1.2 SOFTWARE & SOFTWARE DEVELOPMENT · 1.2.3

Writing & following algorithms

Tracing code with a **trace table** and writing algorithms in **OCR Exam Reference Language**. Spec 1.2.3(c).

01 Following (tracing)

Trace table	One column per variable + output; one row per change.
Loops	Record the loop variable on every pass.
for	`for i = 1 to 4` runs for 1,2,3,4 (inclusive).
Output	Only what is printed goes in the output column.

02 OCR pseudocode

Assign	`x = 5` · compare with ==
I/O	`print(...)` · `input(...)`
Select	`if..then..elseif..else..endif`
Loop	`for..next` · `while..endwhile`
Subs	`function..return..endfunction` · `procedure`

03 Handy built-ins & the marks examiners want

Strings	`text.length` · `text.substring(start, num)` · `ASC(c)` · `CHR(n)`
Maths	`a DIV b` (quotient) · `a MOD b` (remainder)
Write marks	correct loop · correct condition · counter/total updated · **return** the result

FINAL PASS BEFORE THE EXAM

Rapid exam tips

Seven slips on algorithm questions.

01

For "state the output", **draw a trace table** — don't do loops in your head.

02

OCR `for i = 1 to 4` is **inclusive** — it runs for 4 as well.

03

A function must **return** its result; printing is not returning.

04

= assigns; == compares. Don't mix them.

05

Index strings with `text.substring(i, 1)`; loop `0 to text.length - 1`.

06

Check letter ranges with ASCII: capitals are 65–90.

07

When writing an algorithm, test it on **edge cases** (empty input, first/last item, boundary values) before you move on.