Exercise walk-through

9.1.1 Abstraction and decomposition in practice

A-Level Computer Science

Sheet 9.1 – what you need to know

You must be able to:

- explain abstraction and decomposition, and name the skill used
- draw and read a structure chart, with parameters
- complete a state-transition table

What the exam asks

"Draw a structure chart for the three modules, including all parameters and return values." **[3]**

"Explain the meaning of the diamond symbol in the chart." **[2]**

"Complete the table to show the inputs, outputs and next states." **[5]**

Practise on exercise sheet 9.1, then try the real exam questions on past-paper sheet 9.1.

Computational thinking

Computational thinking is the set of mental tools for analysing a problem and designing a solution a computer can run. Two of the key tools:

Abstraction

keep the essential features, ignore the irrelevant detail

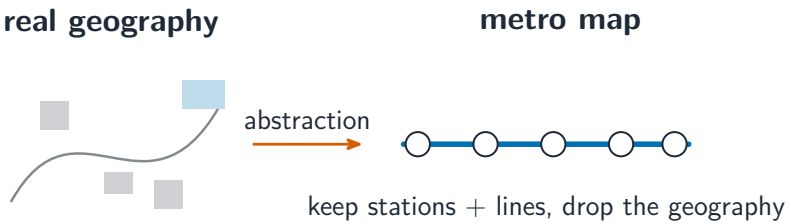
Decomposition

break a large problem into smaller parts, each easier to solve

Abstraction

Abstraction means keeping the essential features of a problem and ignoring the irrelevant detail, giving a simpler model.

- a **train-network map** keeps the stations and lines but drops the geography
- a **class** keeps only the attributes and methods a system needs
- a **function** hides a piece of work behind a name



A train-network map

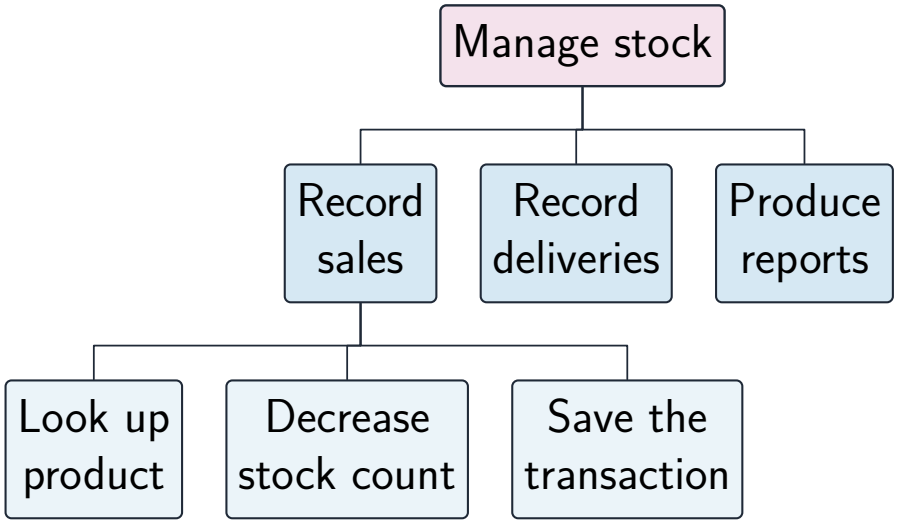
> A full model of a real problem is too big to reason about, so abstraction is essential.

Decomposition

Decomposition means breaking a large problem into smaller sub-problems, each tackled one at a time.

1. find the main parts of the task
2. break each into smaller sub-tasks
3. continue until each is small enough to design directly
4. solve the small tasks and combine them

Each module becomes a **procedure** or function.



Stock control, broken into modules and sub-tasks

Explain why decomposition is used: three marks

Three marks need three different benefits:

1. each **sub-problem** is small enough to **design, code and test on its own**
2. different programmers can work on different **modules** **at the same time**
3. an existing module or library routine can be **reused**, and a fault is **easier to find** because it lies inside one module

A **structure chart** (topic 12) is the diagram of a decomposition: the program at the top, its modules beneath, and the data passed between them.

Exercise 2.1

2.1 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. [5]



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		

Exercise 2.1

2.1 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. [5]



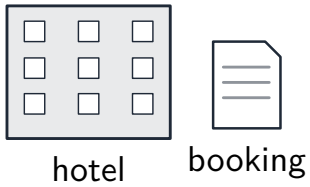
Item	essential	not essential
the number of the bus route		
the name of the bus driver		
the time the bus reaches the stop		
the colour of the bus		
the name of the stop		

2.1

- 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**

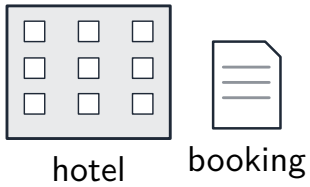
Exercise 2.2

2.2 A hotel uses a computer system for room bookings. State **three** items of data that must be stored for each booking. [3]



Exercise 2.2

2.2 A hotel uses a computer system for room bookings. State **three** items of data that must be stored for each booking. [3]



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

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

Exercise 2.3

2.3 State **two** operations that the hotel's system must carry out on the booking data. [2]



bookings

Exercise 2.3

2.3 State **two** operations that the hotel's system must carry out on the booking data. [2]



bookings

2.3 any **two**, for example:

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

Exercise 2.4

2.4 Write each of these as an **operation** (a verb and the data it acts on). The first one is done for you. [3]



library

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

Exercise 2.4

2.4 Write each of these as an **operation** (a verb and the data it acts on). The first one is done for you. [3]



library

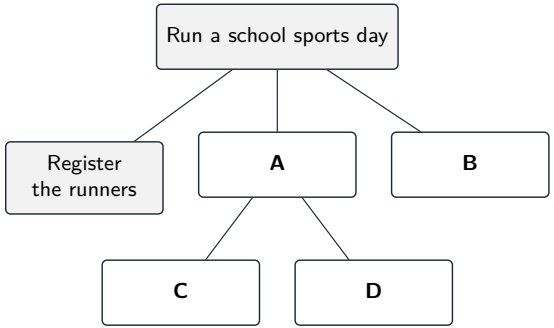
not an operation	operation
late books	*produce a list of the books that are late*
new member	
a member's phone number has changed	
how much the fine is	

2.4

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

Exercise 2.5

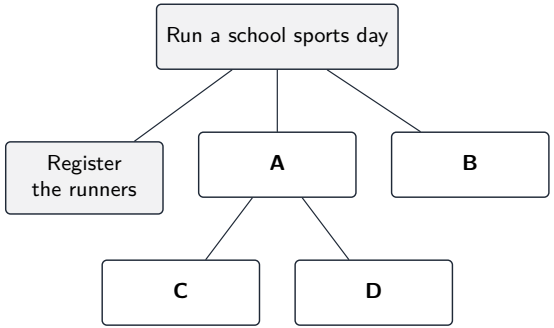
2.5 A program will run a school sports day. The sub-problems are shown in the list. Complete the structure diagram by writing the letter of each empty box next to the correct sub-problem. [4]



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

Exercise 2.5

2.5 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. [4]



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

2.5

- "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**

Full solution on next slide

Full solution 2.5

2.5

- "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**
- its two parts are "Record the time of each runner" and "Work out the winner of each race": **C** and **D**, in either order
- "Print the certificates" is the last main job: **B**

Exercise 2.6

2.6 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. [2]



Exercise 2.6

2.6 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. [2]



2.6

- a **function**
- 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

Exercise 3.1 (a)

3.1 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.

(a) State **three** items of information that the new module needs. **Justify** each choice. [3]



gym



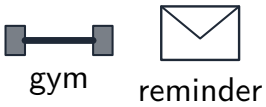
reminder

Exercise 3.1 (a)

3.1 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.

(a) State **three** items of information that the new module needs. **Justify** each choice. [3]



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

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

Exercise 3.1 (b)

3.1 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.

(b) Identify **two** items of information that the new module does **not** need. **Justify** each choice. [2]

Exercise 3.1 (b)

3.1 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.

(b) Identify **two** items of information that the new module does **not** need. **Justify** each choice. [2]

(b) any **two**, each with a justification:

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

Exercise 3.1 (c)

3.1 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.

(c) Identify the computational thinking skill you used in parts (a) and (b). [1]

Exercise 3.1 (c)

3.1 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.

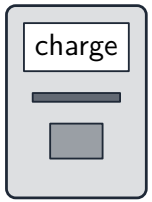
(c) Identify the computational thinking skill you used in parts (a) and (b). [1]

(c) **abstraction**

Exercise 3.2 (a)

3.2 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.

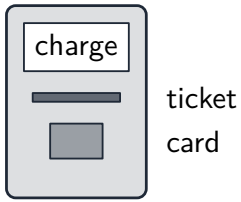
(a) Decomposition is used to break the problem down. Identify **three** program modules that could be used in the design, and describe the use of each one. [3]



Exercise 3.2 (a)

3.2 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.

(a) 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. [3]



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

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

Exercise 3.2 (b)

3.2 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.

(b) One module works out the charge and gives it back to the main program. Identify the type of program module that should be used. [1]

Exercise 3.2 (b)

3.2 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.

(b) One module works out the charge and gives it back to the main program. Identify the type of program module that should be used. [1]

(b) a **function**

Exercise 3.2 (c)

3.2 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.

(c) Three programmers will write the program. Describe **two** benefits of decomposition for this team. [2]

Exercise 3.2 (c)

3.2 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.

(c) Three programmers will write the program. Describe **two** benefits of decomposition for this team. [2]

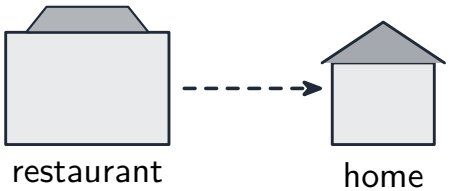
(c) any **two** of:

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

Exercise 3.3 (a)

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

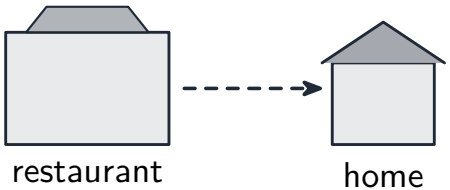
(a) **Explain** the process of abstraction. [2]



Exercise 3.3 (a)

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

(a) **Explain** the process of abstraction. [2]



(a) the details that are not needed for the problem are filtered out

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

Exercise 3.3 (b)

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

(b) State **four** items of data that should be stored for each order. [2]

Exercise 3.3 (b)

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

(b) State **four** items of data that should be stored for each order. [2]

(b) 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

Exercise 3.3 (c)

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

(c) Identify **two** operations that would be needed to process the order data. [2]

Exercise 3.3 (c)

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

(c) Identify **two** operations that would be needed to process the order data. [2]

(c) any **two**, for example:

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

Exercise 3.4 (a)

3.4 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.

(a) State the purpose of applying abstraction to this problem. [1]



Exercise 3.4 (a)

3.4 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.

(a) State the purpose of applying abstraction to this problem. [1]



(a) to remove the details that the module does not need, so that the problem is simpler and the module contains only essential data

Exercise 3.4 (b)

3.4 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.

(b) Tick (✓) **one** box in each row to show whether the item is essential for the new module. [3]

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		

Exercise 3.4 (b)

3.4 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.

(b) Tick (✓) **one** box in each row to show whether the item is essential for the new module. [3]

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		

(b) 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

Full solution on next slide

Full solution 3.4 (b)

(b) 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

Exercise 3.4 (c)

3.4 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.

(c) State **one** item of data that the module needs but that is **not** stored for each appointment. [1]

Exercise 3.4 (c)

3.4 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.

(c) State **one** item of data that the module needs but that is **not** stored for each appointment. [1]

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

Exercise 3.4 (d)

3.4 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.

(d) The task *manage appointments* is decomposed. Identify **three** sub-problems, and describe each one. [3]

Exercise 3.4 (d)

3.4 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.

(d) The task *manage appointments* is decomposed. Identify **three** sub-problems, and describe each one. [3]

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

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

Exercise 3.4 (e)

3.4 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.

(e) **Explain** why each sub-problem should be written as a separate module, not as part of one long program. [2]

Exercise 3.4 (e)

3.4 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.

(e) **Explain** why each sub-problem should be written as a separate module, not as part of one long program. [2]

(e) any **two** of:

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