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