

Exercise walk-through

9.1.1

Abstraction and decomposition in practice

A-Level Computer Science

You must be able to

- produce an abstract model of a system by including only the essential details
- decide which items of data a system or a module needs, and justify each choice
- identify the operations that a system must carry out on its data
- break a problem down into sub-problems, and name a program module for each one
- show a decomposition as a **structure diagram** 结构图

1

The method

Method — An abstract model: which data is essential?

For each item of data ask: *can the **purpose** be achieved without it?* Keep the item only if the answer is **no**, and justify it as “**needed to ...**”.

A parcel company stores each customer's name, address, phone number, date of birth and favourite colour. A new module must send a text message to every customer whose parcel arrives today.

item	needed?	justification
phone number	yes	needed to send the text message
name	yes	needed to address the message
date the parcel arrives	yes	needed to select today's parcels
date of birth	no	the message does not depend on age
favourite colour	no	nothing to do with a delivery

Method — An abstract model: which data is essential?

- A justification links the item to the **purpose**. “It is important” earns nothing.
- Data can be needed **but not stored**: today’s date comes from the computer’s clock.

Method — The operations of a model

A model = the **data** + the **operations** on it: **add**, **delete**, **search**, **update**, **calculate**, **produce** a list. Write a **verb** and its **data**: “produce a list of today’s deliveries”, not “deliveries”.

Method — Decomposing a problem into modules

- 1 Mark each separate **job** in the description (the verbs help).
- 2 Give each job a module with a **meaningful name**, and describe its **use** in one sentence.
- 3 Break a large job down again, and show the result as a **structure diagram**.

A school canteen wants an app: students see the menu, choose items and pay from their account; the kitchen gets a printed list of orders.

Method — Decomposing a problem into modules

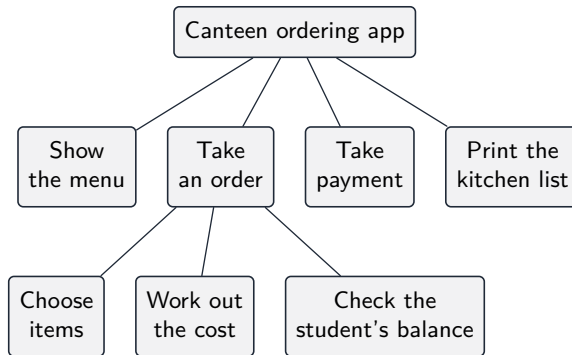
module	its use
ShowMenu	outputs today's items and their prices
TakeOrder	lets the student choose items, and works out the cost
TakePayment	takes the cost from the student's account
PrintKitchenList	outputs all the orders for the kitchen

Method — Decomposing a problem into modules

One module = **one job**. It **gives back one value** (the cost of an order) → a **function** 函数. It only **carries out an action** (printing the list) → a **procedure** 过程.

Method — Decomposing a problem into modules

A school canteen wants an app: students see the menu, choose items and pay from their account; the kitchen...



2

Practice

Practice 2.1 ■ 5 marks

A phone app shows passengers the time of the next bus from their stop. Tick (✓) **one** box in each row to show whether the item is essential for this purpose.

Item	essential	not essential
the number of the bus route		
the name of the bus driver		
the time the bus reaches the stop		
the colour of the bus		
the name of the stop		

Solution 2.1

Method: An abstract model: which data is essential? — p.4

Worked steps

- the purpose is “when does my bus come?”, so keep only what answers that question
- route number, arrival time and stop name are **essential**; the driver's name and the colour of the bus are **not essential** **B1** *for each correct row*

Practice 2.2 ■ 3 marks

A hotel uses a computer system for room bookings. State **three** items of data that must be stored for each booking.

Solution 2.2

Method: An abstract model: which data is essential? — p.4

Worked steps

any **three** sensible items, for example:

- the guest's name (or a guest ID) **B1**
- the room number **B1**
- the date of arrival **B1**
- the number of nights, or the date of leaving **B1**

Practice 2.3 ■ 2 marks

State **two** operations that the hotel's system must carry out on the booking data.

Solution 2.3

Worked steps

any **two**, for example:

- add a new booking **B1**
- delete (cancel) a booking **B1**
- search for a free room on a date **B1**
- update the dates of a booking **B1**

Practice 2.4 ■ 3 marks

Write each of these as an **operation** (a verb and the data it acts on). The first one is done for you.

not an operation	operation
late books	<i>produce a list of the books that are late</i>
new member	
a member's phone number has changed	
how much the fine is	

Solution 2.4

Method: The operations of a model — p.6

Worked steps

- new member → **add a new member's record** B1
- a changed phone number → **update a member's phone number** B1
- how much the fine is → **calculate the fine for a late book** B1

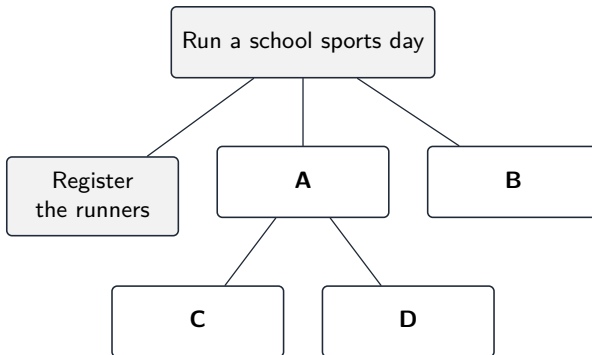
Practice 2.5 ■ 4 marks

A program will run a school sports day. The sub-problems are shown in the list. Complete the structure diagram by writing the letter of each empty box next to the correct sub-problem.

- Record the results of the races
- Record the time of each runner
- Work out the winner of each race
- Print the certificates

Practice 2.5 ■ 4 marks

The sub-problems are shown in the list.



Solution 2.5

Method: Decomposing a problem into modules — p.7

Worked steps

- “Record the results of the races” is the only sub-problem with two smaller jobs inside it, so it is the box with two boxes below it: **A** **B1**
- its two parts are “Record the time of each runner” and “Work out the winner of each race”: **C** and **D**, in either order **B1** **B1**
- “Print the certificates” is the last main job: **B** **B1**

Practice 2.6 ■ 2 marks

A module `RaceWinner` works out the winner of a race and gives the winner's name back to the program that called it. State whether it should be a procedure or a function. Give a reason.

Solution 2.6

Method: Decomposing a problem into modules — p.7

Worked steps

- a **function** B1
- it gives one value, the winner's name, back to the program that called it; a procedure does not return a value in this way B1

3

Exam-style

Exam-style 3.1 ■ 6 marks

A gym stores this information about each member: name, home address, email address, date of birth, date the membership ends, and favourite exercise class. A new module will send an email to every member whose membership ends this month, to remind them to renew it.

Exam-style 3.1 (a) ■ 3 marks

State **three** items of information that the new module needs.
Justify each choice.

Solution 3.1 (a)

Method: An abstract model: which data is essential? — p.4

Worked steps

one mark for each item with a justification linked to the purpose:

- the **email address**: needed to send the reminder (B1)
- the **date the membership ends**: needed to select the members whose membership ends this month (B1)
- the **name**: needed to address the email to the member (B1)

Exam-style 3.1 (b) ■ 2 marks

Identify **two** items of information that the new module does **not** need. **Justify** each choice.

Solution 3.1 (b)

Worked steps

any **two**, each with a justification:

- the home address: the reminder is sent by email, not by post **B1**
- the date of birth: a member's age has no effect on the reminder **B1**
- the favourite exercise class: it is not used to decide who gets a reminder, or what it says **B1**

Exam-style 3.1 (c) ■ 1 mark

Identify the computational thinking skill you used in parts (a) and (b).

Solution 3.1 (c)

Worked steps

abstraction B1

Exam-style 3.2 ■ 6 marks

A programmer is designing the program for a payment machine in a car park. A driver puts in a ticket, the machine works out the charge from the time parked, the driver pays by card, and the machine gives the ticket back so that the driver can leave.

Exam-style 3.2 (a) ■ 3 marks

Decomposition is used to break the problem down. Identify **three** program modules that could be used in the design, and describe the use of each one.

Solution 3.2 (a)

Method: Decomposing a problem into modules — p.7

Worked steps

one mark for each sensible module with its use, for example:

- ReadTicket: reads the time of arrival from the ticket (B1)
- CalculateCharge: works out the charge from the time parked (B1)
- TakePayment: takes the payment from the driver's card (B1)
- ReturnTicket: marks the ticket as paid and gives it back (B1)

Exam-style 3.2 (b) ■ 1 mark

One module works out the charge and gives it back to the main program. Identify the type of program module that should be used.

Solution 3.2 (b)

Worked steps

a **function** B1

Exam-style 3.2 (c) ■ 2 marks

Three programmers will write the program. Describe **two** benefits of decomposition for this team.

Solution 3.2 (c)

Worked steps

any **two** of:

- each programmer can write a different module at the same time, so the program is finished sooner **B1**
- each module is small, so it is easier to understand and to write **B1**
- each module can be tested and corrected on its own **B1**

Exam-style 3.3 ■ 6 marks

A restaurant wants a system that lets customers order food to be delivered to their home. An abstract model of the system is needed.

Exam-style 3.3 (a) ■ 2 marks

Explain the process of abstraction.

Solution 3.3 (a)

Worked steps

the details that are not needed for the problem are filtered out **B1**

- so that only the essential details are kept, which gives a simpler model of the system **B1**

Exam-style 3.3 (b) ■ 2 marks

State **four** items of data that should be stored for each order.

Solution 3.3 (b)

Worked steps

one mark for **two** sensible items, two marks for **four**, for example: the customer's name; the delivery address; the phone number; the items ordered; the cost of the order; the time of the order **B1** **B1**

Exam-style 3.3 (c) ■ 2 marks

Identify **two** operations that would be needed to process the order data.

Solution 3.3 (c)

Worked steps

any **two**, for example:

- add a new order **B1**
- calculate the total cost of an order **B1**
- update the state of an order, for example to “delivered” **B1**
- produce a list of the orders that are waiting **B1**

Exam-style 3.4 ■ 10 marks

A vet (an animal doctor) wants a program to manage appointments. The vet knows this about each animal: its name, its type (dog, cat, ...), its age, its owner's name, its owner's phone number, its colour, its favourite toy, and the dates of its past visits.

A new module must produce a list of tomorrow's appointments. For each appointment the list shows the time, the animal and how to contact the owner.

Exam-style 3.4 (a) ■ 1 mark

State the purpose of applying abstraction to this problem.

Solution 3.4 (a)

Method: An abstract model: which data is essential? — p.4

Worked steps

to remove the details that the module does not need, so that the problem is simpler and the module contains only essential data **B1**

Exam-style 3.4 (b) ■ 3 marks

Tick (✓) **one** box in each row to show whether the item is essential for the new module.

Item	essential	not essential
the animal's name		
the animal's favourite toy		
the owner's phone number		
the animal's colour		
the date and time of the appointment		
the dates of past visits		

Solution 3.4 (b)

Worked steps

the list shows the time, the animal and how to contact the owner, so:

- **essential**: the animal's name, the owner's phone number, the date and time of the appointment
- **not essential**: the favourite toy, the colour, the dates of past visits
- one mark for each **two** correct rows **B1** **B1** **B1**

Exam-style 3.4 (c) ■ 1 mark

State **one** item of data that the module needs but that is **not** stored for each appointment.

Solution 3.4 (c)

Worked steps

tomorrow's date (or today's date): it comes from the computer's clock, and it is used to select the appointments **B1**

Exam-style 3.4 (d) ■ 3 marks

The task *manage appointments* is decomposed. Identify **three** sub-problems, and describe each one.

Solution 3.4 (d)

Worked steps

one mark for each sensible sub-problem with a description, for example:

- make an appointment: store the animal, the date and the time **B1**
- cancel or change an appointment **B1**
- produce the list of tomorrow's appointments **B1**
- search for a free time on a given day **B1**

Exam-style 3.4 (e) ■ 2 marks

Explain why each sub-problem should be written as a separate module, not as part of one long program.

Solution 3.4 (e)

Worked steps

any **two** of:

- a small module is easier to write, to test and to correct than one long program **B1**
- a module can be changed later without changing the rest of the program **B1**
- a module can be reused, for example “search for a free time” is used when an appointment is made and when it is changed **B1**