Past-paper walk-through

Iteration ▪ 3.8

eXercise

Questions in this set

- 2026 Q2
- 2024 Set 1 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

2024 Set 1 ▪ Q2

Refer to your Personalized Project Reference when answering this question.
(a) Consider the first iteration statement included in the Procedure section of your Personalized Project Reference. Describe what is being accomplished by the code in the body of the iteration statement.

Question 2

2024 Set 1 ▪ Q2

Refer to your Personalized Project Reference when answering this question.

(b) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Write two calls to your procedure that each cause a different code segment in the procedure to execute. Describe the expected behavior of each call. If it is not possible for two calls to your procedure to cause different code segments to execute, explain why this is the case for your procedure.

Question 2

2024 Set 1 ▪ Q2

Refer to your Personalized Project Reference when answering this question.

(c) Suppose another programmer provides you with a procedure called `checkValidity(value)`

Question 2

that returns `true` if a value passed as an argument is considered valid by the other programmer and returns `false` otherwise. Using the list identified in the List section of your Personalized Project Reference, explain in detailed steps an algorithm that uses `checkValidity` to check whether all elements in your list are considered valid by the other programmer. Your explanation must be detailed enough for someone else to write the program code for the algorithm that uses `checkValidity`.

Solution 2

(a)

Mark scheme

- Written Response 2(a) — Algorithm Development (1 point). Describe what is being accomplished by the code in the body of the iteration (loop) statement — a summary of what the iteration does in the context of the program, or the purpose of the statements in the loop body. Do NOT award if: the procedure contains no iteration statement; the description does not match the code in the loop body; the response only restates the lines of code; the iteration is trivial; or the described behaviour is implausible/inconsistent with the program.

Solution 2

(b) Mark scheme

- Written Response 2(b) — Errors and Testing (1 point). Give two calls to the student-developed procedure where each call causes a DIFFERENT program code segment to execute, and describe the expected behaviour of each call — OR explain why it is not possible for two calls to cause different code segments to execute. The procedure must use an explicit parameter (defined in the header) whose value changes which code segment runs. A description of each call (rather than exact code) and a general description of argument values are acceptable. Do NOT award if: no procedure/explicit parameter is identified; the parameter is irrelevant; the two calls are to different procedures; or the described behaviour is inconsistent with the program.

Solution 2

(c) Mark scheme

- Written Response 2(c) — Data and Procedural Abstraction (1 point). Explain, in detailed steps, an algorithm (in code, pseudocode, or plain English) that uses the list to check whether all of its elements are valid, describing how each element of the identified list is passed into the validity check at least up to the first invalid element. Do NOT award if: no list is identified; the algorithm assumes the list contains a single element; the identified list is not referenced in the response; the response implements the validity check rather than describing its use; or the response is too vague for another programmer to recreate the algorithm.