KK14 — Debugging and testing techniques for checking modules function correctly

Programming · Comprehensive exam study guide

STUDY DESIGN — YOU MUST KNOW

- use of breakpoints
- use of debugging statements
- construction of relevant test data
- test cases comparing expected and actual output in testing tables

This KK is about **proving a module works**. You must explain breakpoints and debugging statements, construct **relevant test data**, and build a **testing table** that compares expected and actual output — the cornerstone of testing questions.

Techniques

Technique	Purpose
Breakpoint	Pauses execution at a chosen line so the programmer can inspect variable values at that moment.
Debugging statement	A temporary output (e.g. print) showing a variable's value at a point in the code to trace behaviour.
Test data	Carefully chosen inputs used to check the module behaves correctly.
Testing table	A table listing each test: input, expected output, actual output, and pass/fail.

Relevant test data (three kinds)

- **Valid/typical** data — normal expected input (e.g. mark = 72).
- **Boundary** data — values at the edges of a range (e.g. 0 and 100, or 49 and 50 around a pass mark).
- **Invalid** data — input that should be rejected (e.g. mark = –5 or "abc").

Testing table

Test	Input	Expected	Actual	Result
Typical	mark = 72	Pass	Pass	✓
Boundary	mark = 50	Pass	Pass	✓
Invalid	mark = –5	Rejected	Rejected	✓

EXAM TIP

A complete answer always includes boundary AND invalid cases — not just typical input.

Key definitions to memorise

Breakpoint	A marker that pauses execution at a line so variables can be inspected.

Debugging statement	A temporary output added to display a variable's value and trace program behaviour.
Test data	Carefully selected inputs (valid, boundary and invalid) used to test a module.
Testing table	A table comparing expected and actual output for each test case to confirm correctness.

Exam technique

EXAM TIP

- When asked for test data, give valid, boundary AND invalid examples — each earns a mark.
- A testing table must compare **expected vs actual** — say this explicitly.

COMMON MISTAKES

- Giving only typical test data and omitting boundary/invalid cases.
- Confusing a breakpoint (pauses to inspect) with a debugging statement (prints a value).

Quick revision checklist

- ■ I can explain breakpoints and debugging statements.
- ■ I can construct valid, boundary and invalid test data.
- ■ I can draw a testing table comparing expected and actual output.

Exam practice — questions & model answers

Q1. Explain the purpose of a breakpoint.

Explain · 2 marks

Model answer

- It pauses execution at a chosen line (1)
- so the programmer can inspect variable values at that point to locate a fault (1).

Q2. A module accepts a percentage (0–100). Construct one valid, one boundary and one invalid test value.

Construct · 3 marks

Model answer

- Valid/typical: 75 (1).
- Boundary: 0 or 100 (1).
- Invalid: –1 or 150 (1).

Q3. Describe what a testing table is used for.

Describe · 2 marks

Model answer

- It records each test's input and expected output (1)
- and compares it with the actual output to confirm the module functions correctly (pass/fail) (1).