Past-paper walk-through

Program Design and Development ▪ 1.3

eXercise

Questions in this set

- 2025 Set 2 Q2

Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

Question 2

2025 Set 2 ▪ Q2

Refer to your Personalized Project Reference when answering this question.
(a) Consider the first selection statement included in the Procedure section of your Personalized Project Reference. Identify the Boolean expression in this selection statement. Identify a specific value or set of values that will cause this expression to evaluate to `false`. Explain why the specified value(s) will cause this expression to evaluate to `false`.

Question 2

2025 Set 2 ▪ Q2

Refer to your Personalized Project Reference when answering this question.

(b) Consider the code segment in part (ii) of the List section of your Personalized Project Reference. Suppose another programmer modifies this code segment. Describe a modification the other programmer could make to this code segment that would result in a logic error. Explain why this modification would result in a logic error.

Question 2

2025 Set 2 ▪ Q2

Refer to your Personalized Project Reference when answering this question.

(c) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Describe the functionality provided by this procedure. Explain how implementing this functionality as a procedure results in your program being easier to maintain than if the functionality were not implemented as a procedure.

Solution 2

(a) Mark scheme

- Create Performance Task — Written Response 2a: Algorithm Development (0–1 pt). Scored against the FIRST selection (if) statement in the Procedure section of the student's own Personalized Project Reference — no fixed numeric answer. Full credit (1 point) requires ALL three: (1) correctly identify the Boolean expression from that first selection statement (e.g., for 'if (score $>$ 2)' the expression is 'score $>$ 2'; naming the full 'if (...)' header instead of just the expression is also acceptable); (2) give a specific value or set of values for the variable(s) in the expression that makes it evaluate to 'false' — e.g., 'score $=$ 2' or 'score $=$ 0' both make 'score $>$ 2' false; (3) explain WHY that value makes the expression false — e.g., 'because 2 is not greater than 2, the condition score $>$ 2

Solution 2

evaluates to false'. The statement doesn't need to be inside a procedure, and may or may not have an else clause. No credit if: the Procedure section has no selection statement; the identified expression doesn't match the actual code; the given value wouldn't actually make the real expression false; or the expression/value is implausible or inconsistent with the program.

Solution 2

(b) Mark scheme

- Create Performance Task — Written Response 2b: Errors and Testing (0–1 pt). Scored against the code segment shown in part (ii) of the List section of the student's own Personalized Project Reference — no fixed answer. Full credit (1 point) requires: (1) describe ONE specific modification to that code (changing, adding, or deleting a statement — anywhere in the part (ii) code, not necessarily to the list itself) — e.g., 'change the loop bound from len(myList) to len(myList) + 1', or 'change the index used to access the list from i to i + 1'; (2) explain why that modification causes a LOGIC ERROR — i.e., why the program would then behave incorrectly or unexpectedly (a crash counts as a logic error too, e.g. an out-of-bounds index) — e.g., 'this would make the program try to access an index

Solution 2

that doesn't exist in the list, so it would crash or read/use the wrong data, giving an incorrect result even though the program keeps running'. If more than one modification is described, ALL of them must be correctly explained to earn the point. No credit for: stating only 'the program won't run' with no further explanation; a list/collection not actually present in part (ii); a modification/explanation that doesn't match the real code in part (ii); or an implausible/inaccurate explanation.

Solution 2

(c) Mark scheme

- Create Performance Task — Written Response 2c: Data and Procedural Abstraction (0–1 pt). Scored against the procedure identified in part (i) of the Procedure section of the student's own Personalized Project Reference — no fixed answer. Full credit (1 point) requires BOTH: (1) describe what the identified procedure actually does (its functionality) — e.g., 'the checkWin procedure checks whether the player's score has reached the target value and returns true or false'; (2) explain how implementing that functionality AS A PROCEDURE (rather than repeating the same code inline everywhere it's needed) makes the program easier to maintain — e.g., 'because the win-checking logic exists in one place inside the procedure, if the winning condition changes later the programmer only has to edit

Solution 2

it once instead of finding and fixing every copy of that logic scattered through the program, which reduces the chance of missing a spot or introducing new bugs'. The response does not need to mention a parameter; if it does, an explicit or an implicit parameter is both acceptable. No credit if: no procedure is included in part (i); the response describes a different procedure than the one in part (i); or the maintainability explanation is vague or incorrect (e.g., just 'it makes the code shorter/organized' without tying it to why future maintenance is easier).