A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

9.1 Computational Thinking Skills

Name: _____ Class: _____ Date: _____ **Total: 31 marks**

KEY WORDS decomposition 分解 • abstraction 抽象 • computational thinking 计算思维 • pattern recognition 模式识别 • loop 循环

Objective

Build the skills to answer exam questions on **computational thinking** 计算思维 — the two key skills of abstraction and decomposition.

You must be able to:

- explain and use **abstraction** 抽象
- explain and use **decomposition** 分解

1 Worked examples

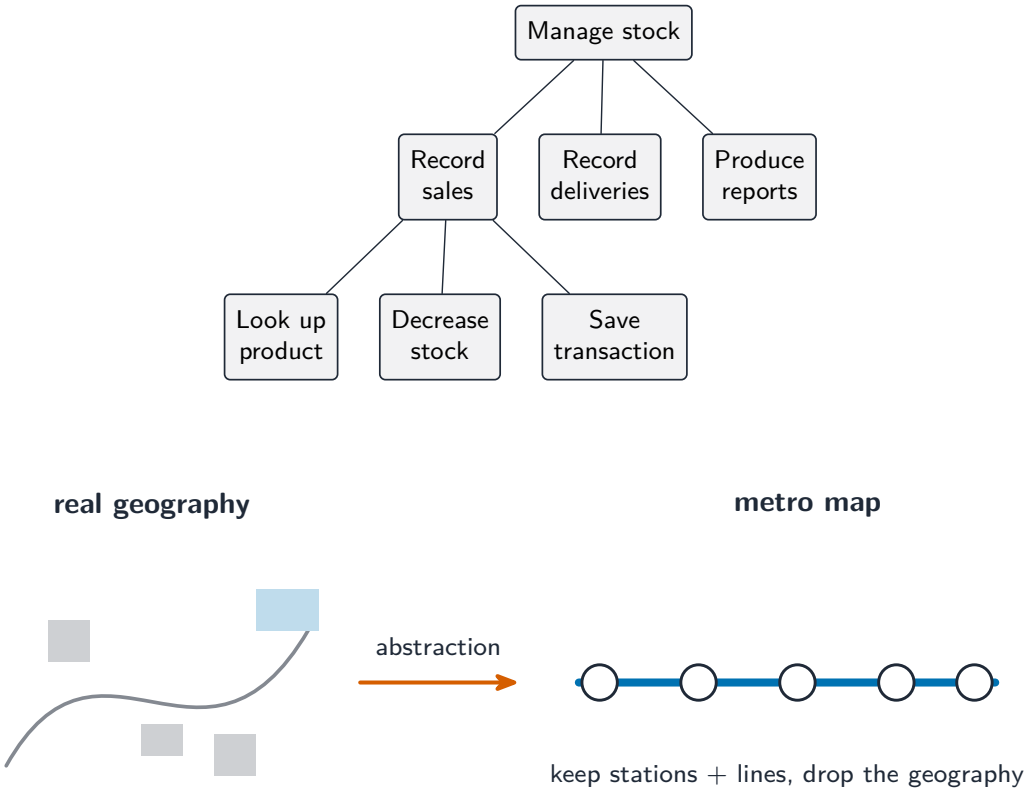
Study these first. Each one shows the method for a question type used later — follow the steps and you can do the Practice and Exam-style questions yourself.

■ Abstraction

Abstraction means keeping the **essential** features of a problem and **ignoring irrelevant detail**, to give a simpler model. Examples: a train-line map keeps the stations and lines but drops the real geography; a function hides a job behind a name; a class keeps only the attributes the system needs. Without abstraction, a full model of a real problem would be too big to reason about.

■ Decomposition

Decomposition means breaking a large problem into smaller **sub-problems**, each easier to solve and combined at the end:



Abstraction hides detail to manage complexity

It makes a big task manageable, lets a **team** divide the work, and produces **modular** code that is easier to test and maintain.

■ Pattern recognition

Pattern recognition 模式识别 means spotting where parts of a problem are the **same or similar**, so one solution can handle them all — for example, noticing that several jobs repeat lets you use a **loop** or reuse one **module** instead of writing the same code many times.

In the exam you may be asked to **identify which skill** a designer used: keeping only the needed detail → **abstraction**; splitting the problem into parts → **decomposition**; spotting repetition → **pattern recognition**.

2 Practice

Now apply the methods above.

2.1 State what **abstraction** means. [1]

2.2 State what **decomposition** means. [1]

2.3 Give one benefit of decomposing a problem. [1]

2.4 Give one everyday example of abstraction. [1]

3 Exam-style questions

3.1 Explain what is meant by **abstraction**, giving an example. [3]

3.2 A team must build a system to run a school library (lending and returning books, managing members, fines). Use **decomposition** to break this into **three** sub-tasks. [3]

3.3 Explain **two** benefits of decomposing a large program into modules. [2]

3.4 For each, state the computational thinking skill used: **(a)** a designer keeps only the data the system needs and ignores the rest; **(b)** a designer notices the same calculation is needed in five places and writes it once as a function. [2]

3.5 A shop's stock-control program is designed using **decomposition**. One module, `Sales()`, records a sale and updates the stock level.

(a) **Explain** what is meant by decomposition, and **give** two benefits of designing this program that way. [4]

(b) **Complete** the identifier table for `Sales()`. [4]

identifier	data type	description
`ItemCode`		
`QuantitySold`		
`UnitPrice`		
`InStock`		

(c) **State** the purpose of an identifier table at the **design** stage, and **explain** why it is written before any code. [3]

(d) The shop manager also wants a weekly report of items that fell below their reorder level. **Outline**, using **stepwise refinement**, the steps of a module `LowStockReport()`. Do **not** write pseudocode. [4]

(e) **Explain** why `Sales()` and `LowStockReport()` should be separate modules rather than one long block of code. [2]