2.1 COMPUTATIONAL THINKING · 2.1.3

Thinking procedurally

Components, sub-procedures & the order of steps. Spec 2.1.3(a)–(d).

01 Decomposition

Def	Break a problem into smaller, solvable **components**.
Keep going	Until each part = one routine.
≠ abstraction	Decompose = split; abstract = remove detail.

02 Sub-procedures

Def	Named routine doing **one part** of the job.
Why	Easier to write/test · reusable · team-friendly.
Name like	`checkStock`, `takePayment`.

03 Structure diagram

Shows	The **hierarchy** of decomposition.
Top	Whole problem.
Below	Sub-tasks → smaller sub-tasks.
aka	Hierarchy chart.

04 Order of steps

Why	Some steps **depend on** others.
Example	Can't take payment before checking stock.
Task	Work out **what must precede what**.

FINAL PASS BEFORE THE EXAM

Rapid exam tips

Six slips on procedural-thinking questions.

01

Name sub-procedures like **routines** (`checkStock`), each one job.

02

Structure diagram = **hierarchy**: top problem → sub-tasks → smaller ones.

03

State **dependencies**: which step must come before which, and why.

04

Decomposition = **split**; abstraction = **remove detail**. Don't mix.

05

Don't over/under-split — each box is **one clear task**.

06

"Complete the structure diagram" → add sensible **missing sub-procedures**.