CAMBRIDGE IGCSE (0478)

Module O: Robust programs

Workbooklet mark scheme · 6 parts · 125 marks

O1 and O4 are GCSE (OCR) only and not on 0478.

Contents

Part	Topic	Marks
O2	Input validation: mark scheme	21
O3	Validation vs verification: mark scheme	20
O5	Why we test: testing in the coding and testing stages: mark scheme	20
O6	Finding and correcting errors in programs: mark scheme	21
O7	Choosing test data: mark scheme	21
O8	Planning tests with test data: mark scheme	22
	Total	**125**

MODULE O ROBUST PROGRAMS · O2

Input validation: mark scheme

21 marks · Cambridge 0478 syllabus 7.5(a) (need for validation checks; range, length, type, presence, format, check digit), 7.7 (trace tables), 7.8 (identify and correct errors), 7.9 (write algorithms), 8.1.4(c) (iteration), 8.1.4(e) (string handling: LENGTH, SUBSTRING)

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. Do not accept English ("is more than") or maths symbols (≥, ≤, ≠) in place of operators. Credit any other valid answer that shows the same understanding.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1(a)
0478 7.5(a)</td><td>Any two from:
• to check that the data is reasonable / sensible / follows set rules (before it is accepted)
• to stop invalid / unreasonable data being accepted // stored // processed
• invalid data could cause incorrect results / outputs // could cause the program to crash / fail
• so the user can be told the data is invalid and asked to re-enter it
• the checks are carried out automatically (by the computer)</td><td>2</td><td>Max 2. Do not accept "to check the data is correct / accurate" or "to make sure there are no errors" (validation cannot show data is correct). Do not accept answers that describe verification (checking data matches the original / was copied accurately). Credit any other valid answer.</td></tr>
<tr><td>2(a)
0478 7.5(a), 8.1.4(e)</td><td>One mark for each correct row:
• <code>Height</code> statement: range check
• <code>LENGTH(TownName)</code> statement: length check
• <code>Surname = ""</code> statement: presence check
• <code>SUBSTRING(BatchCode, 3, 1)</code> statement: format check</td><td>4</td><td>Row 4: the statement tests that the third character is a /, i.e. part of a set pattern. Do not accept presence check or type check for row 4. Do not accept "limit check" for row 1 (use the syllabus name). Spelling of the check name need not be exact if the meaning is clear.</td></tr>
<tr><td>3(a)
0478 7.7, 7.5(a), 8.1.4(c)</td><td>One mark for each:
• <code>Passengers</code> column: 0, 12, 9 (and nothing after 9)
• <code>Error: enter a number from 1 to 9</code> output twice only: once for 0 and once for 12
• <code>Booked for 9 passengers</code> output once, as the last output
<table><tr><th>PASSENGERS</th><th>OUTPUT</th></tr><tr><td>0</td><td>Error: enter a number from 1 to 9</td></tr><tr><td>12</td><td>Error: enter a number from 1 to 9</td></tr><tr><td>9</td><td></td></tr><tr><td></td><td>Booked for 9 passengers</td></tr></table></td><td>3</td><td>The value 4 is never input (9 is valid, so the UNTIL condition is TRUE and the loop ends): a <code>Passengers</code> column that includes 4 does not get the first mark, and any output for 4 loses the mark for that output. The final output may be on the same row as 9. An error message beside 9 loses the second mark. Ignore quotation marks round outputs. Accept <code>Booked for 9 passengers</code> with minor spacing differences.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
3(b) 0478 7.5(a)	• type check	**1**	Do not accept range check, presence check or format check.
4(a) 0478 7.8, 7.5(a), 8.1.4(e)	One mark for each line identified **and** corrected: • Line 04: `AND` should be `OR` // `IF (LENGTH(Surname) < 2) OR (LENGTH(Surname) > 30)` • Line 08: `> 2` should be `>= 2` // `UNTIL (LENGTH(Surname) >= 2) AND (LENGTH(Surname) <= 30)`	**2**	Line number alone, or correction alone: 0 for that error. Accept any logically equivalent correction, e.g. line 08 `LENGTH(Surname) > 1`; line 04 `NOT ((LENGTH(Surname) >= 2) AND (LENGTH(Surname) <= 30))`. Each side of a compound condition must be a full comparison. Why they are errors: with `AND` on line 04 no length can be both under 2 and over 30, so the error message is never output; with `> 2` on line 08 a valid 2-character surname is rejected.
4(b) 0478 7.5(a), 8.1.4(c)	Any one from: • the loop makes the user re-enter the surname as many times as needed // until it is valid; an IF statement would only check (and ask again) once • with only an IF, a second invalid surname would not be checked / would be accepted	**1**	"To repeat it" alone = 0 (must say what is repeated, or until when). Credit any other valid answer.
5(a) 0478 7.5(a) (check digit; see also 2.2.3)	• check digit (check)	**1**	Do not accept parity check, checksum or format check.
5(b) 0478 7.5(a)	Any one from: • the code entered still has 7 digits / characters // the correct number of characters • a length check only checks the number of characters, not what the digits are / their order	**1**	"It is the right length" is enough. "A length check cannot detect errors" alone = 0.

QUESTION	ANSWER	MARKS	GUIDANCE

6(a)
0478 7.9, 7.5(a), 8.1.4(c), 8.1.4(e)
	One mark per bullet point. Max 6:
	• prompt (OUTPUT) and INPUT of the asset tag
	• condition-controlled loop (REPEAT ... UNTIL // WHILE ... DO ... ENDWHILE) with the input inside the loop, so the tag is re-entered
	• length check: `LENGTH(Tag) = 7` (valid) // `LENGTH(Tag) <> 7` (invalid)
	• format check on the first character: `SUBSTRING(Tag, 1, 1) = "H"` (valid) // `<> "H"` (invalid)
	• the two checks combined correctly in the loop condition: AND for valid (e.g. UNTIL) // OR for invalid (e.g. WHILE)
	• error message output only when the tag is invalid, and `Tag accepted` output once, after the loop
	Example:
	`REPEAT`
	`  OUTPUT "Enter the asset tag"`
	`  INPUT Tag`
	`  IF (LENGTH(Tag) <> 7) OR (SUBSTRING(Tag, 1, 1) <> "H")`
	`  THEN`
	`  OUTPUT "Invalid tag: 7 characters, starting H"`
	`  ENDIF`
	`UNTIL (LENGTH(Tag) = 7) AND (SUBSTRING(Tag, 1, 1) = "H")`
	`OUTPUT "Tag accepted"`
		6	Accept `'H'` for `"H"`. Accept any meaningful identifier for the tag. Accept separate IF statements with separate error messages, or a Boolean flag (e.g. `Valid`) set by the checks and used in the loop condition. Each side of a compound condition must be a full comparison. AND joining the two invalid tests // OR joining the two valid tests loses the fifth mark. A single IF with no loop (only one re-entry): max 4 (prompt and input, length check, format check, messages). Credit a correct use of `UCASE` to also allow a lower-case h, but it is not required. Program code instead of pseudocode: 0.

Total: 21 marks

MODULE O ROBUST PROGRAMS · O3

Validation vs verification: mark scheme

20 marks · Cambridge 0478 syllabus 7.5(b) (the need for verification checks; visual check and double entry check, their purpose and algorithms), 7.5(a) (validation checks), 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. 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. Do not accept English ("is not equal to") or maths symbols (≠) in place of operators. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.5(a), 7.5(b)	One mark for each correct row: • compared by eye with the paper form: verification • date only accepted in the form DD/MM/YYYY: validation • data entered is the same as the data meant to be entered: verification • quantity rejected if not a whole number: validation	**4**	A row with both boxes ticked = 0 for that row. Accept a cross or other clear mark in place of a tick.
2(a) 0478 7.5(a), 7.5(b)	Any two from: • 150 is within the range 1 to 500 // 150 is a valid / reasonable value // 150 meets the rule, so it is accepted • a range check / validation only checks that the data is sensible / within the limits / follows the rules • it does not check that the value is correct // that it matches the paper order // that it is the value that was intended (105)	**2**	Max 2. "The range check is wrong / too big" alone = 0. Do not accept "150 is outside the range" (wrong). Credit any other valid answer.
2(b) 0478 7.5(b)	Any two from: • the quantity that was entered (150) is shown / displayed on the screen • the clerk compares / reads it against the original paper order (by eye) • the clerk sees that 150 does not match 105 // sees the two values are different, so does not confirm it and re-enters / corrects it	**2**	Max 2. The comparison must be with the original (paper order / source document) for that mark. "The clerk checks it" alone = 0. Do not credit a description of a double entry check (entering the data twice) or of a validation check.

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) 0478 7.7, 7.5(b)	One mark for each correct column: • `Entry1`: 58204, 58204, 58204 • `Entry2`: 58240, 52804, 58204 • `Attempts`: 0, 1, 2, 3 • OUTPUT: `Entries do not match, start again` (twice, after attempts 1 and 2 only), then `Account number 58204 accepted after 3 attempts` (once, at the end)	**4**	Accept 58204 written once in the `Entry1` column (it is input again but does not change). Any value in a column after the loop has ended loses the mark for that column. A mismatch message on the third pass loses the OUTPUT mark. The final output may be on the same row as the last values or on the next row. Ignore quotation marks round outputs. Accept small spacing differences in the final output.

4(a)
0478 7.9, 7.5(b)
	One mark per bullet point, max 5:
	• condition-controlled loop (e.g. `REPEAT` ... `UNTIL`) around the input, display and reply steps
	• prompt (OUTPUT) and `INPUT` of the registration, inside the loop
	• OUTPUT of the registration that was entered (the variable, not its name in quotes), after it is input
	• prompt asking whether it matches the booking form (Y or N) and `INPUT` of the reply, after the registration is output
	• loop ends only when the reply is Y: `UNTIL Reply = 'Y'`
	• message that the registration has been saved, output once, after the loop
	Example:
	`REPEAT`
	`  OUTPUT "Enter the car registration"`
	`  INPUT Registration`
	`  OUTPUT "You entered ", Registration`
	`  OUTPUT "Does this match the booking form? (Y or N)"`
	`  INPUT Reply`
	`UNTIL Reply = 'Y'`
	`OUTPUT "Registration saved"`
		5	Accept `"Y"` for `'Y'`; accept `UNTIL Reply = 'Y' OR Reply = 'y'` // `UNTIL UCASE(Reply) = 'Y'`. A WHILE loop that inputs the registration and reply before the loop and again inside it can get full marks. `UNTIL Reply = 'N'` or `UNTIL Reply <> 'Y'` (reversed) loses the condition mark. A single IF with no loop (only one re-entry): max 3. Do not penalise extra validation, but it earns no marks. Program code instead of pseudocode: 0.

QUESTION	ANSWER	MARKS	GUIDANCE
5(a) 0478 7.8, 7.5(b)	One mark for each line identified **and** correctly corrected: • line 05: `INPUT ConfirmPin` • line 06: `IF NewPin <> ConfirmPin` // `IF NOT (NewPin = ConfirmPin)` • line 10: `UNTIL NewPin = ConfirmPin` // `UNTIL NOT (NewPin <> ConfirmPin)`	**3**	A line number with no correction, or a correction with no line number = 0. Line 05: `ConfirmPin ← INPUT` is not pseudocode (0); do not accept deleting the line without the INPUT. Line 06: do not accept changing the message on line 08 instead (the mismatch message would then never be output). Line 10: `UNTIL ConfirmPin = NewPin` is fine. Ignore extra lines given if the three errors are found, but a correct line "corrected" into an error loses one mark.

Total: 20 marks

MODULE O ROBUST PROGRAMS · O5

Why we test: testing in the coding and testing stages: mark scheme

20 marks · Cambridge 0478 syllabus (exams 2026-2028) 7.1 (program development life cycle: coding = writing program code and iterative testing; testing = testing program code with the use of test data), 7.6 (suggest and apply suitable test data: normal, abnormal, extreme, boundary), 7.8 (identify errors in given algorithms and suggest ways of correcting these 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. Use the syllabus stage names (analysis, design, coding, testing); "final testing" or "terminal testing" used to mean the testing stage may be credited where the description is correct.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.1	One mark for each correct row: Cost module tested on its own as soon as it is written: Coding Complete program run with normal, abnormal, extreme and boundary data: Testing Error corrected in the receipt module and the module tested again: Coding Complete program checked against all the requirements: Testing	**4**	A row with both boxes ticked scores 0.
2(a) 0478 7.1	Iterative (testing)	**1**	Do not credit "unit testing", "module testing" or "dry run": the syllabus term is required.
3(a) 0478 7.1, 7.6	One mark for each: Test 2: 15 Test 3: 21	**2**	Accept $15 // 15.00 and $21 // 21.00. The expected result comes from the company's charges, not from the code: do not credit 20 for Test 2.
3(b) 0478 7.1	Test 2 // the test with 5 hours	**1**	Mark on its own: no follow through from 3(a). More than one test named scores 0.
3(c) 0478 7.8	One mark for each: Line 03 `IF Hours >= 5` // `IF Hours > 4`	**2**	The two marks are independent. Accept the correction written in words only if it is precise, e.g. "change > to >=" // "change 5 to 4". Accept `IF 5 <= Hours`. Do not credit `IF Hours = 5`, "=>", "≥" or a change to line 05 or 07.

QUESTION	ANSWER	MARKS	GUIDANCE
3(d) 0478 7.1	Tests the function / module again // runs the (three) tests again (to check that the error has been corrected / no new error has been made)	**1**	Do not credit "moves on to the testing stage" or "writes the next module" alone: the module must be tested again first.
4(a) 0478 7.6, 7.1	One mark for each correct row (value **and** accept/reject both correct): Normal: any whole number from 1 to 8, e.g. 3 // Accept Extreme: 1 // 8 // Accept Abnormal: any value that is not a whole number from 1 to 8, e.g. 0 // 9 // −2 // 15 // 2.5 // "six" // Reject	**3**	A row with only one of the two cells correct scores 0 for that row. Normal: values 2 to 7 are the best answers; also accept 1 or 8 (they should be accepted). Extreme: only 1 or 8. Do not credit 0 or 9 as extreme data (they are rejected values). Abnormal: credit any value that should be rejected, including 0 or 9.
4(b) 0478 7.1	Any two from: Compares the actual result / outcome of each test with the expected result / outcome If they are the same, the test is passed // the program works as expected for that data If they are different / a test fails, the error is found and corrected (in the program code) ... and the test(s) are run again Records the results (to show the program meets the requirements)	**2**	max 2. Do not credit "checks it works" or "looks at the results" alone.
5(a) 0478 7.1	Any four from: **Coding stage** Testing is carried out while the program code is being written Each module is tested separately / on its own (e.g. the cost module / `HireCost` function) Errors found are corrected and the module is tested again (iterative testing) **Testing stage** Testing is carried out when the program is complete / after coding has finished The complete / whole program is tested // all the modules are tested together (e.g. hours check, cost and receipt modules working together) Uses (a set of) test data // normal, abnormal, extreme and boundary data (e.g. 0, 1, 4, 8 and 9 hours) Checks that the program meets (all) the requirements (from the analysis stage)	**4**	max 4. Max 3 if only one stage is described. Max 3 if the answer makes no reference to the bike-hire program. Credit a correct example in context once per stage. Do not credit "checks for errors" alone, which is true of both stages.

Total: 20 marks

MODULE O ROBUST PROGRAMS · O6

Finding and correcting errors in programs: mark scheme

21 marks · Cambridge 0478 syllabus 7.8 (identify errors in given algorithms and suggest ways of correcting these errors), 7.7 (complete a trace table to document a dry-run of an algorithm), 7.6 (suggest and apply suitable test data) and 4.2.3 (how errors are reported by a compiler and an interpreter)

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. Corrections should be in pseudocode; ignore minor slips (e.g. <− or = for ←, missing `THEN`) as long as the logic is clear, unless the slip is the error being corrected. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 4.2.3	One mark per correct row: Lines 1 to 7 run before any error is reported: Interpreter One error report lists both errors: Compiler Line 20 error only reported after line 8 is corrected: Interpreter None of the program can be run until both errors are corrected: Compiler	**4**	A row with both boxes ticked scores 0 for that row. An interpreter translates and executes line by line and stops when it reaches an error; a compiler translates the whole code first and gives an error report for the whole code, and no executable file is produced until the errors are corrected.

2(a)
0478 7.8
	One mark for each correct statement:
	`Count ← Count + 1`
	`FOR Hour ← 1 TO 24` // `FOR Hour ← 0 TO 23`
	`IF Length >= 2 AND Length <= 5`
	`Prefix ← SUBSTRING(Code, 1, 3)`
		4	Row 3: accept brackets round each comparison // `2 <= Length AND Length <= 5`. Do not accept `OR`, ≥/≤, or a second comparison with no variable.
			Row 4: the start position must be 1 and the length must be 3. Do not accept `LEFT` or program code.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>3(a)
0478 7.8</td><td>One mark for each error identified and corrected:
Line 09 // <code>UNTIL Weight >= 500 AND Weight <= 4000</code>
Line 13 // <code>ENDWHILE</code> should be <code>ENDIF</code>
Line 14 // <code>NEXT Weight</code> should be <code>NEXT Vehicle</code>
Line 15 // <code>OUTPUT "Vehicles over 3000 kg: ", Heavy</code> // output Heavy instead of Weight</td><td>4</td><td>max 4. The line number alone, or a correction with no way of telling which line it replaces, scores 0.
Line 09: accept <code>UNTIL NOT (Weight < 500 OR Weight > 4000)</code> // <code>Weight > 499 AND Weight < 4001</code> (weights are whole numbers). Do not accept “change OR to AND” if the comparisons are also changed wrongly.
Do not credit changes to lines 04, 10 or 12 (they are correct).</td></tr>
<tr><td>3(b)
0478 7.6, 7.8</td><td>One mark for each:
Weight: any whole number less than 500 or greater than 4000, e.g. 100 // 499 // 4001 // 6000
Type: abnormal (for any value outside the range) // boundary (for 499 or 4001 only)
Expected result: the weight is rejected // an error message is shown and the weight is asked for again</td><td>3</td><td>The type mark must match the weight given: 100 or 6000 is abnormal, not boundary.
Do not accept “extreme” (extreme data is the largest/smallest acceptable value) or “normal”.
A weight from 500 to 4000 does not show the error: it scores 0 for all three marks.
Do not accept “the weight is accepted” as the expected result (that is what the faulty algorithm does).</td></tr>
<tr><td>4(a)
0478 7.7</td><td>One mark per correct column:
Index: 1, 2, 3, 4, 5
Count: 0, 3, 7, 9
OUTPUT: Low rainfall days: 9
<table><tr><th>INDEX</th><th>COUNT</th><th>OUTPUT</th></tr><tr><td></td><td>0</td><td></td></tr><tr><td>1</td><td>3</td><td></td></tr><tr><td>2</td><td></td><td></td></tr><tr><td>3</td><td>7</td><td></td></tr><tr><td>4</td><td></td><td></td></tr><tr><td>5</td><td>9</td><td></td></tr><tr><td></td><td></td><td>Low rainfall days: 9</td></tr></table></td><td>3</td><td>Allow Count values repeated on rows where they do not change.
Allow the starting 0 to be missing if 3, 7, 9 are correct.
OUTPUT: accept 9 alone; ignore quotation marks. Do not accept 3 (the trace must follow the algorithm as written).
Do not penalise a final 6 in the Index column.</td></tr>
<tr><td>4(b)
0478 7.7</td><td>Low rainfall days: 3</td><td>1</td><td>Accept 3 alone.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
4(c) 0478 7.8	Line 05 (identified) … `Count ← Count + 1`	**2**	The correction mark may be given if the line number is missing but the statement makes clear which line is meant. Do not credit changes to line 03 (it is correct).

Total: 21 marks

MODULE O ROBUST PROGRAMS · O7

Choosing test data: mark scheme

21 marks · Cambridge 0478 syllabus 7.6 (suggest and apply suitable test data: normal, abnormal, extreme, boundary) and 7.8 (identify errors in given algorithms and suggest ways of correcting these 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. Test data must be an actual value; a description such as "a number between 1 and 20" does not score. Corrections should be in pseudocode; ignore minor slips (e.g. missing `THEN`) as long as the logic is clear. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) 0478 7.6	Data that is within the acceptable range / that should be accepted (by the program)	**1**	Do not accept an example on its own.
1(b) 0478 7.6	Extreme: the largest / smallest acceptable value (one value, which is accepted) Boundary: the largest / smallest acceptable value **and** the corresponding smallest / largest rejected value // a pair of values, one accepted and one (just outside the range) rejected	**2**	Accept a correct example used to support each point, e.g. range 1 to 100: extreme 100 // boundary 100 and 101. Do not accept "boundary data is at the edge of the range" on its own: the rejected value is needed.
2(a) 0478 7.6	One mark per correct row (type **and** accept/reject both correct): 4: normal, accept 9: extreme, accept 15: abnormal, reject four: abnormal, reject 0: abnormal, reject	**5**	Row 2 (9): accept extreme / boundary / normal (9 is the largest acceptable value, so it is also within the normal range). Row 5 (0): accept abnormal / boundary (0 is the rejected value that pairs with 1). Rows 3 and 4: do not accept boundary or extreme. Accept "accepted / valid" and "rejected / invalid / error message".
3(a) 0478 7.6	8 … accepted / hire booked 9 … rejected / error message output / asked to enter again	**2**	Each mark needs the value **and** its expected outcome. The outcome must be what **should** happen by the rule (8 accepted), not what this algorithm does. Do not credit values for the lower limit (1 and 0).
3(b) 0478 7.6, 7.8	8 is rejected // "Invalid: enter 1 to 8" is output for 8 and the hire length is asked for again (9 is correctly rejected) This shows there is an error at the upper limit // 8 should be accepted // the conditions on line 05 / line 09 are wrong	**2**	FT from (a) only for the value 8. If (a) gave only lower-limit values, accept an answer that correctly describes 8 being rejected. Do not accept "the program crashes".

QUESTION	ANSWER	MARKS	GUIDANCE
3(c) 0478 7.8	One mark for each line identified **and** corrected: Line 05 // `IF Hours < 1 OR Hours > 8` Line 09 // `UNTIL Hours >= 1 AND Hours <= 8`	**2**	Line 05: accept `Hours >= 9` // `8 < Hours`. Line 09: accept `Hours < 9` // `Hours > 0` for the first condition. The line number alone, or the correction alone with no way of telling which line it replaces, scores 0. Do not accept ≥ or ≤ symbols.
4(a) 0478 7.6	Extreme: a password of exactly 10 characters // exactly 16 characters, e.g. `TreeHouse1` // `QuietLake2025xyz` Reason: it is the shortest / longest length that should be accepted Abnormal: a password of 9 or fewer // 17 or more characters, e.g. `TreeHouse` // an empty password Reason: it is outside the range 10 to 16 characters / too short / too long, so it should be rejected	**4**	Count the characters; any characters may be used. Each reason mark must match its example. A reason with no correct example scores 0. Accept a clearly stated length with an example of that length. A length alone with no password (e.g. "10 characters") does not score the example mark.
5(a) 0478 7.6	Any three from: All three values are normal data // it only checks that data in the range is accepted Extreme data should be tested, e.g. 1 // 20 (to check the limits are accepted) Boundary data should be tested, e.g. 20 and 21 // 1 and 0 Abnormal data should be tested, e.g. 35 // −4 // ten (to check it is rejected) Errors at the limits of the range / in the validation would not be found	**3**	max 3. Mark points 2 to 4 each need a correct example for that type. Boundary needs the pair (one accepted, one rejected value). A single value of 1 or 20 credits mark point 2; a single value of 0 or 21 credits mark point 4.

Total: 21 marks

MODULE O ROBUST PROGRAMS · O8

Planning tests with test data: mark scheme

22 marks · Cambridge 0478 syllabus 7.6 suggest and apply suitable test data (normal, abnormal, extreme, boundary); 7.1 testing stage of the program development life cycle; 7.8 identify errors and suggest corrections

How to use. One mark per bullet point unless stated, up to the max shown. **//** separates alternative answers for the same mark. Words in (brackets) are not needed for the mark. **Test data** must be an actual value, not a description ("a number in range" = 0). Use the 0478 definitions: **extreme** = the largest/smallest acceptable value; **boundary** = the largest/smallest acceptable value **and** the corresponding smallest/largest rejected value. Credit any other valid answer.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>1
0478 7.6</td><td>One mark for each correct row:
<table><tr><th>Test data</th><th>Normal</th><th>Abnormal</th><th>Extreme</th><th>Boundary</th></tr><tr><td>25</td><td>✓</td><td></td><td></td><td></td></tr><tr><td>40</td><td></td><td></td><td>✓</td><td>✓</td></tr><tr><td>9</td><td></td><td>✓</td><td></td><td>✓</td></tr><tr><td>forty</td><td></td><td>✓</td><td></td><td></td></tr></table></td><td>4</td><td>Each row must have every required tick and no wrong tick.
Row 40: Extreme and Boundary both needed (largest acceptable value). Allow Normal also ticked (40 is accepted data); Abnormal ticked = 0.
Row 9: Abnormal and Boundary both needed (it is the rejected value next to 10). Extreme or Normal ticked = 0.
Rows 25 and forty: one tick only.</td></tr>
<tr><td>2
0478 7.6</td><td>Test 2: 20 // 45
Test 3: Abnormal // Boundary
Test 4 data: 19 and 20 // 45 and 46
Test 4 outcome: 19 rejected (<code>Pressure out of range</code>) and 20 accepted // 45 accepted and 46 rejected (<code>Pressure out of range</code>)
Test 5: any value that should be rejected, e.g. <code>thirty</code> // 50 // −5 // 18 // 25.5</td><td>5</td><td>One mark per bullet point.
Test 2: 20 or 45 only.
Test 3: accept abnormal or boundary (46 is the rejected value of the 45/46 boundary pair). Do not accept extreme / normal.
Test 4: the two values must be a pair across one edge (one accepted, one rejected). The outcome mark needs the correct outcome for each value; FT the outcome for a wrong pair only if each value's outcome is correct for this check. The message does not need to be repeated.
Test 5: 46 and 19 are also abnormal; accept them. Do not accept a value from 20 to 45.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
3 0478 7.6	A length from 2.0 to 5.0 (e.g. 3.5) … `Fare is 30` A length above 5.0 and up to 12.0 (e.g. 8.0) … `Fare is 55` A length above 12.0 and up to 20.0 (e.g. 15.0) … `Fare is 90`	**3**	One mark per row: the length must be in the correct range **and** the output must match it. Rows in any order; each fare tested once for its mark. Accept 5.0 → 30 and 12.0 → 55. Accept the output as 30 / 55 / 90 alone. Do not accept a length below 2.0 or above 20.0 (it never reaches this algorithm).
4(a) 0478 7.6	Test 2: Rejected // asked to enter the number of seats again Test 3: Rejected // asked to enter the number of seats again Test 4: Accepted // outputs `Booking for 8 seats`	**3**	One mark per row. Test 2 must follow the algorithm as written: 1 > 1 is FALSE, so the loop repeats. "Accepted" for test 2 = 0. Accept "loop repeats" / "prompt shown again" for rejected.
4(b) 0478 7.8	Line 04 `UNTIL Seats >= 1 AND Seats <= 8` // change `Seats > 1` to `Seats >= 1`	**2**	One mark for the line number, one for a correct correction. Accept `Seats > 0` (whole number of seats) and `1 <= Seats`. Accept a correct description, e.g. "change > to >=". Do not accept ≥ or =>. Do not accept a correction that also lets 0 or 9 through.
4(c) 0478 7.6	4 is normal data // 4 is inside the range, away from the edge, so it is accepted whether the test is > or >= The error only affects the smallest acceptable value // only extreme / boundary data (1) finds it	**1**	Any one point. Do not accept "test 1 passed" alone (NE: needs why).
5 0478 7.6	No extreme data … add a password of exactly 8 // exactly 20 characters, expected to be accepted No boundary data … add a pair such as 7 and 8 characters (rejected, accepted) // 20 and 21 characters (accepted, rejected) No abnormal data that is too long … add a password of more than 20 characters, e.g. 25 characters, expected to be rejected Expected outcome "OK" (test 2) is vague // outcomes do not say what happens … write "Accepted" // "Rejected, error message, re-enter password" Tests 1 and 2 test the same thing (both normal) … replace one with extreme / boundary / long abnormal data	**4**	Mark in pairs: one mark per problem and one for a matching change or extra test, max two pairs. Award a problem on its own if the change is missing or wrong. For a length check, the test data may be given as an actual string or as an exact number of characters (e.g. "a 20-character password"); "a long password" = 0. Do not accept "not enough tests" alone. Do not award the same problem twice.

Total: 22 marks