Past-paper walk-through

Identifying and Correcting Errors ▪ 1.4

eXercise

Questions in this set

- 2026 Q2
- 2025 Set 1 Q2
- 2025 Set 2 Q1
- 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

2026 ▪ Q2

Refer to your Personalized Project Reference when answering this question.
(a) Consider the first iteration statement in the Procedure section of your Personalized Project Reference. Identify the variable(s) that determine when the iteration statement stops iterating. Identify specific values for the variable(s) that will cause the iteration statement to stop. Explain why these value(s) would cause the iteration statement to stop. If there are no such variable(s) or specific value(s), explain how the iteration statement stops.

Question 2

2026 ▪ Q2

Refer to your Personalized Project Reference when answering this question.

(b) Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Another programmer is planning to use your procedure in a different program and wants to write some test cases to check whether it will work. Write a procedure call with specific argument(s) that your procedure will accept but will cause your procedure to behave incorrectly. Describe the incorrect behavior. Explain why the incorrect behavior occurs as a result of this call. If it is not possible for a call with accepted arguments to cause your procedure to behave incorrectly, explain why this is the case for your procedure.

Question 2

2026 ▪ Q2

Refer to your Personalized Project Reference when answering this question.

(c) Consider the list included in part (i) of the List section of your Personalized Project Reference. Explain how the list uses abstraction to manage complexity in your program. Suppose the list was not included in your program. Describe how you would need to adjust the code segment in part (ii) of the List section of your Personalized Project Reference so that it has the same behavior. If it is not possible to have the same behavior without the use of the list, explain why.

Question 2

2025 Set 1 ▪ 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 `true`. Explain why the specified value(s) will cause the expression to evaluate to `true`.

Question 2

2025 Set 1 ▪ Q2

Refer to your Personalized Project Reference when answering this question.

(b) Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Suppose another programmer modifies the code within this procedure. Describe a modification the other programmer could make that would cause this procedure to have a logic error. Describe how the behavior of this procedure would change because of the error.

Question 2

2025 Set 1 ▪ Q2

Refer to your Personalized Project Reference when answering this question.

(c) Consider the list included in the List section of your Personalized Project Reference. Suppose another programmer adds several new elements to the end of the list. Explain how the code segment in part (ii) of the List section would need to be modified to account for the additional elements. If no changes to the code segment are necessary, explain why this is the case for your program.

Solution 2

(a) Mark scheme

- Written Response 2a (Algorithm Development), scored 0-1 points against the student's own PPR – no single fixed answer exists. To earn the point the response must: (1) identify the Boolean expression in the first selection statement in the Procedure section of the PPR (if multiple selection statements exist, use the first one; the header form, e.g. 'if (x > 2)', is acceptable instead of just 'x > 2'; it need not be stated verbatim as long as it is described, and need not be inside a procedure), (2) identify a specific value or set of values that will cause that Boolean expression to evaluate to true, and (3) explain why the specified value(s) cause it to evaluate to true. The selection statement may or may not contain an else clause. Do NOT award the point if the PPR's Procedure section has no

Solution 2

selection statement, if the identified Boolean expression doesn't match the first selection statement's code, if the identified true-causing value(s) don't match that code, or if the identified expression/value is implausible, inaccurate, or inconsistent with the program.

Solution 2

(b) Mark scheme

- Written Response 2b (Errors and Testing), scored 0-1 points against the student's own PPR – no single fixed answer exists. To earn the point the response must: (1) describe a modification another programmer could make to the procedure in part (i) of the PPR's Procedure section that would cause it to have a logic error (a mistake causing incorrect/unexpected behavior, including a described crash), and (2) describe how the procedure's behavior would change because of that introduced error. The modification may be a code change, addition, or deletion. If multiple procedures are given, use the one the response references (or the first procedure if none is referenced). If more than one modification is described, the behavior change must be correctly described for every one of them. Do NOT

Solution 2

award the point if no procedure is included in part (i) of the Procedure section, if the response doesn't apply to that procedure, if the response only states the program 'will not run' without further explanation, or if the described modification/behavior change is implausible, inaccurate, or inconsistent with the procedure.

Solution 2

(c) Mark scheme

- Written Response 2c (Data and Procedural Abstraction), scored 0-1 points against the student's own PPR – no single fixed answer exists. To earn the point the response must EITHER (a) explain how the code segment that accesses or manipulates data stored in the list (or other collection type) from part (i) of the PPR's List section would need to change if another programmer appended several new elements to the end of the list, OR (b) explain why no change to that code segment would be necessary. If multiple lists are given, use the list(s) the response references (or the first list if none is referenced); the point can still be earned if the described modification changes the program's functionality. Do NOT award the point if no list/collection is included in part (i) of the List section, if the

Solution 2

response doesn't apply to that list, if the list's use is trivial and doesn't help fulfil the program's purpose, if the described modification makes the list unnecessary, or if the explanation is implausible, inaccurate, or inconsistent with the program.

Question 1

2025 Set 2 ▪ Q1

Identify an unexpected or invalid input that a user could provide to your program. Describe the behavior of your program after it receives this input. If it is not possible for your program to accept an unexpected or invalid input, explain why this is the case.

Solution 1

- Create Performance Task — Written Response 1: Program Design, Function, and Purpose (0–1 pt). This is scored against the student's OWN program, so there is no single fixed answer — credit is based on the rationale given. Full credit (1 point) requires EITHER (a) identify a specific unexpected/invalid input the program could receive — e.g., a user enters a negative number where a positive quantity is expected, enters text where a number is expected, leaves a field blank, or presses a key/button the program doesn't handle — AND describe what the program actually does when it receives that input (e.g., 'the program crashes with an error', 'the program ignores the input and keeps the previous value', 'the program treats it as 0 and continues running'); OR (b) explain concretely why the

Solution 1

program cannot receive any invalid/unexpected input at all (e.g., input is restricted to a fixed set of on-screen buttons so no invalid value can ever be entered). If more than one input is identified, only one needs a correct description. No credit if: the identified input is implausible, inaccurate, or inconsistent with how the program actually works; or the response gives no real detail of the resulting behavior (e.g., just 'it doesn't work') or no plausible reason the program can't receive bad input.

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).