

12.3 Program Testing and Maintenance

Name: _____ Class: _____ Date: _____

Total: 18 marks

Objective

Build the skills to answer exam questions on **testing and maintenance** —error types, test data, testing methods and maintenance types.

You must be able to:

- identify **syntax**, **run-time** and **logic** errors
- choose **normal**, **boundary** and **abnormal** test data
- describe testing methods and the three types of **maintenance**

1 Worked examples

Study these first. Each one shows the method for a question type used later —follow the steps and you can do the Practice and Exam-style questions yourself.

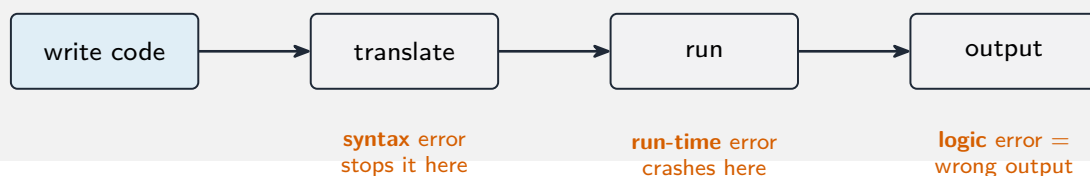
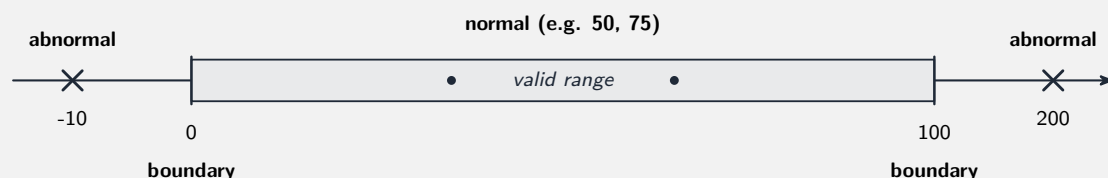
■ Types of error

Error	When	Example
syntax	at translation —won't run	missing ENDIF, misspelt keyword
run-time	while running —crashes	divide by zero, array index too big
logic	runs but wrong output	+ used for -, off-by-one loop

Logic errors are hardest —the only sign is a wrong result, so **trace** and **test** carefully.

■ Choosing test data

For a field that accepts **0 to 100**, test three kinds:



- **normal** —typical valid values the program should accept (50, 75)
- **extreme** —the **largest and smallest valid** values (0 and 100)
- **boundary** —a pair *either side* of a limit, where off-by-one bugs hide (100 accepted, 101 rejected)
- **abnormal** (erroneous) —values that must be **rejected** (-10, 200, "abc")

■ Methods and maintenance

- **dry run** —trace on paper in a **trace table**; **white-box** tests every path from the code; **black-box** tests inputs→outputs from the spec; **alpha** (in-house) then **beta** (real users); **acceptance** (the customer decides).
- Maintenance: **corrective** (fix bugs), **adaptive** (cope with a changed environment), **perfective** (improve features/speed).

2 Practice

Now apply the methods above.

2.1 State the difference between a **syntax error** and a **logic error**. [2]

2.2 Give **one** example of a **run-time** error. [1]

2.3 A field accepts an age from **1 to 120**. Give one **normal**, one **boundary** and one **abnormal** value. [3]

2.4 State what a **dry run** is. [1]

3 Exam-style questions

3.1 Identify the **type of error** in each case. [3]

(a) the program will not translate because an **ENDWHILE** is missing (b) the program stops with an error when it divides by zero (c) the program runs but the calculated average is always wrong

3.2 A field accepts a percentage **0–100**. Complete the test plan with suitable **test data** and a **reason** for each row. [4]

Type	Test data	Reason
normal		
boundary (low)		
boundary (high)		
abnormal		

3.3 State and describe the **three** types of maintenance. [3]

3.4 State what **regression testing** checks. [1]

4 Go further

You are now ready for the real exam questions on this subtopic. Open the **12.3 Program Testing and Maintenance** past-paper sheet in the Library, or try these in **Practice mode**:

- 9618/21 J25 —Q2 (errors and testing)
- 9618/23 N25 —Q6 (choosing test data)
- 9618/22 N24 —Q5 (testing methods)
- 9618/23 J25 —Q4 (testing and maintenance)

Solutions

2.1 A **syntax error** breaks the language's grammar and is caught at **translation** (the program won't run); a **logic error** lets the program run but produces the **wrong output**.

2.2 Any one: divide by zero; array index out of range; file not found; invalid type conversion.

2.3 Any valid set, e.g. normal 40; boundary 1 or 120; abnormal 0, 121 or "abc".

2.4 Tracing the code **by hand on paper**, recording each variable's value (usually in a trace table).

3.1 (a) **syntax** error; (b) **run-time** error; (c) **logic** error.

3.2 Any valid plan, e.g.: normal 50 (inside the range); boundary (low) 0 (lowest accepted); boundary (high) 100 (highest accepted); abnormal 150 or -5 (outside —must be rejected).

3.3 **Corrective** —fixing bugs found in use; **adaptive** —changing it to keep working in a new environment (new OS, law); **perfective** —improving features or performance.

3.4 That a **change has not broken** existing, previously-working features.