A-LEVEL COMPUTER SCIENCE • EXERCISE SHEET

9.1 Computational Thinking Skills

Name: _____ Class: _____ Date: _____ **Total: 14 marks**

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

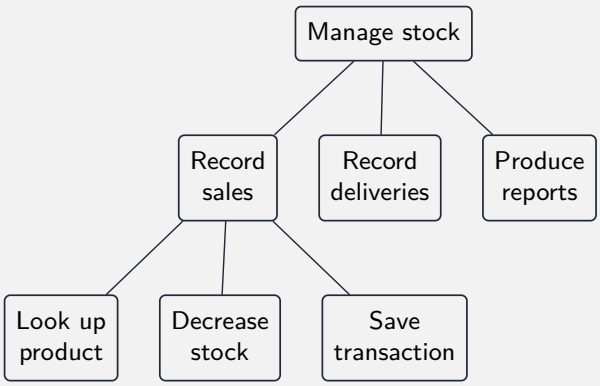
Study these first. Each one shows the method for a question type used later —follow the steps and you can do the Practice and Exam-style questions yourself.

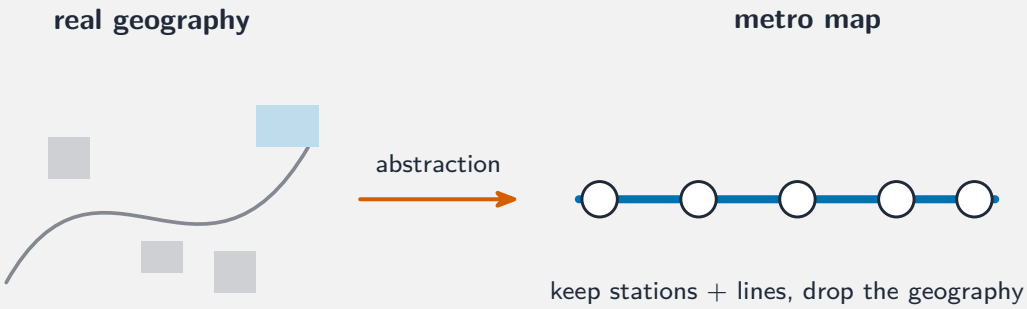
Abstraction

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]

4 Go further

You are now ready for the real exam questions on this subtopic. Open the **9.1 Computational Thinking** past-paper sheet in the Library, or try these in **Practice mode**:

- 9618/21 J24 —Q7 (decomposition and design)
- 9618/22 N24 —Q7 (abstraction and decomposition)
- 9618/22 J24 —Q7 (computational thinking skills)
- 9618/21 J25 —Q1 (identifier table and design)

Solutions

2.1 Keeping the **essential features** of a problem and **ignoring irrelevant detail** (a simpler model).

2.2 Breaking a large problem into smaller, easier **sub-problems**.

2.3 Any one: makes the problem manageable; lets a team share the work; gives modular, reusable, easier-to-test code.

2.4 Any one: a map; a function/method; a model/diagram that drops detail.

3.1 Abstraction keeps the **essential** features and removes irrelevant detail, giving a simpler model that is easier to work with; example, e.g. a metro map shows lines and stops but not real distances.

3.2 Any three sensible sub-tasks, e.g.: lend/return books; manage member records; calculate and record fines; search the catalogue.

3.3 Any two: smaller parts are easier to solve/test; a team can work in parallel; modules can be reused; maintenance is easier.

3.4 (a) **Abstraction**. (b) **Pattern recognition**.