How to use this clinic

This booklet uses authentic QCAA external-assessment questions. Complete each question on the captured exam page first. Then turn the page and compare your work with a worked response.

> **Do not copy first.** Cover the worked response, attempt the question under paper-exam conditions, then use a different-coloured pen to improve your answer.

What to look for in every worked response

- the inputs, outputs and data structures the question requires
- control structures that match the task
- clear variable names and consistent indentation
- a desk check that records changing values, not only a final answer
- comments only where they help link code to a criterion.

> **Authentic does not mean one rigid answer.** Pseudocode conventions vary. A response can be different from the model and still be correct when it is unambiguous, logical and meets all requirements.

Included question types

Skill	Questions in this booklet
Caesar cipher construction and tracing	2021
One-time-pad construction, desk checking and supplied functions	2022
Logic-error correction and justification	2023
Traversing JSON records	2024
Nested-loop desk checking and criteria-driven code	2025

Question captures are reproduced from QCAA external assessments (2021–25), © State of Queensland (QCAA), CC BY 4.0. Worked explanations are newly written for this class.

Before Question 1: Caesar ciphers from the beginning

A **cipher** is a rule for changing readable text into text that is harder to read. A Caesar cipher is one of the simplest possible ciphers. It does not hide letters in a complicated way; it moves every letter the same number of places around the alphabet.

Suppose the key is `3`. Start at `A`, count three places forward, and you reach `D`. So a message letter `A` becomes `D`. In the same cipher, `B` becomes `E`. The number of places moved is called the **key shift**.

Plain letter	Move forward 3 places	Encrypted letter
A	B → C → D	D
X	Y → Z → A	A

The second example matters. The alphabet is a circle, not a straight line. After `Z`, start again at `A`. To **decrypt**, reverse the movement. If encryption moved three places forward, decryption moves three places backwards.

> **What a desk check means here:** do not guess one letter at a time. Test a possible key shift on the whole encrypted word, write the full result, then try the next key shift. A real word is evidence that the key may be correct.

Question 1: Caesar cipher

c) You discover that the app uses a Caesar cipher to encrypt data, allowing you to intercept a password in transit. Perform a desk check for five iterations to decrypt the password '`KYIWW`'. *[2 marks]*

Do not write outside this box.

2021 external assessment, Question 12c, page 6 of the question-response book.

Question 1: Caesar cipher continued

d) A friend is interested in learning more about encryption. Use pseudocode to symbolise a simple Caesar cipher as a demonstration. *[8 marks]*

Do not write outside this box.

2021 external assessment, Question 12d, page 7 of the question-response book.

Worked response: Caesar cipher

Imagine the alphabet as a circular dial. Moving one step backwards from `B` reaches `A`; moving one more step backwards wraps around to `Z`. A Caesar cipher uses the same movement for every letter. The problem does not tell us the movement, so we test possible shifts until the result is a sensible password.

Desk check first

The key shift is not supplied. Test one possible backward shift on the *whole* password per iteration. The fifth row reveals a meaningful plaintext word.

Key shift tested	Result after shifting every letter backwards
0	KYIWW
1	JXHVV
2	IWGUU
3	HVFTT
4	GUESS

A clear algorithmic model

This algorithm is the same idea written for a computer. It takes in the message and the number of places to move. Then it deals with one letter at a time: find where that letter sits in the alphabet, move it by the key, and add the new letter to the growing encrypted message.

```
BEGIN
INPUT plainText
INPUT keyShift
SET alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
SET encryptedText = ""

FOR EACH character IN plainText DO
    SET position = index of character in alphabet
    SET newPosition = (position + keyShift) MOD 26
    SET encryptedText = encryptedText + alphabet[newPosition]
ENDFOR

RETURN encryptedText
END
```

Why this earns marks: the question is really asking, “Can your algorithm repeat one clear rule for every letter?” This response shows the starting information, the alphabet it needs, the repeated action and the final result. `MOD 26` is the wrap-around rule: if a position goes past 25, it returns to the beginning of the 26-letter alphabet.

Official marking guide: Caesar cipher

The response:

- performs a desk check that
  - decrypts the password using iterations **[1 mark]**
  - contains no errors **[1 mark]**

- symbolises
  - INPUT for password string **[1 mark]**
  - INPUT for key or shift integer **[1 mark]**
  - declarative statement for alphabet **[1 mark]**
  - a FOR loop to traverse password one character at a time **[1 mark]**
  - a function to encrypt password one character at a time, using the key **[1 mark]**
  - RETURN for encrypted password **[1 mark]**
  - an algorithm with no logic errors **[1 mark]**
- effectively uses pseudocode conventions **[1 mark]**

QCAA Digital Solutions marking guide and response, 2021, Questions 12c?d. © State of Queensland (QCAA), CC BY 4.0.

Stimulus for Question 2

Stimulus 2

These are sample procedures and programming structures for a one-time pad encryption algorithm.

Pseudocode sub-procedures

```
indexOf( <character>, <array> )
```

Purpose: To return the index of the located character in the array.

```
lengthOf( <string of characters> )
```

Purpose: To return the number of characters in a string.

```
checkLowerCase( <string of characters> )
```

Purpose: To return 1 if only lower case characters exist or return 0 otherwise.

```
SET
lowerCaseAlphabet ['a','b','c','d','e','f','g','h','i','j','k','l','m',
'n','o','p','q','r','s','t','u','v','w','x','y','z']
```

Example of usage: `lowerCaseAlphabet[4] is equal to 'e'`

2022 external assessment stimulus book, Stimulus 2.

Before Question 2: One-time pad from the beginning

A **one-time pad** encryption algorithm uses two pieces of text of the same length: the message to protect and a secret **key**. The key is another sequence of letters. The algorithm combines the first message letter with the first key letter, then the second with the second, and so on.

Computers cannot add letters directly, so the algorithm first gives every lower-case letter a number. This stimulus uses an **array** called `lowerCaseAlphabet`. An array is an ordered list: each item has a numbered position, called an **index**. This array starts counting at 0, not 1.

Letter	Index	Letter	Index	Letter	Index
a	0	b	1	c	2
q	16	r	17	z	25

The four supplied helpers

`indexOf(letter, alphabet)` asks, "Which numbered position contains this letter?" For example, `indexOf("q", lowerCaseAlphabet)` returns `16`. `lengthOf(word)` counts letters. `checkLowerCase(word)` checks whether every character is lower-case. These helpers save you from having to write those jobs yourself.

What `MOD 26` actually does

`MOD` means "the remainder after division". `MOD 26` means divide by 26 and keep only the remainder. There are 26 lower-case letters, indexed 0 to 25, so this is the rule that brings a number back into the valid alphabet positions.

Calculation	Why	Resulting alphabet index
`7 MOD 26`	7 is already inside 0–25.	7
`26 MOD 26`	26 divides exactly by 26, leaving remainder 0.	0
`33 MOD 26`	26 fits once into 33, leaving remainder 7.	7
`51 MOD 26`	26 fits once into 51, leaving remainder 25.	25

> **Why message and key length must match:** if the message has six letters but the key has only five, the sixth message letter has no key letter to combine with. The algorithm must stop and show an error instead of accessing a missing array position.

Question 2: One-time pad and desk check

QUESTION 13 (12 marks)

Refer to Stimulus 2 in the stimulus book.

a) Use pseudocode to symbolise a one-time pad encryption algorithm with any lower case letters. Users must input the plain text and key. *[7 marks]*

Do not write outside this box.

2022 external assessment, Question 13a, page 4 of the question-response book.

Question 2: Desk-check evidence

b) Evaluate the encryption algorithm developed in Question 13a) by performing a desk check. Use the plain text and key provided. Specify all inputs and outputs in table form.

Plain text: `Queens`

Key: `ryjwah`

Example output: `hsnanz` *[5 marks]*

Do not write outside this box.

2022 external assessment, Question 13b, page 5 of the question-response book.

Worked response: One-time pad

Think of this as two alphabet rulers lined up under each other: one ruler is the message and one is the secret key. At each position, turn both letters into numbers, add the numbers, then use the new number to choose an encrypted letter. The key must be the same length as the message because every message letter needs its own partner.

Before doing any encryption, the algorithm checks that both inputs are lower-case and equal length. This prevents it from trying to match a sixth message letter with a key that only has five letters.

```
BEGIN
INPUT userText
INPUT otpKey
SET alphabet = "abcdefghijklmnopqrstuvwxyz"
SET cipherOutput = ""

IF lengthOf(userText) = lengthOf(otpKey) AND
   checkLowerCase(userText) AND checkLowerCase(otpKey) THEN
    FOR i = 0 TO lengthOf(userText) - 1 DO
        SET textIndex = indexOf(userText[i], alphabet)
        SET keyIndex = indexOf(otpKey[i], alphabet)
        SET encryptedIndex = (textIndex + keyIndex) MOD 26
        SET cipherOutput = cipherOutput + alphabet[encryptedIndex]
    ENDFOR
    OUTPUT cipherOutput
ELSE
    OUTPUT "Text and key must be equal-length lower-case strings"
ENDIF
END
```

Desk check: queens with `ryjwah`

The alphabet starts counting at 0, so `q` is 16 and `r` is 17. Their total is 33. There is no alphabet position 33, so `33 MOD 26` leaves 7: position 7 is `h`. Do the same small calculation for every pair of letters.

Text letter/index	Key letter/index	(text + key) MOD 26	Output
q / 16	r / 17	7	h
u / 20	y / 24	18	s
e / 4	j / 9	13	n
e / 4	w / 22	0	a
n / 13	a / 0	13	n
s / 18	h / 7	25	z

> **Desk-check habit:** A desk check is like showing your working in maths. The final word `hsnanz` is not enough because it does not show where a wrong index, wrong key letter or missing wrap-around rule would have happened.

Official marking guide: One-time pad

Use this after attempting Question 2. The published criteria cover both the algorithm and its desk check.

The response:
• symbolises, without logic errors, an algorithmic statement for – user input of text **[1 mark]** – user input of key **[1 mark]** – 'indexOf' sub-procedure **[1 mark]** – 'lengthOf 'sub-procedure **[1 mark]** – 'checkLowerCase' sub-procedure **[1 mark]** – 'lowerCaseAlphabet' **[1 mark]** • effectively uses pseudocode conventions **[1 mark]**
• demonstrates accurate use of – one programming feature that specifies all inputs and outputs in table form **[1 mark]** – a second programming feature that specifies all inputs and outputs in table form **[1 mark]** – a third programming feature that specifies all inputs and outputs in table form **[1 mark]** – a fourth programming feature that specifies all inputs and outputs in table form **[1 mark]** – a fifth programming feature that specifies all inputs and outputs in table form **[1 mark]**

QCAA Digital Solutions marking guide and response, 2022, Question 13a?b. © State of Queensland (QCAA), CC BY 4.0.

Stimulus for Question 3

Stimulus 3

This code library of functions enables secure access to student results in a database.

Function	`hashPassword`
Description	Apply a hash function to a password
Input	**Return**
• Plain text password	• Hashed value
Function call	`hashPassword(plaintext)`

Function	`checkLogin`
Description	Look up a student ID and hashed password in database
Input	**Return**
• Student ID • Hashed value	• True/False
Function call	`checkLogin(studentID, password)`

Function	`getResults`
Description	Retrieve all results for a student ID
Input	**Return**
• Student ID	• Array of line of encrypted data
Function call	`getResults(studentID)`

Function	`decryptData`
Description	Apply a decryption algorithm to the data using the supplied key
Input	**Return**
• Key • Line of encrypted data	• Line of decrypted data
Function call	`decryptData(key, resultsLine)`

2022 external assessment stimulus book, Stimulus 3.

Before Question 3: Functions are named jobs

A **function** is a named job that has already been written. Think of a vending machine: you provide the required input, press the button, and receive the defined output. You do not need to know how the machine works inside, but you must use the correct button and supply what it expects.

Function call	Input you supply	Returned result or action
`hashPassword(password)`	Readable password	A hashed version of that password
`checkLogin(studentID, hashedPassword)`	ID and hashed password	`TRUE` or `FALSE`
`getResults(studentID)`	Student ID	A list of encrypted result lines
`decryptData(key, resultLine)`	Key and one encrypted line	One readable result line

A value returned by a function is like the answer to a question. Store it in a variable if it is needed later. For example, `SET results = getResults(studentID)` means “ask for the results, then keep the returned list in a labelled box named `results`”.

Question 3: Complete supplied pseudocode

QUESTION 14 (6 marks)

Refer to Stimulus 3 in the stimulus book.

A secondary school stores encrypted student results in a database. To access their results, students must log in using their student ID and password. Login details are checked against the database to ensure only authorised students can gain access.

Passwords are encrypted for added security. Once logged in, students must enter a public decryption key to have their results displayed.

The database contains overall results for all subjects and students. Only results linked to the relevant student ID are displayed; students cannot view other students' results.

The algorithm for this system is incomplete. Complete the algorithm using the supplied code library.

```
BEGIN
        GET studentID from input form
        GET password from input form
        GET key from input form
```

Do not write outside this box.

2022 external assessment, Question 14, page 6 of the question-response book.

Worked response: Read the function contract

Provided functions are like tools borrowed from a toolbox. You do not need to build the tool, but you must use the right tool with the right input. Read the function description before writing the line: `hashPassword` changes the password, `checkLogin` answers yes or no, `getResults` fetches encrypted result lines, and `decryptData` turns one encrypted line into readable data.

```
BEGIN
GET studentID from input form
GET password from input form
GET key from input form

IF checkLogin(studentID, hashPassword(password)) THEN
    SET results = getResults(studentID)
    FOR i = 0 TO size of results - 1 DO
        DISPLAY decryptData(key, results[i])
    ENDFOR
ELSE
    DISPLAY "Access denied"
ENDIF
END
```

Required action	Evidence in the response
Protect password	`hashPassword(password)` is passed into the login check.
Check identity	`checkLogin` receives the student ID and hashed password.
Handle a failed login	The `ELSE` branch displays an access-denied message.
Limit results to that student	`getResults(studentID)` retrieves only their records.
Decrypt each record	The loop calls `decryptData` once per result.

> **Common error:** Do not give a function superpowers. Here `getResults` is a courier: it collects the student's result lines, but it does not check the password and it does not unlock the lines. The algorithm must do those jobs with the other supplied functions.

Official marking guide: Supplied functions

Use this after attempting Question 3. Check that your algorithm performs every required job.

The response:

- symbolises, without logic errors, an algorithmic statement for
  - password encryption using 'hashPassword' **[1 mark]**
  - checking student ID and hashed password using 'checkLogin' **[1 mark]**
  - handling login errors **[1 mark]**
  - retrieving data for specific student ID using 'getResults' **[1 mark]**
  - outputting data for specific student ID using 'decryptData' **[1 mark]**
- effectively uses pseudocode conventions **[1 mark]**

QCAA Digital Solutions marking guide and response, 2022, Question 14. © State of Queensland (QCAA), CC BY 4.0.

Question 4: Find four logic errors

QUESTION 13 (9 marks)

A one-time pad algorithm encrypts an 8-character word and outputs the result. The algorithm uses 0 as the first index in an array. There are four errors on lines 9 to 21 inclusive.

```
1  BEGIN
2     INPUT word
3     word = word converted to lowercase
4     plainText = word converted to an array
5     //The GenerateOTPKey function will return an array of 8 random
      numbers
6     key = GenerateOTPKey()
7     cipherText = array of 8 blank characters
8     alphabet = lowercase alphabet converted to an array
9     FOR i = 0 to 9
10      alphabetLocation = -1
11      FOR j = 0 to 25
12        IF plainText[j] = alphabet[i]
13          alphabetLocation = j
14        ENDIF
15      ENDFOR
16      //Check if not found
17      IF alphabetLocation ≥ 0 THEN
18        cipherText[i] = alphabet(alphabetLocation + key[i])mod 26
19      ENDIF
20    ENDFOR
21    OUTPUT cipherText[i]
22 END
```

Do not write outside this box.

2023 external assessment, Question 13, page 6 of the question-response book.

Question 4: Correction task

Use pseudocode to correct the four errors on lines 9 to 21. Justify your response.

Do not write outside this box.

2023 external assessment, Question 13, page 7 of the question-response book.

Worked response: Locate the first wrong value

Debugging means acting like a careful proof-reader. Start at the first instruction that could send the algorithm in the wrong direction. Ask: “What does this variable represent? What values is it allowed to use? What would this line do on the first and last loop?” The table identifies the four places where the answer is “not what the task needs”.

Line	Correction	Why
9	`FOR i = 0 TO 7`	An eight-character word has indices 0–7. Iterating to 9 accesses positions that do not exist.
12	`IF plainText[i] = alphabet[j]`	`i` selects the current character of the word; `j` searches the alphabet.
18	`cipherText[i] = alphabet[(alphabetLocation + key[i]) MOD 26]`	The brackets make the wrapped numeric index before it is used to select a letter from `alphabet`.
21	`OUTPUT cipherText`	The whole completed array must be output, not only the last indexed element.

Justification pattern: do not just write “the index is wrong”. Say what the index is meant to point to, what the valid positions are, and what the original line would accidentally do. For example: an eight-letter array has positions 0 to 7, so a loop ending at 9 tries to reach letters that do not exist.

Official marking guide: Four logic errors

Use this after attempting Question 4. Marks are awarded for each correction and each justification.

The response:

- logically symbolises a solution for the error on
  - line 9 **[1 mark]**
  - line 12 **[1 mark]**
  - line 18 **[1 mark]**
  - line 21 **[1 mark]**
- justifies symbolised solution for
  - line 9 **[1 mark]**
  - line 12 **[1 mark]**
  - line 18 **[1 mark]**
  - line 21 **[1 mark]**
- does not introduce new logic errors **[1 mark]**

 © State of Queensland (QCAA), CC BY 4.0.

Before Question 5: Reading JSON without guessing

JSON is a way of storing structured data using labels and values. Curly brackets `{ }` mean one object, like one labelled information card. Square brackets `[ ]` mean a list of items. A colon joins a label to its value.

```
{
  "gamerTag": "Madskills",
  "gamesPlayed": 10,
  "gamesWon": 4
}
```

This is one player object. The label `"gamerTag"` points to text. The other labels point to numbers. To access a value, use the exact label: `player["gamesWon"]`. The quotation marks matter because the label itself is text.

In the question, the outer JSON object contains a list labelled `"players"`. The algorithm must first get that list, then use a `FOR EACH` loop. `FOR EACH player IN players` means “take the first player, do the work; take the next player, do the same work; continue until there are no players left”.

> **Percentage won:** first calculate the part: wins ÷ games played. For 4 wins from 10 games, `4 / 10 = 0.4`. Then turn the decimal into a percentage: `0.4 × 100 = 40`, so the player won 40% of games.

Question 5: Traverse JSON player data

QUESTION 12 (8 marks)

An esports club records player details and results for playing a popular online game. The data is stored in JSON format. The club wants to display the gamer tag of each player, their age and the percentage of games won. A sample of the JSON data is shown.

```
{
'players': [
        {
            'name': 'Brandon Rioli',
            'gamerTag': 'Madskills',
            'dateOfBirth': '22/12/2007',
            'gamesPlayed': 10,
            'gamesWon': 4
        },
        {
            'name': 'Chloe Pezer',
            'gamerTag': 'PezerGirl',
            'dateOfBirth': '03/04/2007',
            'gamesPlayed': 58,
            'gamesWon': 55
        }
    ]
}
```

Use pseudocode to symbolise the algorithmic statements needed to display the required data.

Do not write outside this box.

2024 external assessment, Question 12, page 6 of the question-response book.

Worked response: JSON records

JSON is organised like labelled storage boxes. The outer box is called `clubData`. Inside it, the label `"players"` leads to a list. Each item in that list is one player box, with smaller labelled values such as `"gamerTag"` and `"gamesWon"`.

The loop means “repeat these instructions once for every player box”. Inside the loop, the algorithm calculates values for the *current* player, then displays that player’s gamer tag, age and win percentage before moving to the next one.

```
READ jsonData INTO clubData
SET players = clubData["players"]

FOR EACH player IN players DO
    SET age = age calculated from player["dateOfBirth"]
    SET percentageWon = (player["gamesWon"] / player["gamesPlayed"]) * 100
    OUTPUT player["gamerTag"], age, percentageWon
ENDFOR
```

Mark-by-mark self-check

What the question requires	Where it appears
Retrieve JSON data	`READ jsonData` and access `["players"]`.
Process every player	`FOR EACH player IN players`.
Calculate age and percentage won	Two named calculations inside the loop.
Display three required values	`OUTPUT` includes gamer tag, age and percentage.

Accuracy check: Percentage won asks, “Out of every 100 games, how many did this player win?” First calculate the fraction: wins divided by games played. Then multiply by 100. It is not the number of wins divided by age or a count of players.

Official marking guide: JSON player data

Use this after attempting Question 5. Use this list to check every required calculation and output.

Q	The response:
12	• symbolises, without logic errors, algorithmic statements for – retrieving JSON data **[1 mark]** – looping through data **[1 mark]** – calculating age **[1 mark]** – calculating percentage won **[1 mark]** – displaying gamer tag **[1 mark]** – displaying age **[1 mark]** – displaying percentage won **[1 mark]** • effectively uses pseudocode conventions **[1 mark]**

QCAA Digital Solutions marking guide and response, 2024, Question 12. © State of Queensland (QCAA), CC BY 4.0.

Before Question 6: What a desk check records

A **desk check** is a slow, written simulation of an algorithm. You become the computer and follow each instruction in order. Do not jump straight to the final answer. Write down the values that change so you can prove how the result was produced.

This question has a **nested loop**: a loop inside another loop. The outer loop chooses one movie. The inner loop then visits every review for that one movie. When the inner loop finishes, the outer loop moves to the next movie and the inner loop starts again for that movie.

`totalRating` is a running total. It starts at 0, then grows as ratings are added. `numberOfReviews` is a running count. It starts at 0, then increases by 1 for every review. An average is only calculated after the algorithm has processed all reviews for that movie:

```
averageRating = totalRating / numberOfReviews
```

> **The reset is important:** these two variables belong to one movie only. Reset them to 0 at the start of every outer-loop iteration, otherwise the second movie accidentally uses the first movie's ratings.

Stimulus for Question 6

Stimulus 1

The JSON sample shows review data for two movies.

```
{
  "movies": [{
      "title": "The Legends List",
      "reviews": [{
          "review_id": 1,
          "reviewer_name": "Amanda",
          "rating": 5,
          "review_text": "Amazing movie!"
        },
        {
          "review_id": 2,
          "reviewer_name": "Julian",
          "rating": 4,
          "review_text": "Great concept"
        },
        {
          "review_id": 3,
          "reviewer_name": "Saoirse",
          "rating": 3,
          "review_text": "Average"
        }
      ]
    },
    {
      "title": "The Sequel",
      "reviews": [{
          "review_id": 1,
          "reviewer_name": "Gary",
          "rating": 2,
          "review_text": "Nope"
        },
        {
          "review_id": 2,
          "reviewer_name": "Jaspreet",
          "rating": 1,
          "review_text": "Why?"
        },
        {
          "review_id": 3,
          "reviewer_name": "Kim",
          "rating": 2,
          "review_text": "Two hours of my life I'd like back, please!"
        }
      ]
    }
  ]
}
```


2025 external assessment stimulus book, Stimulus 1.

Question 6: Desk check nested review data

QUESTION 14 (8 marks)

Refer to Stimulus 1 in the stimulus book.

The algorithm shown calculates the average ratings from movie review data.

```
0  BEGIN
1   READ jsonData into movieData
2   INITIALISE a dictionary object to store averageRatings
3
4   FOR each movie in movieData DO
5     SET totalRating = 0
6     SET numberOfReviews = 0
7
8     GET movieTitle = movie["title"]
9     GET reviews = movie["reviews"]
10
11    FOR each review in reviews DO
12      totalRating = totalRating + review["rating"]
13      numberOfReviews = numberOfReviews + 1
14    ENDFOR
15    IF numberOfReviews > 0 THEN
16      averageRating = totalRating / numberOfReviews
17    ELSE
18      averageRating = "No ratings"
19    ENDIF
20    STORE averageRating in the dictionary object with the movie title
      as the key
21    OUTPUT "Average Rating for ", movieTitle, " is: ", averageRating
22  ENDFOR
23  RETURN averageRatings
24 END
```

Do not write outside this box.

2025 external assessment, Question 14, page 8 of the question-response book.

Question 6: Desk-check table

Desk check all variables used to calculate the average rating for each movie.

Do not write outside this box.

2025 external assessment, Question 14, page 9 of the question-response book.

Worked response: Nested-loop desk check

Imagine sorting ratings into two separate buckets, one bucket per movie. When the algorithm starts a movie, it puts an empty total and an empty count beside that bucket. Each review adds its rating to the total and adds one to the count. Only after all reviews for that movie have been processed can it divide total by count to find the average.

When the outer loop moves to the next movie, the total and count must return to zero. If they do not, ratings from the first movie leak into the second movie's average.

Movie	review["rating"]	totalRating	numberOfReviews	averageRating
The Legends List — start	—	0	0	—
The Legends List — review 1	5	5	1	—
The Legends List — review 2	4	9	2	—
The Legends List — review 3	3	12	3	4
The Sequel — start	—	0	0	—
The Sequel — review 1	2	2	1	—
The Sequel — review 2	1	3	2	—
The Sequel — review 3	2	5	3	1.67

What this table demonstrates: the actual stimulus ratings, running total, running count and resulting average for both movies. A final average alone is not enough desk-check evidence because it hides whether you reset variables in the right place.

Official marking guide: Movie-rating desk check

Use this after attempting Question 6. QCAA requires evidence for both movies, not only final averages.

	The response:
	• desk checks, for one movie, the values of – totalRating **[1 mark]** – numberOfReviews **[1 mark]** – review["rating"] **[1 mark]** • desk checks, for a second movie, the values of – totalRating **[1 mark]** – numberOfReviews **[1 mark]** – review["rating"] **[1 mark]** • correctly calculates averageRating for one movie **[1 mark]** • correctly calculates averageRating for a second movie **[1 mark]**

QCAA Digital Solutions marking guide and response, 2025, Question 14. © State of Queensland (QCAA), CC BY 4.0.

Before Question 7: Turn criteria into visible code

A **criterion** is a requirement the final solution must meet. Treat each criterion as a checklist item, not as background information. Before writing code, turn every requirement into a question: “What data tells me this has happened?” and “What action will my algorithm take?”

Criterion in this question	Question to ask yourself	Likely code action
Detects hazard locations	What sensor evidence shows danger, and where is it?	Use an `IF` condition and store the hazard location.
Prioritises emergency vehicles	Which roads and crossings are on the emergency route?	Loop through the route and change signals.
Keeps pedestrians safe	Which crossing must stop showing “walk”?	Check the signal and set it to a danger/stop state.

A **comment** begins with #. It is a note for a human reader, not an instruction the computer performs. Here, comments help the marker quickly see which code block meets which criterion.

> **Do not invent a function:** the code library is the approved set of buttons. Use the names and inputs given in the stimulus. Your algorithm supplies the decision-making around those buttons.

Stimulus for Question 7: Code library

Stimulus 2

Real-time data are collected to monitor the status of adaptive traffic signals, pedestrian signals and traffic conditions. The code library of functions enables the processing of real-time data for the smart traffic management system.

Function call:

`getAdaptiveSignal(signal_name)`

Purpose:

This function gets the adaptive traffic signal for a given road or crossing.

It returns text describing the signal status, e.g. `"red"`, `"yellow"` or `"green"`

Function call:

`setAdaptiveSignal(signal_name, signal)`

Purpose:

This function sets the adaptive traffic signal for a given road or crossing.

Function call:

`evaluateCongestion(signal_name)`

Purpose:

This function evaluates the current congestion level for a given road or crossing based on the real-time traffic conditions.

It returns text describing the congestion, e.g. `"none"`, `"low"`, `"medium"` or `"high"`

Function call:

`getVehicleCount(lane)`

Purpose:

This function retrieves and returns the number of vehicles currently detected on a specific lane. It uses real-time sensor data to estimate traffic volume.

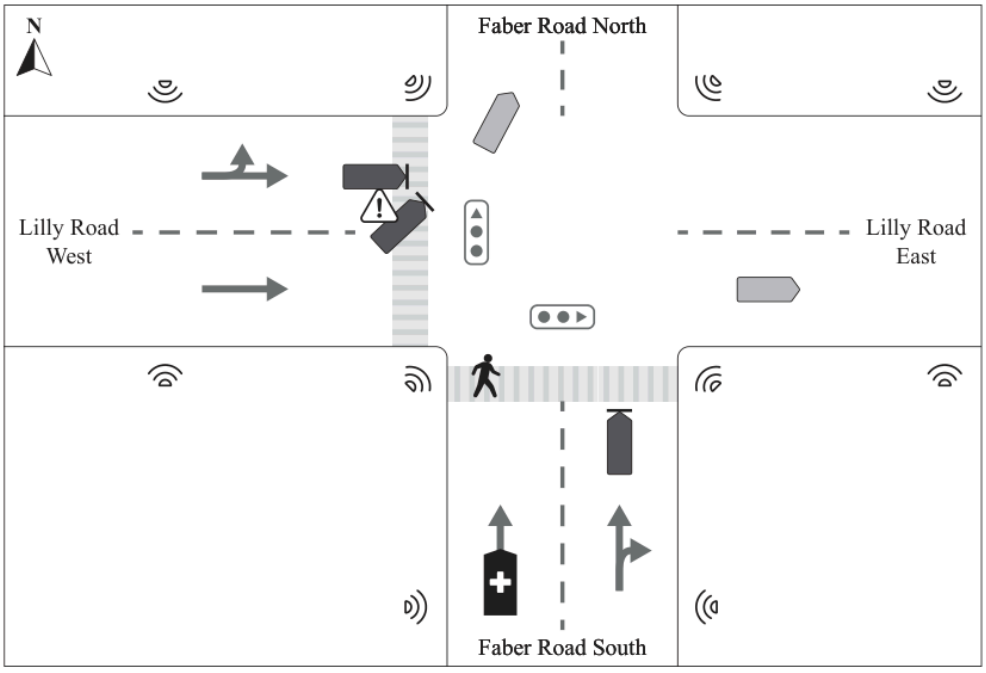
It returns the number of vehicles currently detected.

2025 external assessment stimulus book, Stimulus 2, page 2.

Stimulus for Question 7: Traffic scenario

The diagram shows a traffic scenario at the intersection of Lilly Road and Faber Road. Both are one-way roads with two lanes of traffic. All traffic signals for Faber Road are red, and a pedestrian is crossing the road. An accident has occurred on Lilly Road West, and an emergency vehicle is approaching from Faber Road South.





2025 external assessment stimulus book, Stimulus 2, page 3.

Question 7: Build from criteria and functions

You have been asked to develop a programmed component that responds to the hazardous situations at the Lilly and Faber Roads intersection.

Criteria:

- detects hazard locations
- prioritises passage for emergency vehicles
- keeps pedestrians safe.

b) Use pseudocode to complete the algorithm for the programmed component, addressing all the criteria. Incorporate at least three code library functions. Include comments to identify where each criterion is addressed. *[8 marks]*

```
BEGIN

#Define the intersection roads and crossings
SET roads = ["LillyE", "LillyW","FaberS","FaberN"]
SET crossings = ["LillyE_cross", "LillyW_cross","FaberS_cross",
"FaberN_cross"]
```

Do not write outside this box.

2025 external assessment, Question 16b, page 14 of the question-response book.

Worked response: Criteria drive the code

Start this kind of question as if you are giving instructions to a traffic controller who cannot guess your intention. The criteria are the controller's three jobs: find danger, protect pedestrians, and help the emergency vehicle move through. Each job needs a visible block of pseudocode so a marker can point to the evidence.

The supplied functions are the controller's buttons: check the signal, change the signal and check congestion. The algorithm decides *when* to press each button; it does not invent new buttons with unknown behaviour.

```
BEGIN
SET roads = ["LillyE", "LillyW", "FaberS", "FaberN"]
SET crossings = ["LillyE_cross", "LillyW_cross", "FaberS_cross", "FaberN_cross"]

# Criterion: detect hazard locations using sensor data
IF sensor data shows a stopped collision OR wrong-way vehicle THEN
    SET hazardLocation = reported sensor location
    # Criterion: keep pedestrians safe
    FOR EACH crossing near hazardLocation DO
        CALL setAdaptiveSignal(crossing, "danger")
    ENDFOR
ENDIF

# Manage congestion when there is no current hazard or emergency vehicle
IF hazardLocation IS NONE AND emergency vehicle is not detected THEN
    FOR EACH road IN roads DO
        IF evaluateCongestion(road) = "high" THEN
            CALL setAdaptiveSignal(road, "green")
        ENDIF
    ENDFOR
ENDIF

# Criterion: prioritise passage for the known emergency vehicle
FOR EACH road IN emergency vehicle predicted path DO
    CALL setAdaptiveSignal(road, "green")
ENDFOR

FOR EACH crossing IN emergency vehicle predicted path DO
    SET pedestrianSignal = getAdaptiveSignal(crossing)
    IF pedestrianSignal = "walking" THEN
        CALL setAdaptiveSignal(crossing, "danger")
    ENDIF
ENDFOR
END
```

Coverage check: Pretend you are the marker with a highlighter. Can you highlight one block for danger, one for pedestrians and one for the emergency vehicle? Can you circle three different supplied functions? If you cannot point to the evidence, the answer may sound sensible but it has not yet proved that it meets the criteria.

Official marking guide: Criteria-driven programmed component

Use this after attempting Question 7. Check the criteria, library functions and comments one by one.

	• uses pseudocode to symbolise an unambiguous algorithm that – implements control structures consistent with a coherent and logical solution **[1 mark]** – detects hazard locations **[1 mark]** – prioritises passage for emergency vehicles **[1 mark]** – keeps pedestrians safe **[1 mark]** • incorporates – one code library function **[1 mark]** – a second code library function **[1 mark]** – a third code library function **[1 mark]** • includes code comments to identify where criteria are addressed **[1 mark]**

 © State of Queensland (QCAA), CC BY 4.0.

Final revision routine

1. Redo one question without looking at the model.
2. Use the marking-point tables to check that every required feature is visible.
3. For a desk check, add a row whenever a loop repeats or a tracked variable changes.
4. For construction questions, underline each criterion and annotate the block that addresses it.
5. Write one personal “watch for” statement before the next practice paper.

My next focus: _____

Sources: QCAA Digital Solutions external assessment question-response books and marking guides, 2021–25. © State of Queensland (QCAA), licensed CC BY 4.0.