OCR GCSE (J277)

Module O: Robust programs

Workbooklet mark scheme · 7 parts · 149 marks

O3 is IGCSE (Cambridge) only and not on J277.

Contents

Part	Topic	Marks
O1	Defensive design: anticipating misuse: mark scheme	21
O2	Input validation: mark scheme	22
O4	Authentication in programs: mark scheme	22
O5	Why we test, and iterative vs final testing: mark scheme	20
O6	Syntax errors vs logic errors: mark scheme	22
O7	Choosing test data: mark scheme	21
O8	Writing a test plan: mark scheme	21
	Total	**149**

MODULE O ROBUST PROGRAMS · O1

Defensive design: anticipating misuse: mark scheme

21 marks · OCR J277 specification 2.3.1 (defensive design considerations: anticipating misuse; issues a programmer should consider to ensure a program caters for all likely input values; how to deal with invalid data in a program; practical experience of designing input validation), with 2.2.3 (arrays) in Q4

How to use. 1 mark per bullet up to the maximum shown. **/** separates alternative words or answers. Words in (brackets) are not needed for the mark. **...** means the mark depends on the one before. Code may be in OCR Exam Reference Language or any high-level language: mark the logic, and do not penalise minor syntax slips (e.g. a missing `then`, `endif`, colon or bracket); penalise a repeated slip once, then follow through. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.3.1	Max 2 from: Thinking ahead about / predicting how users might use the program wrongly / enter unexpected / invalid data Considering all likely input values (including invalid ones) ... and designing / writing the program so that it caters for / handles them (e.g. rejects them, does not crash)	**2**	The third bullet can be given with either of the first two. Allow misuse that is accidental (mistakes) or deliberate (e.g. trying to break in / hack). Do not allow "stopping people misusing the program" on its own with no idea of planning for it. Do not allow answers about the Computer Misuse Act.
1(b) J277 2.3.1	1 mark for a method, 1 mark for the expansion. For example: Limit the number of password attempts ... ... e.g. lock the account / stop logins after 3 wrong attempts / until an administrator unlocks it OR Add a delay after each wrong attempt ... ... so guessing many passwords takes too long OR Ask for a second factor / code (sent to the staff member) ... ... so a guessed password alone does not give access	**2**	Allow any number of attempts as the example. Allow "require strong / long passwords" ... "so they are harder to guess" for 2. Allow "do not say whether the username or the password was wrong" ... "so the attacker learns less from each guess" for 2. Do not allow "use a firewall / anti-malware" or "encrypt the password": they do not stop repeated guessing.

QUESTION	ANSWER	MARKS	GUIDANCE
2(a) J277 2.3.1	1 mark per row: **Hours:** any value outside 1 to 8, e.g. 0 / 9 / −2 // a non-whole number, e.g. 2.5 // text, e.g. "three" // left blank **Type of bike:** any choice other than 1, 2 or 3, e.g. 4 / 0 // text, e.g. "tandem" / "E" // left blank **Surname:** left blank / nothing entered // far too long, e.g. 500 characters // only digits or symbols, e.g. "12345" / "??"	**3**	The answer must be an actual value or a clear description of one ("nothing entered", "a 500-letter name"). Do not allow vague answers such as "a wrong number", "invalid data" or "a mistake". Do not allow valid values (1 to 8 for hours; 1, 2 or 3 for bike type; a normal surname). Surname: do not allow a surname with a hyphen, apostrophe or space (e.g. O'Neil, Smith-Jones): these are real surnames the program should accept.
2(b) J277 2.3.1	Max 3 from: Check / validate the input (before it is used) Reject the value / do not accept / do not store / do not use it Output an error message (that says what is wrong / what is allowed, e.g. "Enter 1 to 8") Ask the customer to enter the value again / re-prompt Repeat (in a loop) until a valid value is entered The program should not crash / should keep running	**3**	Do not allow "it ignores it" on its own, or "the program stops / closes". Do not allow answers about choosing test data: this question is about how the running program deals with the value.
3(a) J277 2.3.1	0	**1**	With `count` = 0 the loop does not run and line 07 divides by 0.
3(b) J277 2.3.1	Any one from: A negative number, e.g. −3 Text / a non-number, e.g. "five" A decimal / real number, e.g. 2.5 Nothing / left blank	**1**	Do not allow 0 (already given in (a)). Allow a very large number, e.g. 1 000 000, if the answer says it is unrealistic for the number of readings. Do not allow a positive whole number on its own (e.g. 4).

QUESTION	ANSWER	MARKS	GUIDANCE

3(c)
J277 2.3.1
	1 mark each:
	Input of `count` (stored as a whole number)
	Attempt at a loop that repeats the input
	... with a correct condition: repeats while `count < 1` / `count <= 0` // until `count >= 1` / `count > 0`
	Error message output AND count input again inside the loop
	Example answer:
	`count = int(input("How many?"))`
	`while count < 1`
	`  print("Must be 1 or more")`
	`  count = int(input("How many?"))`
	`endwhile`
	Alternative:
	`do`
	`  count = int(input("How many?"))`
	`  if count < 1 then`
	`  print("Must be 1 or more")`
	`  endif`
	`until count >= 1`
		4	A single `if` with one re-input (no loop) cannot get marks 2 or 3; it can get marks 1 and 4.
			Allow `count = input(...)` without `int()` (the question says assume a whole number).
			Do not allow ≤, ≥ or => written as maths symbols.
			Do not allow `count <= 1` / `count > 1` (these reject 1, which is valid).
			In a do-until answer, the error message must only show when the value is invalid.

4(a) J277 2.3.1, 2.2.3	Max 2 from: `slot - 1` is 6 / the program looks for `snacks[6]` The array only has indexes 0 to 5 / there is no element at index 6 / the index is outside the array (out of range) The value 7 was not anticipated / not checked before it was used	**2**	Allow "there are only 6 items in the array" for the second bullet. Do not allow "7 is not a valid number" on its own.

4(b)
J277 2.3.1, 2.2.3
	1 mark each:
	Attempt at selection (`if`) checking the value of `slot`
	... correct condition for a valid slot: `slot >= 1 AND slot <= 6`
	Outputs `"You chose " + snacks[slot - 1]` when valid AND an error message otherwise
	Example answer:
	`if slot >= 1 AND slot <= 6 then`
	`  print("You chose " + snacks[slot - 1])`
	`else`
	`  print("Invalid slot")`
	`endif`
		3	Allow equivalent conditions: `slot > 0 AND slot < 7`; `slot - 1 >= 0 AND slot - 1 <= 5`; or the reverse test `slot < 1 OR slot > 6` with the outputs swapped.
			Do not allow a condition that checks only one end (e.g. `slot <= 6`): it still lets 0 through. Do not allow `slot >= 1 AND <= 6` (second half has no variable).
			Allow a loop that re-asks for the slot until it is valid, with an error message, followed by the output line, for all 3 marks.

Total: 21 marks

MODULE O ROBUST PROGRAMS · O2

Input validation: mark scheme

22 marks · OCR J277 spec 2.3.1 (defensive design: input validation; how to deal with invalid data; practical experience of designing input validation), with 2.3.2 (logic errors), 2.2.3 (string length) and 2.2.1 (iteration)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. Code answers are marked for logic, not exact syntax: accept OCR Exam Reference Language or any high-level language, and allow any logically valid method. Minor syntax slips are not penalised; a repeated slip (e.g. = for comparison) is penalised once then follow through. Capitals in keywords are ignored. Do not accept maths symbols (≥, ≤, ≠) or English ("is more than") in place of operators. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.3.1	Checks data is sensible / follows set rules before it is accepted: True If an input passes validation, it must be correct: False A type check could reject `six` when a whole number is expected: True	**3**	1 mark per correct row. Both boxes ticked in a row: 0 for that row. Row 2 is False because validation only checks that data is reasonable / follows the rules: a valid value can still be the wrong value (e.g. 5 entered instead of 7).
2(a) J277 2.3.1	Member number: format check Number of days: range check Contact phone number: length check Name of the tool: presence check	**4**	1 mark per row. Accept a clear description of the check instead of its name (e.g. "checks it is between 1 and 14" for row 2; "checks it is not empty" for row 4). Row 2: accept "range check and type check"; "type check" alone = 0 (it does not test the limits 1 to 14). Row 3: do not accept range check; "format check" alone = 0 (the rule is only about the number of characters). Accept "pattern check" for row 1.
2(b) J277 2.3.1	1 mark each to max 2: Reject the value / do not accept, store or use 20 (it fails the range check) Output an error message (that tells the member what is wrong, e.g. the number must be from 1 to 14) Ask the member to enter the number of days again / re-prompt ... ... and keep asking (in a loop) until a value from 1 to 14 is entered The program should not crash / should carry on running	**2**	Max 2. The fourth bullet depends on re-input being stated or clearly implied. "Validate it" / "check it" alone = 0. Do not accept "change it to 14" / "round it down" (the program must not silently change the input).
3(a) J277 2.3.1, 2.3.2	(Line) 02	**1**	Accept 2.

QUESTION	ANSWER	MARKS	GUIDANCE
3(b) J277 2.3.1, 2.3.2	`Booking for 9 players`	**1**	Working: no number can be both less than 2 and greater than 6, so the `while` condition is always False; the loop body never runs and 9 is accepted. Ignore quotation marks. Do not accept `Team size must be 2 to 6` (alone or with other output).
3(c) J277 2.3.1, 2.3.2	`while players < 2 OR players > 6`	**1**	Accept any logically equivalent condition, e.g. `while players <= 1 OR players >= 7` / `while NOT(players >= 2 AND players <= 6)`. Each side must be a full comparison: `while players < 2 OR > 6 = 0`. Allow FT only if 3(a) identified line 02 or the line is clearly the loop condition.
3(d) J277 2.3.1	1 mark for any one of: So the user is asked again / can enter a new value each time the number is invalid So the new value is checked again (by the loop condition) before it is accepted Without it, the value would never change, so an invalid value would make the loop repeat forever / an infinite loop	**1**	"To get an input" alone = 0 (must link to repeating / re-checking).
4(a) J277 2.3.1, 2.2.3	73: `Code must be 4 characters` 73915: `Code must be 4 characters` 7391: `Locker set` (only)	**3**	1 mark per row. Ignore quotation marks round outputs. Do not award a row if `print(...)` is written round the output. Row 3: `Code must be 4 characters` as well as / instead of `Locker set` = 0 (the length is 4, so the `if` is False and the loop ends). Working: lengths are 2, 5 and 4.

QUESTION	ANSWER	MARKS	GUIDANCE

5(a)
J277 2.3.1, 2.2.3, 2.2.1
	1 mark each to max 6:
	Inputs the username and stores it in a variable
	Presence check: tests whether the username is blank, e.g. `username == ""` / `username.length == 0` / `username.length < 1`
	Length check: tests the length correctly, e.g. `username.length > 12` (invalid) / `username.length <= 12` (valid)
	Attempt at a condition-controlled loop (`while` or `do ... until`) ...
	... that repeats while the username is blank OR too long / until it is not blank AND not too long, with the input (re-prompt) inside the loop
	Error message output only when invalid, and `Welcome` + username output once, after the loop
	Example:
	`username = input("Enter a username")`
	`while username == "" OR username.length > 12`
	`  print("Username must be 1 to 12 characters")`
	`  username = input("Enter a username")`
	`endwhile`
	`print("Welcome " + username)`
	Example with `do ... until`:
	`do`
	`  username = input("Enter a username")`
	`  if username.length < 1 OR username.length > 12 then`
	`  print("Invalid username, try again")`
	`  endif`
	`until username.length >= 1 AND username.length <= 12`
	`print("Welcome " + username)`
		6	Allow any logically valid method, including separate `if` statements with their own messages, a Boolean flag, or a Python `while True:` loop with `break`. The fifth bullet needs the two checks joined the right way for the loop used (AND joining the two invalid tests, or OR joining the two valid tests, loses this mark). A single `if` with no loop (only one re-entry): max 3 (input, presence check, length check). Accept `len(username)`. `username.length >= 12` as invalid (rejects 12 characters) loses the length-check mark. Do not accept `input` as a variable name. Do not accept flowcharts, structured English or bullet points.

Total: 22 marks

MODULE O ROBUST PROGRAMS · O4

Authentication in programs: mark scheme

22 marks · OCR J277 specification 2.3.1 (defensive design: anticipating misuse, authentication to confirm the identity of a user, practical experience of designing simple authentication, e.g. username and password), with 2.1.2 (trace tables) and 2.2.3 (arrays, functions)

How to use. 1 mark per bullet up to the maximum shown. **/** separates alternative words or answers; **//** separates alternative statements for the same mark. Words in (brackets) are not needed for the mark. … at the end of one point and the start of the next means the second depends on the first. Code may be in OCR Exam Reference Language or any high-level language: mark the logic, and do not penalise minor syntax slips (e.g. a missing `then`, `endif`, colon or bracket). Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.3.1	Confirming / checking the identity of a user (before they are allowed access to the program / its data)	**1**	Allow: checking that a user is who they say they are // checking a user is allowed to access the system. An example alone (e.g. "entering a username and password") is NE. Allow BOD if the example is linked to the identity check (e.g. "checking the password matches the stored one to prove who the user is"). Do not accept: checking that the data entered is sensible / valid (that is validation).
1(b) J277 2.3.1	1 mark per correct row: Seats between 1 and 8: Input validation Password entered again before a booking is deleted: Authentication Username and password compared with the stored ones: Authentication	**3**	A row with both boxes ticked scores 0.
2(a) J277 2.3.1, 2.2.1	1 mark per correct row: `boxoffice2`, `Curtain#47`: Access granted `boxoffice2`, `Curtain#74`: Access denied `boxoffice3`, `Curtain#47`: Access denied	**3**	Ignore quotation marks and case of the output text. The message must be the program's exact output: do not accept "denied" / "wrong password" / "no access".

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>2(b)
J277 2.3.1, 2.2.1</td><td>Max 2 from:
Access would be granted if only one of the username or the password is correct // both no longer need to be correct
Someone who knows / guesses (only) a valid username could log in without the password // someone could log in with the password but any / a wrong username
So people who are not authorised / not the staff member could get in // the program no longer confirms the user's identity</td><td>2</td><td>Allow the correct row of 2(a) as an example: the 2nd and 3rd inputs would now give Access granted (1 mark, as MP1 or MP2).
“Less secure” / “anyone could get in” on its own is NE.
Do not accept: “nobody could log in” / “both must still be correct”.</td></tr>
<tr><td>3(a)
J277 2.1.2, 2.3.1</td><td>1 mark per bullet:
<code>attempts</code>: 1, 2, 3 (after the given 0)
<code>loggedIn</code>: changes to <code>true</code> only once, on the 3rd attempt (the row where <code>pwd</code> is Curtain#47)
<code>pwd</code>: Curtain47, Curtain#74, Curtain#47 (in this order)
Output: Try again, Try again, Welcome (in this order, Welcome last)
<table>
<tr><th>ATTEMPTS</th><th>LOGGEDIN</th><th>PWD</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>false</td><td></td><td></td></tr>
<tr><td></td><td></td><td>Curtain47</td><td>Try again</td></tr>
<tr><td>1</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>Curtain#74</td><td>Try again</td></tr>
<tr><td>2</td><td></td><td></td><td></td></tr>
<tr><td></td><td>true</td><td>Curtain#47</td><td></td></tr>
<tr><td>3</td><td></td><td></td><td>Welcome</td></tr>
</table></td><td>4</td><td>Values may be repeated on later rows (e.g. <code>false</code> written on each row until it changes). Allow rows to be combined, e.g. <code>attempts</code> 1 on the same row as Curtain47 and Try again.
Allow True/False, T/F, 1/0 for <code>loggedIn</code>.
<code>attempts</code>: do not award if 4 appears.
Output: do not award if Locked appears, or if a third Try again appears. Ignore quotation marks.</td></tr>
<tr><td>3(b)
J277 2.3.1 (anticipating misuse, authentication), 1.4.1</td><td>Max 2 from:
To stop someone (repeatedly) guessing / trying many passwords // to prevent a brute-force attack …
… after 3 wrong attempts the account / program is locked // the person only gets 3 tries …
… so an unauthorised person is very unlikely to find the correct password // it would take far too long to try every possible password</td><td>2</td><td>“To make it more secure” / “to stop hackers” on its own is NE.
Allow “stops a program / bot trying every combination” for MP1.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE

4(a)
J277 2.3.1, 2.2.3 (arrays, functions), 2.2.1
	1 mark per bullet to max 5:
	Loop / iteration that goes through the array(s)…
	… covering every item, index 0 to 19 / `users.length - 1` (and no further)
	Compares `users[i]` with the parameter `user`
	Compares `passwords[i]` with `pwd` using the **same** index as the matching username
	Returns `true` when both match
	Returns `false` only after every item has been checked with no match (e.g. after the loop)
	Example answer
	`function checkLogin(user, pwd)`
	`    for i = 0 to users.length - 1`
	`        if users[i] == user AND passwords[i] == pwd then`
	`            return true`
	`        endif`
	`    next i`
	`    return false`
	`endfunction`
	Alternative (flag)
	`function checkLogin(user, pwd)`
	`    found = false`
	`    i = 0`
	`    while i < 20 AND found == false`
	`        if users[i] == user then`
	`            if passwords[i] == pwd then`
	`                found = true`
	`            endif`
	`        endif`
	`        i = i + 1`
	`    endwhile`
	`    return found`
	`endfunction`
		5	Allow `for i = 0 to 19`, `for i in range(20)` / `range(len(users))`. Do not award MP2 for `0 to 20` or `1 to 20`.
			MP3 and MP4 may be in one condition joined with AND, or as nested IFs. Do not award MP4 if the password is searched for in a separate loop / at a different index (e.g. any password in the array).
			Do not award MP6 if `return false` is in an `else` inside the loop (the function would stop after checking only the first item).
			A flag variable returned at the end is acceptable for MP5 and MP6.
			Must use the parameters: if `user` / `pwd` are overwritten by `input`, max 4.
			Printing "true"/"false" instead of returning: do not award MP5 or MP6.
			Allow = or == for comparison.

5 J277 2.3.1 (anticipating misuse, authentication), 1.4.2	1 mark for a way, 1 mark for how it makes logging in harder, e.g.: Make staff use strong passwords (e.g. a minimum length, a mix of letters, numbers and symbols) … so passwords are harder to guess / take longer to brute-force Show the same message whichever item is wrong / do not say whether it was the username or the password … so an attacker cannot find out which usernames are valid Add a second step / two-step (two-factor) authentication, e.g. a code sent to the staff member's phone … so a person who knows the password also needs the staff member's device Add a delay / wait time after each failed attempt … so guessing many passwords takes too long Hide the password as it is typed (e.g. show *) … so people nearby cannot read it Store the passwords encrypted / not as plain text in the code or file … so someone who sees the program / file cannot read the passwords	**2**	Mark the first way only; the 2nd mark must explain that way. Credit any other valid way with an explanation (e.g. biometrics; regular password changes). Do not accept: locking the account after failed attempts (that is limiting attempts); firewall / anti-malware; encryption not linked to the stored passwords (e.g. 'encrypt the data' alone); 'more secure' alone.

Total: 22 marks

MODULE O ROBUST PROGRAMS · O5

Why we test, and iterative vs final testing: mark scheme

20 marks · OCR J277 spec 2.3.2 (the purpose of testing; types of testing: iterative, final/terminal; the difference between testing modules of a program during development and testing the program at the end of production; identify syntax and logic errors; refining algorithms)

How to use. 1 mark per bullet up to the Max shown. **/** separates alternative answers. Words in (brackets) are not needed for the mark. **...** at the start of a bullet means it can only be awarded with the bullet before it. Credit any other valid answer that shows the same understanding. Code answers: mark the logic, not the language; allow OCR Exam Reference Language or any high-level language; do not accept maths symbols such as ≥ in code.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.3.2	1 mark for each correct row: Cost module tested as soon as it is written: Iterative Finished program run to check it does everything the museum asked for: Final/terminal Receipt module tested again after an error is corrected: Iterative All three modules checked working together before opening: Final/terminal	**4**	A row with both boxes ticked scores 0.
2(a) J277 2.3.2	1 mark each to max 2: To find / detect errors / bugs (so they can be fixed / removed) To check the program works as intended / gives the correct output / result (e.g. the correct ticket cost) To check the program meets the (museum's / user's) requirements To check the program does not crash / is robust // copes with invalid / unexpected input (e.g. a customer pressing the wrong key) To make sure there are no (known) problems before the program is released / used by the public	**2**	Two different purposes are needed: "find bugs" and "find errors" are the same point. Allow answers that say what could happen without testing (e.g. "the kiosk might charge the wrong amount"). Do not accept "to make the program faster" or "to make it more secure" alone. Credit any other valid purpose.
3(a) J277 2.3.2	1 mark each to max 2: Testing carried out during development / while the program is being written Each module / part (of the program) is tested (on its own) as it is completed / written Errors found are corrected / the code is changed / refined ... ... and the module is tested again // the cycle of test, fix and re-test is repeated (until the module works)	**2**	Must be a description, not just a repeat of the name (e.g. "testing that iterates" = 0). Do not accept a description of test data (e.g. normal or boundary data) on its own.

QUESTION	ANSWER	MARKS	GUIDANCE
3(b) J277 2.3.2	1 mark each to max 2: Testing carried out at the end of development / production // when the program is finished / complete // before the program is released The whole program is tested (as one) // all the modules are tested together To check the program meets all the requirements / works as a whole	**2**	"Testing at the end" on its own scores 1 (first bullet). Do not accept "the last test" or "testing that is final" with no further detail.
4(a) J277 2.3.2	1 mark each: Test 2: 90 Test 3: 225	**2**	Allow £90 / 90.00 and £225 / 225.00. The expected result comes from the museum's rule, not from the code: do not accept 120 for Test 2.
4(b) J277 2.3.2	Test 2 // the test with 10 people	**1**	Mark 4(b) on its own: do not follow through from a wrong expected result in 4(a). Naming more than one test = 0.
4(c) J277 2.3.2	1 mark each: Line 02 `if people >= 10 then` // `if people > 9 then`	**2**	The two marks are independent. Allow the corrected line without `then` or with a colon (Python style). Allow `if 10 <= people then`. Do not accept ≥, `=>` or `if people == 10`. Do not accept a change to line 03 or line 05.
4(d) J277 2.3.2	1 mark for: Test the function / module again // run the tests (Test 1 to Test 3) again (to check the error is fixed / that no new errors were made)	**1**	Do not accept "carry out final testing" or "release the program" alone: the module must be re-tested first.

QUESTION	ANSWER	MARKS	GUIDANCE
5(a) J277 2.3.2	1 mark for each reason, 1 mark for an expansion of that reason, to max 4 (max 2 per reason), e.g.: Modules that work on their own may not work together ... ... e.g. the cost module may pass the wrong value / a value in the wrong form to the payment / receipt module Checks the whole program meets all the museum's requirements ... ... iterative testing only checked each module against its own part of the task // a requirement that involves several modules (e.g. a receipt showing the correct total paid) has not been tested yet A change made to one module during development may have caused an error in another ... ... so the finished program must be tested as a whole before release Tests the program in the conditions it will be used in ... ... e.g. on the real kiosk / with real customers' inputs, which may cause errors that did not appear in development Last chance to find errors before the program is released / used by the public ... ... errors found after release could cost the museum money / charge customers the wrong amount / stop the kiosk working	**4**	Award an expansion only when it follows its own reason. Two reasons that make the same point (e.g. "modules may not work together" and "the parts may not link up") score as one reason. Credit any other valid reason with its expansion. A generic answer with no reference to the kiosk program: max 3.

Total: 20 marks

MODULE O ROBUST PROGRAMS · O6

Syntax errors vs logic errors: mark scheme

22 marks · OCR J277 specification 2.3.2 (identify syntax and logic errors; syntax errors break the grammatical rules of the language and stop it from being run/translated; logic errors produce unexpected output; the purpose of testing) and 2.1.2 (identify syntax/logic errors in code and suggest fixes; trace tables)

How to use. 1 mark per bullet up to the maximum shown. **/** separates alternative words or answers. Words in (brackets) are not needed for the mark. Corrected code may be in OCR Exam Reference Language or any high-level language: mark the logic, and do not penalise minor syntax slips in the student's own code (e.g. a missing `then` or colon) unless the slip is the error the question asks about. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.3.2	Max 2 from: An error that breaks the (grammatical) rules of the programming language / code not written in the correct form for the language It stops the program from being run / translated / compiled / interpreted The translator / IDE reports it (with an error message / line number) An example, e.g. a misspelt keyword / missing bracket / missing quote mark	**2**	The example mark is only given alongside one of the first three points (an example on its own scores 0). Do not accept "an error in the syntax" (repeats the question) or "a spelling mistake" on its own. Do not accept "the program crashes while running".
1(b) J277 2.3.2	Max 2 from: An error that produces unexpected / incorrect output / results / the program does not do what the programmer intended The program (still) runs / is translated / does not break the rules of the language No error message is given / the translator does not detect it An example, e.g. > used instead of >= / a loop that runs the wrong number of times / the wrong variable used	**2**	The example mark is only given alongside one of the other points. Do not accept "an error in the logic" (repeats the question) or "the program does not work" / "the program crashes".
2(a) J277 2.3.2, 2.1.2	1 mark per correct row: `print("Card full"`: Syntax error `discount = price * 0.02`: Logic error `if stamps > 10 then`: Logic error `endfi`: Syntax error	**4**	A row with both boxes ticked scores 0 for that row. Row 2 works out 2%, not 20%. Row 3 leaves out exactly 10 stamps. Both run without an error message.
3(a) J277 2.3.2	Syntax (error)	**1**	
3(b) J277 2.3.2	Logic (error)	**1**	Do not accept "runtime error": the program ran to the end and gave a result.

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>3(c)
J277 2.3.2, 2.1.2</td><td>Max 2 from:
Test the program with test data whose correct / expected result is known (e.g. 2 hours should give £16)
Compare the actual output with the expected output
Use a trace table / dry run to follow the values of the variables line by line
... to find the line where a value first goes wrong
Use IDE debugging tools, e.g. breakpoints / stepping through the code / watching variable values
Read through / check the calculation (e.g. the hours used, the multiplication by 8)</td><td>2</td><td>Do not accept “the translator / error message shows the line”: a logic error gives no error message.
Do not accept “test it” on its own without saying what is done with the result.</td></tr>
<tr><td>4(a)
J277 2.1.2, 2.3.2</td><td>Line 06 (identified)
... <code>print("Charge: " + str(charge) + " pounds")</code> / closing bracket added at the end</td><td>2</td><td>The correction mark may be given if the line number is missing but the rewritten line makes clear which line is meant.
Allow a description, e.g. “add a) at the end of line 06”.</td></tr>
<tr><td>4(b)
J277 2.1.2, 2.3.2</td><td>Line 03 (identified)
... <code>if charge < 6 then</code></td><td>2</td><td>Allow <code>charge <= 6 / 6 > charge / kg < 3 / kg <= 3 / kg <= 2</code> (all give the right charge for whole kilograms).
Do not allow ≤ or =<. Do not credit changes to lines 02 or 04 (they do not fix the error).</td></tr>
<tr><td>4(c)
J277 2.1.2</td><td>Charge: 6 pounds</td><td>1</td><td>Allow 6 on its own. Ignore quotation marks.
Do not accept 10 (that is the output of the fully corrected program).</td></tr>
<tr><td>5(a)
J277 2.1.2</td><td><code>i</code>: 1, 2, 3 AND <code>reading</code>: 12, 9, 15 (in this order)
<code>total</code>: 0, 1, 3, 6
<code>average</code>: 2 AND Output: 2 (once, at the end)
<table><tr><th>I</th><th>READING</th><th>TOTAL</th><th>AVERAGE</th><th>OUTPUT</th></tr><tr><td></td><td></td><td>0</td><td></td><td></td></tr><tr><td>1</td><td>12</td><td>1</td><td></td><td></td></tr><tr><td>2</td><td>9</td><td>3</td><td></td><td></td></tr><tr><td>3</td><td>15</td><td>6</td><td></td><td></td></tr><tr><td></td><td></td><td></td><td>2</td><td>2</td></tr></table></td><td>3</td><td>Values may be repeated on later rows. Allow 2.0 for <code>average</code> and Output.
Allow the starting 0 for <code>total</code> to be missing if 1, 3, 6 are correct.
Do not give the third mark for 12 (the intended average): the trace must follow the code as written.</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
5(b) J277 2.1.2, 2.3.2	Line 04 (identified) … `total = total + reading`	**2**	The correction mark may be given if the line number is missing but the rewritten line makes clear which line is meant. Allow `total += reading`. Do not allow `total =+ reading`.

Total: 22 marks

MODULE O ROBUST PROGRAMS · O7

Choosing test data: mark scheme

21 marks · OCR J277 specification 2.3.2 (selecting and using suitable test data: normal, boundary, invalid/erroneous; identify suitable test data for a given scenario)

How to use. 1 mark per bullet up to the maximum shown. **/** separates alternative words or answers. Words in (brackets) are not needed for the mark. **...** means the mark depends on the point before it. Test data must be an actual value; a description such as "a number between 1 and 50" does not score. Credit any other valid answer that shows the same understanding.

QUESTION	ANSWER	MARKS	GUIDANCE
1(a) J277 2.3.2	Data that should be accepted by the program (without causing errors) / typical valid data	**1**	Allow "data the program is expected to work with / that should pass the validation". Do not allow an example on its own.
1(b) J277 2.3.2	Invalid: data of the correct data type (that should be rejected) Erroneous: data of the incorrect / wrong data type (that should be rejected)	**2**	The data type idea is needed for each mark. "Both should be rejected" on its own scores 0. Allow a correct example used to support the point, e.g. invalid: 101 for a range 1 to 100 (right type, out of range); erroneous: "ten" for a number.
2(a) J277 2.3.2	1 mark per correct row (type **and** accept/reject both correct): 8: Normal ... Accept 2: Boundary ... Accept 12: Boundary ... Accept 20: Invalid ... Reject "ten": Erroneous ... Reject	**5**	Rows 2 and 3: allow "Boundary and normal" (the values are also valid). Row 4: allow "Invalid/erroneous" written as one category. Do not allow "Erroneous" alone: 20 is the correct data type. Row 5: allow "Invalid/erroneous" written as one category. Do not allow "Invalid" alone: "ten" is the wrong data type. Allow "allowed / valid / yes" for Accept and "not allowed / error message / no" for Reject.

QUESTION	ANSWER	MARKS	GUIDANCE
3(a) J277 2.3.2	A username of exactly 6 characters, e.g. `Pebble` A username of exactly 12 characters, e.g. `RiverOtter42`	**2**	Count the characters. Any characters may be used. Two different 6-character usernames (or two 12-character usernames) score 1 mark only: the two edges are different tests. Allow a clearly stated length with an example of that length. A length alone with no username (e.g. "6 letters") scores 0.
3(b) J277 2.3.2	A username of 5 or fewer / 13 or more characters, e.g. `Ab12` / `Pebble1234567` It is the correct data type / it is text / a string … but its length is outside 6 to 12, so it should be rejected	**2**	Mark 2 needs the idea of the correct data type. Allow an empty username for mark 1. Do not allow "it is too short" on its own for mark 2.
4(a) J277 2.3.2	1	**1**	More than one value given scores 0.
4(b) J277 2.3.2	It rejects 1 / outputs "Setting must be 1 to 8" / asks for the setting again It should accept 1 / save the setting / output "Setting saved" (because 1 is in the range 1 to 8)	**2**	Mark (b) for the value 1 only: 5, 8 and 9 are handled correctly by this code, so no FT from a wrong answer to (a). Do not allow "it crashes".
4(c) J277 2.3.2	Boundary (test data)	**1**	Allow "boundary and normal".
4(d) J277 2.3.2	Any value that is not a whole number / integer, e.g. `"five"` / `4.5` / `"#"`	**1**	Do not allow a whole number such as 0, 9 or −3 (this is invalid test data, the correct data type). Do not allow a description such as "a word" with no actual value.
5(a) J277 2.3.2	Max 4 from: All three values are normal data / only checks that valid data is accepted Boundary test data should be used, e.g. 1 / 50 (to check the edges are accepted) Invalid test data should be used, e.g. 0 / 51 / −5 (to check it is rejected) Erroneous test data should be used, e.g. `"ten"` / 2.5 (to check it is rejected) Errors at the edges of the range / in the validation (e.g. > used instead of >=) would not be found	**4**	Mark points 2 to 4 each need a correct example value for that type. Allow "Invalid/erroneous" as one category if the example fits. If 0 or 51 is given as "boundary" data that should be rejected, credit it for mark point 3 (invalid). Do not allow a whole number as erroneous. Do not allow "they should test more values" on its own.

Total: 21 marks

MODULE O ROBUST PROGRAMS · O8

Writing a test plan: mark scheme

21 marks · OCR J277 spec 2.3.2: ability to create/complete a test plan; selecting and using suitable test data (normal, boundary, invalid/erroneous); identify syntax and logic errors; refining algorithms

How to use. 1 mark per bullet up to the maximum shown. **/** separates alternative words; **//** 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 between 1 and 14" = 0). **Types of test data** must use the OCR names: normal, boundary, invalid, erroneous ("extreme" and "abnormal" are not OCR terms; do not credit them). For "give two" items, mark the first two answers only. 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)
J277 2.3.2</td><td>Test number
Purpose of the test / what is being tested / description of the test
Type of test (data) // normal / boundary / invalid / erroneous
Expected result / expected output
Actual result
Pass / fail // whether the actual result matches the expected result
Action needed / comments / what was changed</td><td>2</td><td>1 mark per bullet, max 2. Credit any other valid item of a test plan. Do not accept "the test data" (excluded by the question) or "the code" / "the program". Do not award the same item twice in different words.</td></tr>
<tr><td>1(b)
J277 2.3.2</td><td>So it can be compared with the actual result (to see if the test passes / fails)
To check whether the program works as intended // to show whether there is an error (e.g. a logic error)</td><td>1</td><td>Any 1 bullet. Do not accept "so the program works" / "to test the program" alone (NE).</td></tr>
<tr><td>2
J277 2.3.2</td><td>Test 2, type: Boundary
Test 2, expected result: Accepted, (outputs) cost £168
Test 3, test data: 1
Test 4, type: Invalid
Test 5, test data: any value of the wrong data type, e.g. <code>ten</code> / <code>abc</code> / <code>7.5</code>
<table><tr><th>Test</th><th>Test data</th><th>Type of test</th><th>Expected result</th></tr><tr><td>2</td><td>14</td><td>Boundary</td><td>Accepted, cost £168</td></tr><tr><td>3</td><td>1</td><td>Boundary</td><td>Accepted, cost £12</td></tr><tr><td>4</td><td>0</td><td>Invalid</td><td>Rejected, Enter 1 to 14 days</td></tr><tr><td>5</td><td>ten</td><td>Erroneous</td><td>Rejected, Enter 1 to 14 days</td></tr></table></td><td>5</td><td>1 mark per bullet.
Test 2 type: do not accept normal / extreme. Test 2 result: accept "accepted" / "valid" / "outputs £168"; if a cost is given it must be £168. Do not accept "it works" (NE).
Test 4 type: accept "invalid/erroneous" written together; do not accept "erroneous" alone (0 is the correct data type).
Test 5: accept a real number (e.g. 2.5) or text; accept an empty input. Do not accept a whole number (e.g. 20 or −3: correct data type, so invalid not erroneous).</td></tr>
</table>

<table>
<tr><th>QUESTION</th><th>ANSWER</th><th>MARKS</th><th>GUIDANCE</th></tr>
<tr><td>3
J277 2.3.2</td><td>A weight above 0 and up to 2 (e.g. 1.5) … returns <code>Small</code>
A weight above 2 and up to 10 (e.g. 6) … returns <code>Medium</code>
A weight above 10 and up to 30 (e.g. 12) … returns <code>Large</code></td><td>3</td><td>1 mark per row: the weight must be in the correct range and the returned value must match it. Rows may be in any order, but each route must be tested once for its mark (two Small rows score only once). Accept the edge values 2 (Small) and 10 (Medium). Do not accept weights of 0 or less, or above 30 (the function never receives them). Accept returned values with or without quotation marks.</td></tr>
<tr><td>4(a)
J277 2.3.2</td><td>Test 2: Fan off … Fail
Test 3: Reading rejected … Pass
Test 4: Fan off … Pass
<table><tr><th>Test</th><th>Actual result</th><th>Pass/Fail</th></tr><tr><td>2</td><td>Fan off</td><td>Fail</td></tr><tr><td>3</td><td>Reading rejected</td><td>Pass</td></tr><tr><td>4</td><td>Fan off</td><td>Pass</td></tr></table></td><td>3</td><td>1 mark per row: both the actual result and Pass/Fail must be correct. The trace must follow the code as written: 28 is not greater than 28, so the else branch runs. FT Pass/Fail from the candidate's actual result only if the Pass/Fail correctly compares it with the expected result (max 2 if the actual result for test 2 is wrong).</td></tr>
<tr><td>4(b)
J277 2.3.2</td><td>Line 04
<code>elseif temp >= 28 then</code></td><td>2</td><td>1 mark for the line number, 1 for the refined line. Accept <code>temp > 27</code> (temp is a whole number) and <code>28 <= temp</code>. Accept the refined line without <code>elseif/then</code>, or in a high-level language, if the comparison is correct. Do not accept => or ≥. Do not accept an English description alone ("change > to greater than or equal") for the second mark.</td></tr>
<tr><td>4(c)
J277 2.3.2</td><td>Run the tests again // re-test (using the same test plan / test data)
Repeat the tests to check test 2 now passes (and the other tests still pass)</td><td>1</td><td>Any 1 bullet. Accept "update the actual results and Pass/Fail columns after re-running". Do not accept "change the expected result" (the expected result was correct) or "delete test 2".</td></tr>
</table>

QUESTION	ANSWER	MARKS	GUIDANCE
5 J277 2.3.2	Weakness: no boundary data is tested … Improvement: add tests for 2 and 10, both expected to be accepted Weakness: no erroneous data is tested … Improvement: add a test of the wrong data type, e.g. `six` / `4.5`, expected to be rejected Weakness: only one invalid value / nothing below the range is tested … Improvement: add an invalid test below the range, e.g. 1 or 0, expected to be rejected Weakness: expected results are vague / not specific ("It works", "Error") … Improvement: state exactly what should happen, e.g. "accepted, booking continues" / "rejected, error message shown and group size asked for again" Weakness: tests 1 and 2 test the same thing (both normal) … Improvement: replace one of them with a boundary / erroneous / other test Weakness: no actual result / pass-fail columns … Improvement: add columns to record the actual result and whether the test passed	**4**	Mark in pairs: 1 mark for each weakness, 1 for a matching improvement, max 2 pairs. The improvement must match its weakness. An improvement that names test data must give an actual value in the right category (e.g. boundary 2 or 10, not 11). Award a weakness on its own if the improvement is missing or wrong. Do not accept "there are not enough tests" alone (NE) unless the missing type is named. Do not award the same weakness twice.

Total: 21 marks