



# Cambridge O Level

CANDIDATE  
NAMECENTRE  
NUMBER

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|--|--|--|--|--|

CANDIDATE  
NUMBER

|  |  |  |  |
|--|--|--|--|
|  |  |  |  |
|--|--|--|--|

## COMPUTER SCIENCE

2210/22

Paper 2 Algorithms, Programming and Logic

October/November 2025

1 hour 45 minutes

You must answer on the question paper.

No additional materials are needed.

### INSTRUCTIONS

- Answer **all** questions.
- Use a black or dark blue pen. You may use an HB pencil for any diagrams or graphs.
- Write your name, centre number and candidate number in the boxes at the top of the page.
- Write your answer to each question in the space provided.
- Do **not** use an erasable pen or correction fluid.
- Do **not** write on any bar codes.
- Calculators must **not** be used in this paper.

### INFORMATION

- The total mark for this paper is 75.
- The number of marks for each question or part question is shown in brackets [ ].
- No marks will be awarded for using brand names of software packages or hardware.

This document has **16** pages. Any blank pages are indicated.



- 1 Tick (✓) **one** box to complete this sentence.

The result of the arithmetic operation  $4^2$  is

- |          |    |                          |
|----------|----|--------------------------|
| <b>A</b> | 2  | <input type="checkbox"/> |
| <b>B</b> | 8  | <input type="checkbox"/> |
| <b>C</b> | 16 | <input type="checkbox"/> |
| <b>D</b> | 42 | <input type="checkbox"/> |

[1]

- 2 This pseudocode algorithm is intended to input a first name from a user and search for it in a two-dimensional (2D) array. If the name is found, the algorithm outputs all the remaining details for that name. The algorithm will continue to allow other names to be entered for searching, until the user input stops it.

```

01 DECLARE Contacts : ARRAY[1:500, 1:4] OF REAL
02 DECLARE Row : INTEGER
03 DECLARE Column : INTEGER
04 DECLARE Continue : BOOLEAN
05 DECLARE Stop : BOOLEAN
06 DECLARE FirstName : STRING
07 Continue ← TRUE
08 Stop ← FALSE
09 WHILE Continue
10     Row ← 1
11     OUTPUT "Enter a first name "
12     OUTPUT FirstName
13     REPEAT
14         IF Contacts[Row, 1] = FirstName
15             THEN
16                 FOR Column ← 1 TO 4
17                     OUTPUT Contacts[Row, 1]
18                 NEXT Column
19                 Stop ← TRUE
20             ELSE
21                 Row ← Row + 1
22             ENDIF
23     NEXT Stop OR Row > 500
24     OUTPUT "Search for another name? (Y or N)"
25     INPUT Answer
26     IF Answer = 'N' OR Answer = 'n'
27         THEN
28             Continue ← TRUE
29     ENDIF
30 ENDWHILE

```

- (a) Identify the line numbers of **five** errors in the pseudocode and suggest a correction for each error.

Error 1 line number .....

Correction .....



Error 2 line number .....

Correction .....

.....

Error 3 line number .....

Correction .....

.....

Error 4 line number .....

Correction .....

.....

Error 5 line number .....

Correction .....

.....

[5]

- (b) A procedure `NewData` allows data for new contacts to be entered and stored in the 2D array `Contacts[]`. A parameter, `Number`, is used to pass values from the main algorithm to the procedure for the number of new contacts. Next, four pieces of contact data for each new contact are entered and stored at the start of the array.

- (i) Complete the pseudocode for `PROCEDURE NewData`

`PROCEDURE NewData (.....)`

`FOR Row ← 1 TO .....`

`FOR Column ← 1 TO 4`

`INPUT .....`

.....

.....

`ENDPROCEDURE`

[4]

- (ii) Write the pseudocode to input the number of new contacts and to use procedure `NewData`. A variable declaration is **not** required.

.....

.....

.....

..... [2]



(c) A first name is used as the search key in the algorithm on page 2. If the name is found, all the data for that name is output and the algorithm stops searching.

(i) Explain why the given algorithm may **not** always give the expected results.

.....

.....

..... [2]

(ii) Explain how you could improve the algorithm so that the search is more likely to find the required results.

.....

.....

..... [2]

3 A program is written that will only accept values between  $-99.99$  and  $+99.99$ , inclusive. The program is to be tested.

The table, when completed, shows appropriate test data that matches the type of test data and purpose of the test data.

Complete the table by inserting the missing information.

| Test data | Type of test data | Purpose of test data                                                         |
|-----------|-------------------|------------------------------------------------------------------------------|
| 0         |                   | .....<br>.....                                                               |
|           | boundary          | .....<br>.....                                                               |
|           |                   | to make sure that the program rejects data that is outside acceptable limits |

[6]





**4 (a)** A length check is a type of validation check.

State the purpose of a length check.

..... [1]

**(b)** An email address can be validated by checking that it contains an @ symbol.

Write the pseudocode for an algorithm that will allow an email address to be input and check whether or not it contains an @ symbol.

Output a suitable message with the result.

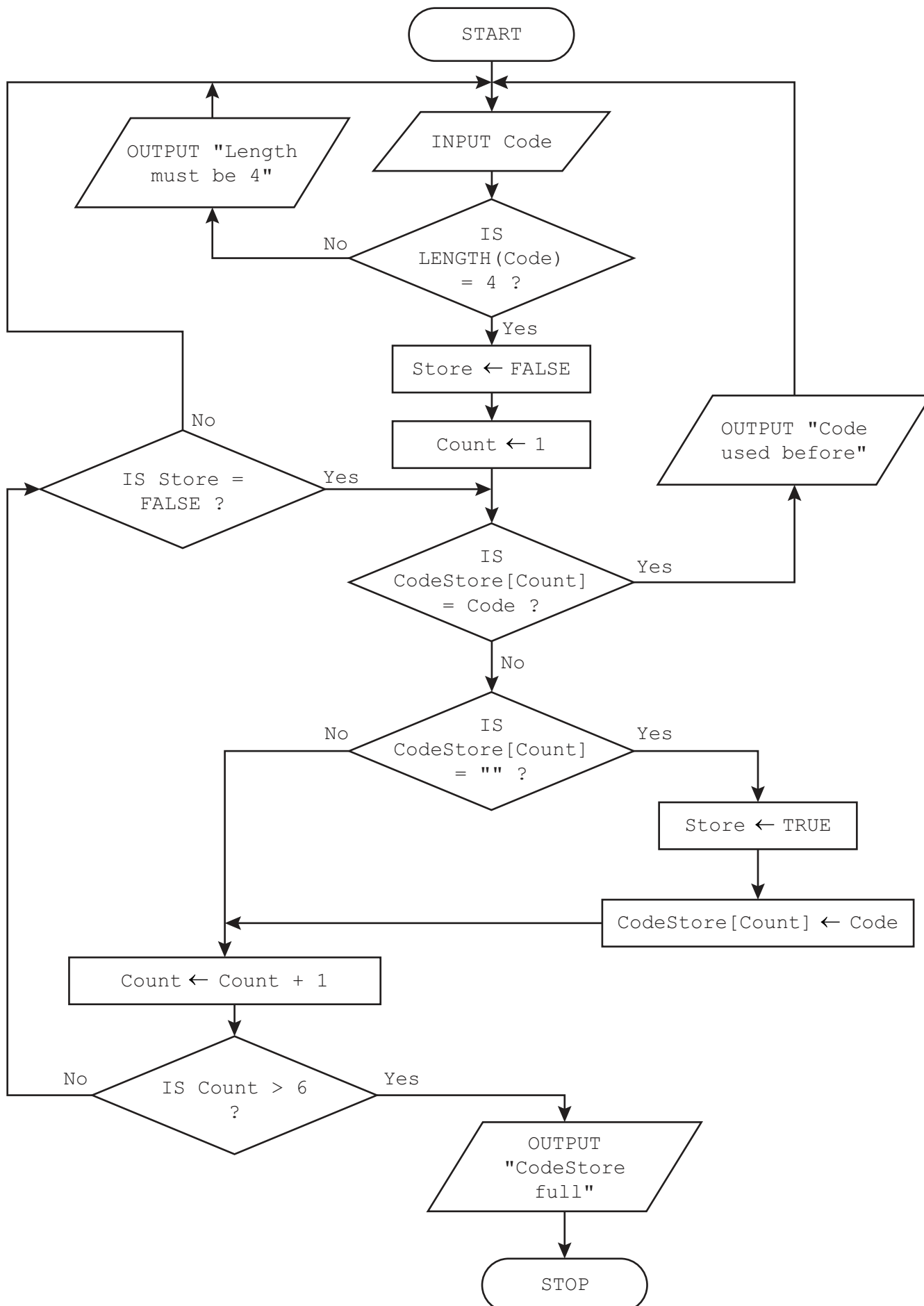
Use the string function `SUBSTRING(EmailAddress, Start, 1)` in your solution to return a string of length 1 character, starting at position `Start`, where `EmailAddress` is the identifier representing your input.

You do **not** need to declare any variables for this algorithm, but variables must be initialised as necessary. Assume there will **not** be more than **one** @ symbol in the email address input.

[6]



5 This flowchart represents an algorithm.





**(a)** The first data row of the trace table shows the values currently stored in `CodeStore[]`

Complete the trace table for the algorithm, using the input data:

7686, Face, Speed, 432U, Yp79

[illegible]

[6]





(b) Describe the processes in the algorithm on page 6.

.....

.....

.....

.....

.....

..... [3]

(c) The algorithm on page 6 requires the array `CodeStore[]` to be declared as a one-dimensional (1D) string array with six elements that are initially set to the null string "".

Write the pseudocode required to declare and initialise the array `CodeStore[]`  
You do need to declare any variables used.

.....

.....

.....

.....

.....

.....

.....

..... [4]



6 Consider the logic expression:

$$X = (J \text{ XOR NOT } K) \text{ NAND NOT } L$$

- (a) Draw a logic circuit for the logic expression.  
Each logic gate must have a maximum of **two** inputs.  
Do **not** simplify the logic expression.



[4]

- (b) Complete the truth table for the given logic expression.

| J | K | L | Working space | X |
|---|---|---|---------------|---|
| 0 | 0 | 0 |               |   |
| 0 | 0 | 1 |               |   |
| 0 | 1 | 0 |               |   |
| 0 | 1 | 1 |               |   |
| 1 | 0 | 0 |               |   |
| 1 | 0 | 1 |               |   |
| 1 | 1 | 0 |               |   |
| 1 | 1 | 1 |               |   |

[4]



- 7 A database table **WATERFOWL** stores some details of a range of birds found in the UK that live on or near water. The length and wingspan dimensions are in centimetres and the typical lifespan is in years.

| Species             | Family | Length | Wingspan | CoastalHabitat | Lifespan |
|---------------------|--------|--------|----------|----------------|----------|
| Barnacle goose      | Goose  | 63.5   | 130.0    | FALSE          | 14       |
| Bewick's swan       | Swan   | 120.2  | 190.5    | TRUE           | 8        |
| Brent goose         | Goose  | 58.0   | 94.3     | TRUE           | 11       |
| Canada goose        | Goose  | 78.5   | 160.0    | FALSE          | 6        |
| Eider               | Duck   | 58.0   | 94.5     | TRUE           | 8        |
| Goldeneye           | Duck   | 44.5   | 72.1     | TRUE           | 6        |
| Goosander           | Duck   | 63.3   | 90.0     | TRUE           | 7        |
| Long-tailed duck    | Duck   | 44.0   | 76.6     | TRUE           | 5        |
| Mallard             | Duck   | 59.7   | 90.8     | FALSE          | 3        |
| Mute swan           | Swan   | 150.0  | 220.2    | FALSE          | 10       |
| Pink-footed goose   | Goose  | 69.6   | 150.0    | TRUE           | 8        |
| Pintail             | Duck   | 60.0   | 88.6     | TRUE           | 3        |
| Smew                | Duck   | 41.5   | 62.1     | FALSE          | 7        |
| White-fronted goose | Goose  | 71.5   | 140.0    | FALSE          | 6        |
| Whooper swan        | Swan   | 150.0  | 230.0    | TRUE           | 9        |

- (a) State the purpose of a primary key in a database table.

.....  
 ..... [1]

- (b) Complete the table for each of the given data types, to identify **one** field from the table **WATERFOWL** that is most likely to use it.

Each field must be different.

| Data type | Field |
|-----------|-------|
| Boolean   |       |
| Integer   |       |
| Real      |       |
| Text      |       |

[2]



(c) Give the output that would be produced by the structured query language (SQL) statement:

```
SELECT Species, Length, Wingspan
FROM WATERFOWL
WHERE Family = "Swan"
ORDER BY Lifespan;
```

.....

.....

.....

.....

.....

..... [3]

(d) Complete the SQL statement to list only the species, family and lifespan of all waterfowl whose typical lifespan is at least 10 years, sorted in order of species.

SELECT .....

.....

.....

..... [4]



- 8 A program is required to test the fairness of the random number generator.

The one-dimensional (1D) array `RandomNumber[]` is used to store 100 000 random integers.

The two-dimensional (2D) array `CountedNumber[]` is used to store the integers 1 to 10, inclusive, and the frequency of each integer (the number of times each integer was generated).

Write a program that meets the following requirements:

- Generate 100 000 random integers between 1 and 10, inclusive.
- Store each integer in the appropriate array.
- Calculate the frequency of each of the integers 1 to 10 and store these values in the appropriate array.
- Sort the contents of the counted number array in **descending order** of frequency.
- Calculate the chance of generating each of the integers 1 to 10, rounded to four decimal places.  
Use the formula:  $\text{chance} = \text{frequency} / 100000$
- Output each of the integers 1 to 10 in the order they are stored in the counted number array, along with the chance of generating them.

The following is an example of an algorithm that sorts a 2D array in an ascending order:

```
F ← TRUE
WHILE F DO
  F ← FALSE
  FOR I ← 1 TO 9
    IF A[I, 2] > A[I + 1, 2]
      THEN
        T1 ← A[I, 1]
        T2 ← A[I, 2]
        A[I, 1] ← A[I + 1, 1]
        A[I, 2] ← A[I + 1, 2]
        A[I + 1, 1] ← T1
        A[I + 1, 2] ← T2
      F ← TRUE
    ENDIF
  NEXT I
ENDWHILE
```

You must use pseudocode or program code **and** add comments to explain how your code works.

All arrays, variables or constants used must be declared for this algorithm.

All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

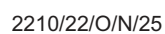
.....

.....





© UCLES 2025



**[Turn over**



..... [15]







Permission to reproduce items where third-party owned material protected by copyright is included has been sought and cleared where possible. Every reasonable effort has been made by the publisher (UCLES) to trace copyright holders, but if any items requiring clearance have unwittingly been included, the publisher will be pleased to make amends at the earliest possible opportunity.

To avoid the issue of disclosure of answer-related information to candidates, all copyright acknowledgements are reproduced online in the Cambridge Assessment International Education Copyright Acknowledgements Booklet. This is produced for each series of examinations and is freely available to download at [www.cambridgeinternational.org](http://www.cambridgeinternational.org) after the live examination series.

Cambridge Assessment International Education is part of Cambridge Assessment. Cambridge Assessment is the brand name of the University of Cambridge Local Examinations Syndicate (UCLES), which is a department of the University of Cambridge.

