calls `DrawLine` (which outputs a line of dashes)<br>• control then returns to the line that follows the call // returns to line 10 | **2** | "It outputs the title" alone, with no idea of the procedure being called, is too vague. Do not accept "the procedure is declared/defined on line 09". Do not accept "control returns to line 09" for the third point. |

| QUESTION | ANSWER | MARKS | GUIDANCE |
|---|---|---|---|
| **3(a)**<br>0478 8.1.6(b) | One mark for each:<br>• `PROCEDURE SafetyNotice` (header with no parameters)<br>• `OUTPUT "Keep hold of your ticket"` and `OUTPUT "Stand behind the yellow line"`, in this order, inside the procedure<br>• `ENDPROCEDURE`<br>Example:<br>`PROCEDURE SafetyNotice`<br>`  OUTPUT "Keep hold of your ticket"`<br>`  OUTPUT "Stand behind the yellow line"`<br>`ENDPROCEDURE` | **3** | Accept `PROCEDURE SafetyNotice()`. Do not award the header mark for `FUNCTION`, for `RETURNS`, or for a header with a parameter. Accept one OUTPUT statement per line or both lines in a single OUTPUT with suitable separation. Ignore small differences in the case/spacing of the notice text. Program code (e.g. Python) in place of pseudocode: 0. |
| **3(b)**<br>0478 8.1.6(b) | • `CALL SafetyNotice` | **1** | Accept `CALL SafetyNotice()`. Do not accept `SafetyNotice` without `CALL`, or an assignment such as `X ← SafetyNotice`. |
| **4(a)**<br>0478 7.8, 8.1.6(b) | One mark for each error identified **and** corrected:<br>• Line 01: `PROCEDURE Goodbye` // remove `RETURNS STRING` (a procedure does not return a value)<br>• Line 04: `ENDPROCEDURE` // `ENDFUNCTION` should be `ENDPROCEDURE`<br>• Line 11: `CALL Goodbye` // the procedure call needs `CALL` | **3** | The line number and its correction must both be right for the mark. Do not accept changing line 01 to a `FUNCTION` (no value is returned, and line 11 would then not be a valid call). Do not accept changes to lines 05 to 10, 12 or 13; the REPEAT ... UNTIL loop already stops when `Q` is entered. |

| QUESTION | ANSWER | MARKS | GUIDANCE |
|---|---|---|---|
| **5(a)**<br>0478 7.9, 8.1.6(b) | One mark per bullet point. Max 4:<br>• `PROCEDURE FinishReturn` … `ENDPROCEDURE` defined at the start, before the main algorithm<br>• procedure contains `OUTPUT "Clean the tool"` then `OUTPUT "Charge the battery"`, and no other statements<br>• `CALL FinishReturn` after `"Good condition"` **and** after `"Damage recorded"`, replacing the repeated statements<br>• rest of the algorithm kept: input, nested IF (or equivalent), the three different messages, and `FinishReturn` not called for an unrecognised code<br>Example:<br>`PROCEDURE FinishReturn`<br>`  OUTPUT "Clean the tool"`<br>`  OUTPUT "Charge the battery"`<br>`ENDPROCEDURE`<br><br>`DECLARE Condition : CHAR`<br>`INPUT Condition`<br>`IF Condition = 'G'`<br>`  THEN`<br>`  OUTPUT "Good condition"`<br>`  CALL FinishReturn`<br>`  ELSE`<br>`  IF Condition = 'D'`<br>`  THEN`<br>`  OUTPUT "Damage recorded"`<br>`  CALL FinishReturn`<br>`  ELSE`<br>`  OUTPUT "Code not recognised"`<br>`  ENDIF`<br>`ENDIF` | **4** | Accept any equivalent structure that gives the same output for every input. Call without `CALL`: do not award the third mark. Call in only one branch, or before the first message (changes the order of output): do not award the third mark. A single call after the outer `ENDIF` for every input: do not award the fourth mark. Procedure defined after the main algorithm: do not award the first mark. Do not penalise a missing `DECLARE`. |

| | | | |
|---|---|---|---|
| **5(b)**<br>0478 8.1.8 | One mark for a benefit, one mark for explaining it in this algorithm. Any **one** pair:<br>• the two statements are written only once // less repeated code ...<br>• ... so the algorithm is shorter / quicker to write (they are not typed out in both the 'G' and 'D' branches)<br>• easier to maintain / update ...<br>• ... if a finishing instruction changes, it only needs changing in one place // the 'G' and 'D' branches always give the same instructions<br>• easier to read / understand ...<br>• ... the identifier `FinishReturn` shows what the two statements do<br>• can be tested / reused ...<br>• ... it can be tested once on its own // called again if another condition code is added | **2** | The explanation mark must match the benefit given. Max 1 if the answer does not refer to this algorithm (the tools, the two branches or the finishing instructions). "Faster" or "more efficient" alone: no mark. Do not accept "the algorithm runs faster". |

**Total: 22 marks**