

Task 7 Marking Key: Programming Practical Test

Task 7: Programming Practical Test - Unit 4 - 7.5%

Task 7 Marking Key: Programming Practical Test

Marking Notes

Award marks for evidence completed under supervised practical test conditions. The student does not need to produce a large program. Award marks for a small, clear, working or partially working solution that demonstrates the required programming concepts.

Where a response uses a different approved programming language or tool, mark the programming evidence against the criteria rather than against one exact syntax style.

Mark Allocation

Section A: Program Understanding And Design

Description	Expected response	Marks
Identifies the required inputs, processing and outputs from the scenario.	Inputs include name, theory minutes and practice minutes. Processing includes total time calculation, feedback decision and balanced-session count. Outputs include total time, feedback and final balanced count.	1-3
Selects suitable data types for the main values used by the program.	Name as text/string; minutes and counts as integer; balanced status or similar as Boolean if used. Accept equivalent types supported by the approved environment.	1-2
Uses meaningful variable names that suit the scenario and values stored.	Variable names communicate purpose, such as <code>theoryMinutes</code> , <code>practiceMinutes</code> , <code>totalMinutes</code> , <code>balancedCount</code> or equivalent.	1-2
Provides a clear algorithm plan using pseudocode, a flow chart or structured code comments.	Plan shows the main sequence, three-session loop, total calculation, feedback decision and counter update.	1-3
Subtotal		10

Section B: Program Construction

Description	Expected response	Marks
Constructs a program that accepts the required input values.	Program accepts student name, theory minutes and practice minutes for each session.	1-2
Constructs processing that correctly uses assignment, calculations or comparisons.	Program calculates total minutes and uses comparisons to classify the session.	1-3
Applies sequence so instructions run in a logical order.	Inputs, calculation, feedback, counter and final output occur in an order that supports the task.	1-2
Applies selection so the program makes at least one meaningful decision.	Program uses conditional logic to output short, balanced or long-session feedback.	1-3
Applies iteration so part of the program repeats appropriately.	Program repeats the session entry and classification for three sessions or an equivalent controlled loop.	1-3
Produces clear output that is useful for the scenario.	Program displays total time, feedback and final balanced-session count clearly.	1-2
Subtotal		15

Section C: Testing, Debugging And Documentation

Description	Expected response	Marks
Selects and applies suitable normal, boundary and/or invalid test data.	Evidence includes relevant valid test data and boundary values such as 30 and 60. Invalid or unexpected data is tested or identified as a limitation if validation is not built.	1-2
Uses a trace table or desk check to show how values change through part of the algorithm.	Trace table shows key variables changing correctly through at least one iteration or key decision.	1-2
Identifies or corrects a syntax, run-time or logical error.	Evidence shows an error found and corrected, or a clear written identification of an error and how it should be fixed.	1
Provides brief internal or external documentation explaining important code or logic.	Comments or written notes explain a meaningful part of the program, not just restate obvious code.	1
Subtotal		5
Total		30

Practical Judgement Guidance

Use professional judgement where the exact language syntax differs from the task notation. For example, a JavaScript `for` loop, Python `for` loop, Scratch repeat block or MakeCode loop can all demonstrate fixed iteration if the logic is correct.

Do not award construction marks for a decorative interface with no working programming logic. Do not penalise a plain text interface if it clearly meets the programming requirements.

