Solutions

Each answer shows the steps, then the **result**.

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
- that gives a simpler model that is easier to work with
- example: 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**

3.5

(a) decomposition means breaking a problem into smaller sub-problems, each solved by its own module

- benefits, any two: each module can be written and tested independently, so faults are easier to find; modules can be reused elsewhere, and several people can work on different modules at once

(b) `ItemCode` — `STRING`, the unique code identifying the product

- `QuantitySold` — `INTEGER`, the number of units in this sale
- `UnitPrice` — `REAL` (or `CURRENCY`), the price of one unit

- `InStock` — `INTEGER`, the number of units remaining after the sale

(c) it lists every variable, its data type and what it is for

- so the programmer knows exactly what to declare and nobody invents a second name for the same thing
- writing it first forces the data to be thought through before the logic, which is where most design faults are cheapest to fix

(d) any sensible four steps, in order: read the reorder level and the stock level for each item in turn

- compare the two and select the items whose stock is below the level
- accumulate or format those items into a report, including code, description and quantity remaining
- output the report and record the date it was produced

(e) each does a **different job**, so keeping them separate means one can be changed or tested without disturbing the other

- it also allows the low-stock check to be called from more than one place, such as at the end of a sale and again weekly