# Past-paper walk-through

## Computational Thinking Skills ▪ 9.1

eXercise

# Questions in this set

- 9618/21 J25 Q1 ▪ 12 marks
- 9618/22 N24 Q7 ▪ 10 marks
- 9618/23 N24 Q7 ▪ 9 marks
- 9618/21 J24 Q7 ▪ 17 marks
- 9618/22 J24 Q7 ▪ 9 marks

# Reading the mark scheme

- **B1** a mark that stands on its own – the point is either made or not.
- **M1** a *method* mark: the right approach, even if the number is wrong.
- **A1** an *answer* mark, and it usually needs its M mark first.
- **C1** a working mark on the way to the answer.
- // separates answers the examiner accepts equally.
- **ecf** = error carried forward: a later part is marked on your own earlier answer.
- **owtte** = "or words to that effect" – the wording need not match.

# Question 1

9618/21 J25 ▪ Q1 ▪ 12 marks

A program is being developed to help the manager of a shop control the stock.

**(a)** An identifier table has been used during the design stage.

Complete the identifier table:

| Example value | Explanation | Variable name | Data type |
|---|---|---|---|
| `"Fruit"` | a category of stock that is sold in the shop | | |
| 20/02/2025 | when an item was sold | | |
| `12.67` | the cost of an item | | |
| `TRUE` | to indicate if an item is in stock | | |

# Question 1

**[4]**

# Question 1

| Pseudocode extract | Assignment | Selection | Iteration |
|---|---|---|---|
| `Result ← CalculateTotal()` | | | |
| `WHILE IsClosed` | | | |
| `REPEAT`<br>`  INPUT Value`<br>`UNTIL Sales[4] > Value` | | | |
| `IF Sales[Current] <= 150 THEN`<br>`  Discount ← TRUE`<br>`ENDIF` | | | |
| `CASE OF Option` | | | |

[5]

# Question 1

| Example value | Explanation | Variable name | Data type |
|---|---|---|---|
| `"Fruit"` | a category of stock that is sold in the shop | | |
| `20/02/2025` | when an item was sold | | |
| `12.67` | the cost of an item | | |
| `TRUE` | to indicate if an item is in stock | | |

[4]

# Question 1

9618/21 J25 ▪ Q1 ▪ 12 marks

A program is being developed to help the manager of a shop control the stock.

**(b)** A module `Sales()` is part of the stock control program.

The table contains pseudocode extracts from the module `Sales()`

Each extract may include all or part of:

- assignment
- selection
- iteration (repetition).

# Question 1

Complete the table by placing one or more ticks (✓) in each row:

| Pseudocode extract | Assignment | Selection | Iteration |
|---|---|---|---|
| `Result <- CalculateTotal()` | | | |
| `WHILE IsClosed` | | | |
| `REPEAT`<br>`INPUT Value`<br>`UNTIL Sales > Value` | | | |
| `IF Sales[Current] <= 150 THEN`<br>`Discount <- TRUE`<br>`ENDIF` | | | |
| `CASE OF Option` | | | |

# Question 1

**[5]**

# Question 1

9618/21 J25 ▪ Q1 ▪ 12 marks

A program is being developed to help the manager of a shop control the stock.

**(c)** Decomposition has been used to design the program to help the shop manager control the stock.

Describe decomposition. **[3]**

# Solution 1

## (a) Worked steps

Two marks per idea are on offer here — the **name** and the **type** — so read each example value for what it physically is, not for what the sentence beside it is about.

| Example value | Variable name | Data type |
|---|---|---|
| "Fruit" | Category | STRING |
| 20/02/2025 | DateSold | DATE |
| 12.67 | ItemCost | REAL |
| TRUE | ItemInStock | BOOLEAN |

# Solution 1

Two things decide the marks: a name that says what the value **is** (`DateSold`, not `Date1`), and `REAL` rather than `INTEGER` for money, because 12.67 has a fractional part. A date has its own type in this syllabus — do not call it a string. **Mark scheme**

- Example value
- Explanation
- Variable name
- Data type
- "Fruit" a category of stock that is sold in the shop
- Category
- STRING
- 20/02/2025 when an item was sold
- DateSold
- DATE
- 12.67 the cost of an item

# Solution 1

- (Item)Cost/Price
- REAL
- TRUE to indicate if an item is in stock
- (Item)InStock
- BOOLEAN
- Mark as follows:
- 1 mark per row for each variable name and data type

# Solution 1

## (b) Worked steps

Classify each extract by what the **statement itself** does, and remember a row may need more than one tick.

| Extract | Ticks |
|---|---|
| `Result <- CalculateTotal()` | Assignment |
| `WHILE IsClosed` | Iteration |
| `REPEAT … INPUT Value … UNTIL Sales[4] > Value` | Assignment (the INPUT) and Iteration |
| `IF Sales[Current] <= 150 THEN Discount <- TRUE ENDIF` | Assignment and Selection |
| `CASE OF Option` | Selection |

# Solution 1

The two rows that carry the extra marks are the ones doing **two** things at once: an input inside a loop, and an assignment inside an IF. **Mark scheme**

- Pseudocode extract
- Assignment
- Selection
- Iteration
- Result <- CalculateTotal()
- ✓
- WHILE IsClosed
- ✓
- REPEAT
- INPUT Value
- UNTIL Sales[4] > Value

# Solution 1

- ✓
- ✓
- IF Sales[Current] <= 150 THEN
- Discount <- TRUE
- ENDIF
- ✓
- ✓
- CASE OF Option
- ✓
- Mark as follows:
- One mark for each correct row

# Solution 1

**(c) Worked steps**

Three marks means three separate points, and at least one must be about **this** program.

- Decomposition means breaking a problem down into smaller sub-problems.
- Each sub-problem is then understood and solved on its own — for the stock control program, separate parts for recording a sale, updating stock levels and reordering.
- Each becomes its own module (procedure or function), which makes the program easier to test and debug.

# Solution 1

A general answer that never mentions stock control is capped at two marks, so name the program's own modules. **Mark scheme**

- Decomposition involves:
- MP1
- Breaking down a problem into sub problems
- MP2
- In order to explain / understand the process/task of how a (stock control) can be managed in more detail
- MP3

# Solution 1

- Leading to the concept of the program designed as a series of modules
- / procedures / functions
- // or by example of different stock control modules
- MP4
- Makes the program easier to test / debug
- Max 3
- Max 2 if no mention of stock control

# Question 7

9618/22 N24 ▪ Q7 ▪ 10 marks

A coffee shop runs a computerised loyalty card system.
Customers are issued with a loyalty card with their name together with a unique customer ID.
Loyalty points are added to their card each time they spend money at the shop.
The following information is stored for each customer: ID, name, home address, email address, mobile phone number, date of birth, number of points, date of last visit and amount of money spent at last visit.
A new module will generate a personalised email message to each loyalty card customer who has not visited the coffee shop in the last three months. The message will include a unique voucher code which can be used to authorise a discount if the customer goes to the shop within the next two weeks.

# Question 7

**(a)** Identify **three** items of customer information that will be used by the new module **and** justify your choices.

Item 1

Item 2

Item 3 **[3]**

# Question 7

9618/22 N24 ▪ Q7 ▪ 10 marks

A coffee shop runs a computerised loyalty card system.
Customers are issued with a loyalty card with their name together with a unique customer ID.
Loyalty points are added to their card each time they spend money at the shop.
The following information is stored for each customer: ID, name, home address, email address, mobile phone number, date of birth, number of points, date of last visit and amount of money spent at last visit.

A new module will generate a personalised email message to each loyalty card customer who has not visited the coffee shop in the last three months. The message will include a unique voucher code which can be used to authorise a discount if the customer goes to the shop within the next two weeks.

**(b)** Identify the computational thinking skill that you needed to use to answer part (a). **[1]**

# Question 7

9618/22 N24 ▪ Q7 ▪ 10 marks

A coffee shop runs a computerised loyalty card system.
Customers are issued with a loyalty card with their name together with a unique customer ID.
Loyalty points are added to their card each time they spend money at the shop.
The following information is stored for each customer: ID, name, home address, email address, mobile phone number, date of birth, number of points, date of last visit and amount of money spent at last visit.
A new module will generate a personalised email message to each loyalty card customer who has not visited the coffee shop in the last three months. The message will include a unique voucher code which can be used to authorise a discount if the customer goes to the shop within the next two weeks.

**(c)** It is decided to adopt a formal program development life cycle model for the development of the new module.

# Question 7

The analysis of the new module is complete and the project moves on to the design stage. During this stage all the necessary algorithms and module designs will be defined.

State **three other** items that will be defined for the new module during the design stage.

1. 2. 3. **[3]**

# Question 7

9618/22 N24 ▪ Q7 ▪ 10 marks

A coffee shop runs a computerised loyalty card system.
Customers are issued with a loyalty card with their name together with a unique customer ID.
Loyalty points are added to their card each time they spend money at the shop.
The following information is stored for each customer: ID, name, home address, email address, mobile phone number, date of birth, number of points, date of last visit and amount of money spent at last visit.

A new module will generate a personalised email message to each loyalty card customer who has not visited the coffee shop in the last three months. The message will include a unique voucher code which can be used to authorise a discount if the customer goes to the shop within the next two weeks.

**(d)** Part of the coffee shop program contains three program modules as follows:

- Module `Init()` has no parameters and returns a Boolean.

# Question 7

- Module `Reset()` takes a string as a parameter and returns an Integer.
- Module `Check()` repeatedly calls `Init()` followed by `Reset()`.

Draw a structure chart to represent the relationship between the **three** modules, including all parameters and return values. **[3]**

# Solution 7

## (a) Mark scheme

- ▪ Customer ID – to reference the other customer details
- ▪ Email address – to send the email
- ▪ Name – to personalise the email
- ▪ Date of last visit – to select which customers should receive an email
- ▪ Unique voucher code (or method of code generation) – to include in the email
- Mark as follows:
- One mark per item and justification
- Max 3 marks

# Solution 7

**(b)**

**Mark scheme**

- Abstraction

# Solution 7

**(c) Worked steps**

The question already names algorithms and module designs, so those two are off the table — "three **other** items" means three more. Think about what a programmer would still be unable to start coding without.

- **Data structures** — the arrays, records and files the program will use. Equivalently a **data dictionary** or **identifier table**, and the **validation rules** each field must satisfy.
- **Data-flow diagrams** or **state-transition diagrams** — how data moves between the parts of the system, and how the system changes state.
- The **user interface** — the screen layouts, and here also the **format of the email** the module sends.

# Solution 7

One mark each. The trap is naming things from the wrong stage: a requirement specification belongs to **analysis**, and test data and test plans belong to **testing**. Both are common answers and neither scores.

A useful test before writing an item down: could a programmer produce this *before* knowing what the system must do (too early, that is analysis), or only *after* the code exists (too late, that is testing)? If neither, it is a design item. **Mark scheme**

---

- MP1 Data structures // data dictionary // identifier table(s) // validation rules
- MP2 Data-flow diagram // state-transition diagram
- MP3 User interface // Format for the email
- MP4 Testing method / Test plan / Test data / Trace tables

# Solution 7

- MP5 Choice of email protocol to be used // Programming language to be used // Development environment
- MP6 Use of library routines // program to send the email
- Max 3 marks

# Solution 7

## (d) Worked steps

Turn each line of the specification into a feature of the chart, in order.
**The hierarchy.** `Check()` calls the other two, so it is the root, with `Init()` and `Reset()` below it. Nothing calls `Check()`, and neither of the others calls anything.
**The arrows.** Every parameter and return value is a labelled arrow on the line joining a module to its caller, pointing the way the data travels.

- `Init()` has no parameters and returns a Boolean: one arrow **up**, labelled Boolean, and no arrow down.
- `Reset()` takes a string and returns an integer: an arrow **down** labelled with the string and one **up** labelled with the integer.

# Solution 7

**The annotation.** `Check()` calls them **repeatedly**, one after the other — that is **iteration**, drawn as a curved arrow looping round the lines to both modules. Put `Init()` to the left of `Reset()`, because in a structure chart left-to-right order is the order of execution and the specification says `Init()` comes first.
Three marks, three features: the hierarchy, the parameters and return values, and the iteration. Boxes with no arrows score one of the three, and a module with no parameters still needs its return arrow drawn. **Mark scheme**

---

- MP1
- Three boxes correctly labelled and correct hierarchy
- MP2

# Solution 7

- Parameter and return values
- MP3
- Iteration arrow
- 3
- October/November
- 2024

# Question 7

9618/23 N24 ▪ Q7 ▪ 9 marks

**7** A coffee shop owner wants to introduce a computerised loyalty card system.

A programmer discusses the details of the system with the shop owner.

**(a)** Identify the stage of the program development life cycle that this discussion is part of **and** give a document that will be produced during this stage.

# Question 7

# Question 7

# Question 7

[2]

**(b)** The shop will give each customer a loyalty card that displays a unique customer ID as a bar code. A customer will be able to present their card each time they make a purchase. The system will scan the bar code, calculate points, and add them to the customer's total. When the customer next makes a purchase and presents their card, they will have the option to exchange points for a discount.

The designer decides that this activity will be handled by a new module. Decomposition will be used to break the problem of designing the new module down into sub-problems (sub-modules).

Identify **four** sub-modules that could be used in the design of the new module **and** describe their use.

# Question 7

# Question 7

# Question 7

# Question 7

# Question 7

# Question 7

# Question 7

# Question 7

# Question 7

# Question 7

# Question 7

# Question 7

[4]

# Solution 7

## (a) Worked steps

The programmer is **finding out what the system must do** by talking to the person who wants it. That is **analysis** — the first stage, before any design decision is made. One mark for the stage, one for a document it produces:

- A **requirement specification**
- A **definition of system objectives**
- A **list of problems with the existing system**
- **Survey results**
- A **feasibility study**

# Solution 7

The distinction being tested is between analysis and design, and the test is simple: analysis asks **what** the system must do, design decides **how** it will do it. A discussion with the shop owner about how the loyalty scheme should work is entirely the first. A test plan and a program specification are the usual wrong answers. Both come later, once the requirements exist. **Mark scheme**

---

- One mark per answer:
- MP1 Analysis
- MP2 Named document, examples include:
- ▪
- A requirement specification

# Solution 7

- ▪
- Definition of System Objectives
- ▪
- List of problems with existing system
- ▪
- Survey results
- ▪
- Feasibility study
- ▪
- Interview notes

# Solution 7

- Max 2 marks
- 2
- 9618/23
- October/November
- 2024

# Solution 7

**(b) Worked steps**

Decomposition means breaking the task into sub-modules that each do **one identifiable job**, and each mark is for a **name plus its use** — so give both halves. Sub-modules that fall naturally out of a loyalty-card transaction:

- `ScanCard()` or `GetID()` — read the barcode and obtain the customer's ID.
- `GetPoints()` — look up how many points that customer currently has.
- `CalculatePoints()` — work out the points earned by this purchase.
- `ExchangePoints()` or `UpdateCard()` — reduce the stored points when they are spent.
- `CalculateDiscount()` or `GetDiscount()` — work out the discount those points buy.

# Solution 7

Two habits get the marks. **Name each module as a verb phrase** — a module does something, and a name like `Card()` says nothing about what. And make each one **a single job**: a `DoEverything()` that scans, calculates and updates is not a decomposition, and its use cannot be stated in one line.
A quick check on a proposed set: could each module be written and tested on its own, by somebody who knows only its name and its parameters? If not, it has not been decomposed far enough. **Mark scheme**

---

- One mark for name and use
- Mark as follows:
- Examples include:

# Solution 7

- Sub-module: ScanCard()/ GetID()
- Use: Read the barcode from the loyalty card / Get the customer ID from the
- barcode
- Sub-module: GetPoints()
- Use: Get the number of points the customer has
- Sub-module: ExchangePoints() / UpdateCard()
- Use: Reduce the number of loyalty points
- Sub-module: GetDiscount() / CalculateDiscount()
- Use: Calculate the discount
- Sub-module: CalculatePoints()

# Solution 7

- Use: Calculate points (following a purchase)
- Sub-module: UpdatePoints()
- Use: Update total points a customer has (following a purchase)
- Sub-module: ShowDiscount()
- Use: Display the discount
- Max 4 marks
- 4
- 9618/23
- October/November
- 2024

# Question 7

9618/21 J24 ▪ Q7 ▪ 17 marks

**7** A fitness club has a computerised membership system. The fitness club offers a number of different exercise classes.

The following information is stored for each club member: name, home address, email address, mobile phone number, date of birth and the exercise(s) they are interested in.

**(a)** When an exercise class is planned, a new module will send personalised text messages to each member who has expressed an interest in that exercise. Members wishing to join the class send a text message back. Members may decide **not** to receive future text messages by replying with the message 'STOP'.

The process of abstraction is used to filter out unnecessary information.

**(i)** State **one** advantage of applying abstraction to this problem.

# Question 7

**(ii)** Identify **three** items of information that will be required by the new module. Justify your choices with reference to the given scenario.

# Question 7

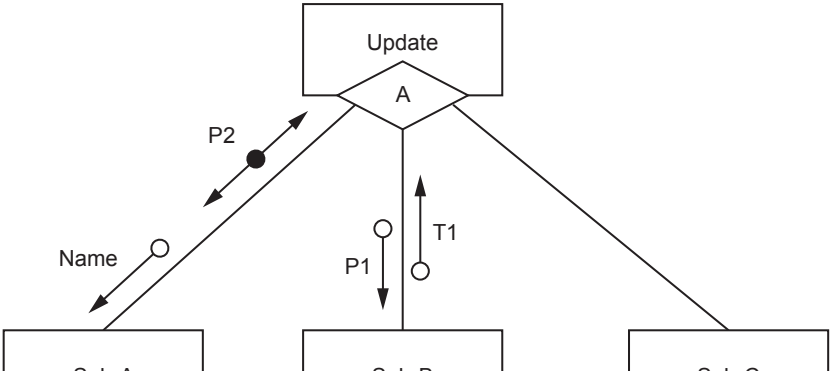
[3]

**(iii)** Identify **two** operations that would be required to process data when the new module receives a text message back from a member.

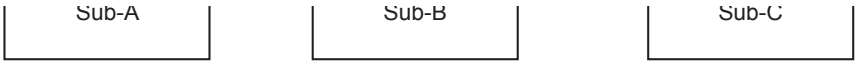
# Question 7

[2]

**(b)** The structure chart illustrates part of the membership program:



# Question 7




Data item notes:

- Name contains the name of a club member
- P1 and T1 are of type real.

**(i)** Explain the meaning of the diamond symbol (labelled with the letter A) in the chart.

# Question 7

**(ii)** Write the pseudocode module headers for Sub-A and Sub-B.

Sub-A

# Question 7

Sub-B

# Question 7

[4]

# Solution 7

**(a)(i)**

## Worked steps

- Decomposition makes the solution **easier to design, implement and solve**.

That is the mark, and the reason behind it is worth carrying: a problem broken into named sub-problems can be worked on one piece at a time, by more than one person, and each piece can be tested on its own. Breaking a large problem into smaller ones is the whole of computational thinking's first step.

## Mark scheme

- To make the solution easier to design / implement / solve
- 1
- 9618/21
- May/June 2024

# Solution 7

## (a)(ii) Mark scheme

- One mark per item and justification
- Item: mobile phone number
- Justification: to send the text message
- Item: name
- Justification: to personalise the text message
- Item: exercise interest
- Justification: to determine whether this member would be interested
- 3

# Solution 7

## (a)(iii) Worked steps

Two marks, one per module, and each must be a **distinct job** the system needs. Examples that fit this fitness-club system:

- **Add a member** to the list of those interested in a new class.
- **Remove a member** from future SMS messages.
- **Read or process a message** that has arrived.
- **Identify who a message is from.**

# Solution 7

Each is one identifiable task, expressible as a verb phrase, and each could be written and tested by somebody who knows only its name and parameters. That is the test to apply before writing a module down.
Modules that merely restate the whole system — "handle messages" — are not a decomposition, because their use cannot be stated in one line. **Mark scheme**

---

- Examples include:
- ▪
- Add a member to a list of those interested in the new class
- ▪
- Remove the member from future SMS messages

# Solution 7

- ▪
- Read/process Message
- ▪
- Identify who from
- One mark for each.
- Max 2 marks
- 2

# Solution 7

## (b)(i) Worked steps

Two marks: what the notation **means**, and **naming the modules**.

- The symbol on the chart means that `Update` calls **one of** `Sub-A`, `Sub-B` or `Sub-C` — not all three.
- That is **selection** — a decision, an IF or CASE inside `Update`.

Both halves are marked separately, so say the word *selection* (or decision) **and** name all four modules. Describing the shape without naming the modules earns one mark of two.

The three notations on a structure chart are worth having ready: a **diamond** for selection, a **curved arrow** looping round the lines for iteration, and plain lines for a sequence executed left to right. **Mark scheme**

# Solution 7

- Means that Update calls (one of) either Sub-A, Sub-B or Sub-C
- One mark for each point:
- ▪
- reference to selection / decision / if
- ▪
- naming all four modules correctly
- 2

# Solution 7

## (b)(ii) Worked steps

Read each module's arrows off the structure chart and turn them into a header. One mark per underlined part, so every element is marked separately.

```
PROCEDURE Sub-A(Name : STRING, BYREF P2 : BOOLEAN)

FUNCTION  Sub-B(P1 : REAL) RETURNS REAL
```

The three decisions behind them:

# Solution 7

- `Sub-A` **is a procedure** — nothing comes back as a *result* — but `P2` has an arrow going back up on its parameter line, so `P2` is passed **BYREF**. That is how a procedure returns something.
- `Sub-B` **is a function**, because a value returns to the caller as its result rather than through a parameter: `RETURNS REAL`.
- **Every parameter carries its type**, taken from the chart's own labels, and a function carries its return type too.

The pair is deliberately contrasted: both modules send information back, and the chart distinguishes them by whether the upward arrow sits on a **parameter line** or on the **module itself**.

# Solution 7

Because the marking is per underlined part, partial credit is real — write the header out in full even if one element is uncertain, rather than leaving it blank.

**Mark scheme**

- PROCEDURE Sub-A (Name : STRING, BYREF P2 : BOOLEAN)
- FUNCTION Sub-B (P1 : REAL) RETURNS REAL
- One mark per underlined part in each case
- 4
- 9618/21
- May/June 2024

# Question 7

9618/22 J24 ▪ Q7 ▪ 9 marks

**7** A fitness club has a computerised membership system.

The system stores information for each club member: name, home address, email address, mobile phone number, date of birth and exercise preferences.

Many classes are full, and the club creates a waiting list for each class. The club adds details of members who want to join a class that is full to the waiting list for that class.

When the system identifies that a space is available in one of the classes, a new module will send a text message to each member who is on the waiting list.

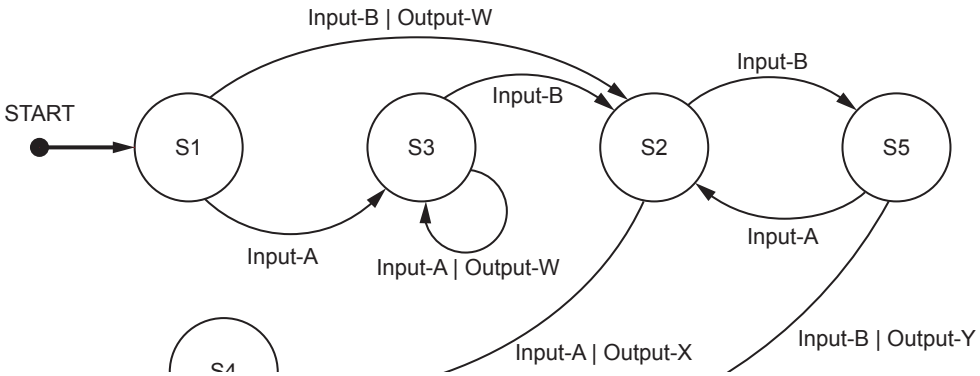
**(a)** Decomposition will be used to break the new module into sub-modules (sub-problems).

Identify **three** sub-modules that could be used in the design **and** describe their use.

# Question 7

[3]

**(b)** A different part of the program is represented by the following state-transition diagram.



# Question 7


**(i)** Complete the table to show the inputs, outputs and next states.

Assume that the current state for each row is given by the ‘Next state’ on the previous row. For example, the first Input-A is made when in state S1.

If there is no output for a given transition, then the output cell should contain ‘none’.

The first two rows have been completed.

# Question 7

| Input | Output | Next state |
|---|---|---|
| | | S1 |
| Input-A | none | S3 |
| | Output-W | |
| | none | |
| Input-B | | |
| Input-A | | |
| | | S4 |

[5]

# Question 7

**(ii)** Identify the input sequence that will cause the minimum number of state changes in the transition from S1 to S4.

# Solution 7

## (a) Mark scheme

- Examples include:
- Module: IdentifyMember()
- Use: Identifies a club member who has expressed an interest in a given class
- Module: GetMemberPhoneNumber()
- Use: Gets the mobile phone number of a member
- Module: CreateMessage()
- Use: Generates a text message to a member
- Module: SendMessage()

# Solution 7

- Use: Sends a text message to a member in the waiting list
- One mark for name and use
- Note: max 3 marks
- 3

# Solution 7

## (b)(i) Mark scheme

- Input
- Output
- Next state
- S1
- Input-A
- none
- S3
- Input-A

# Solution 7

- Output-W
- S3
- Input-B
- none
- S2
- Input-B
- none
- S5
- Input-A
- none

# Solution 7

- S2
- Input-A
- Output-X
- S4
- One mark per row 3 to 7/5

# Solution 7

**(b)(ii)**

**Mark scheme**

- Input-B, Input-A
- 1