* 0000800000001 *

Cambridge Assessment International Education

# Cambridge O Level

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

*3393343827*

**COMPUTER SCIENCE** **2210/22**

Paper 2 Algorithms, Programming and Logic **October/November 2024**

**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.

DC (KAS) 347703
© UCLES 2024

**[Turn over**

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

A solution to a problem may be represented using pseudocode, flowcharts or

**A** procedures. ☐

**B** processes. ☐

**C** structure diagrams. ☐

**D** sub-systems. ☐

[1]

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

A pseudocode example of a selection statement is

**A** `CALL Sorting(Value1, Value2)` ☐

**B** `DECLARE Count : INTEGER` ☐

**C** `IF X = 7 THEN Y ← 21 ENDIF` ☐

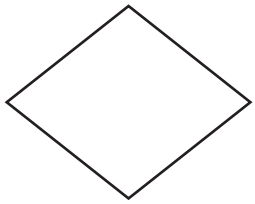
**D** `WHILE X <> -1 DO` ☐

[1]

**3** **Four** flowchart symbols and **five** purposes are shown.

**(a)** Draw **one** line to link each flowchart symbol to its correct purpose.

**Not** all purposes will be used.

| Flowchart symbol | Purpose |
|---|---|
|  | subroutine |
|  | process |
|  | flow |
| | decision |
|  | terminator |

[4]



DO NOT WRITE IN THIS MARGIN DO NOT WRITE IN THIS MARGIN DO NOT WRITE IN THIS MARGIN DO NOT WRITE IN THIS MARGIN DO NOT WRITE IN THIS MARGIN

**(b)** An algorithm needs to total 50 numbers between 1 and 100 inclusive.

Draw a flowchart that:

- uses a count-controlled loop from 1 to 50
- uses an appropriate prompt to ask for a number between 1 and 100
- totals the numbers as they are entered
- outputs the total after the loop has completed with an appropriate message.

[6]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

**4** This pseudocode algorithm is intended to sort a pre-populated one-dimensional (1D) array named `ItemList` into alphabetical order using a bubble sort.

```
01 DECLARE ItemList : ARRAY[1:100] OF STRING
02 DECLARE Counter : STRING
03 DECLARE Limit : INTEGER
04 DECLARE Pass : INTEGER
05 DECLARE Swapped : BOOLEAN
06 DECLARE Temp : STRING
07 Limit ← 100
08 Pass ← 1
09 Temp ← TRUE
10 WHILE Swapped = TRUE OR Pass <= Limit - 1 DO
11     Swapped ← FALSE
12     FOR Counter ← 1 TO Limit – Pass
13         IF ItemList[Counter] > ItemList[Counter + 1]
14           THEN
15             Temp ← ItemList[Counter]
16             ItemList[Counter] ← ItemList[Counter + 1]
17             ItemList[Counter] ← Temp
18             Swapped ← TRUE
19         ENDCASE
20         Pass ← Pass + 1
21     NEXT Counter
22 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 .....

.....

DO NOT WRITE IN THIS MARGIN

Error 5 line number .....

Correction .....

.....

[5]

**(b)** A bubble sort algorithm can be written to include features that make it more efficient.

Explain why the **corrected** bubble sort algorithm is efficient.

.....

.....

.....

.....

.....

..... [3]

**5** Analysis is one stage in the program development life cycle.

**(a)** State **one** other stage in the program development life cycle.

..... [1]

**(b)** Describe the analysis stage of the program development life cycle.

.....

.....

.....

.....

.....

..... [3]

**6** Outline **one** type of verification check that could be used when inputting data.

.....

.....

.....

..... [2]

DO NOT WRITE IN THIS MARGIN

**7** This pseudocode represents an algorithm.

An input of –1 will terminate the algorithm.

```
DECLARE Count : INTEGER
DECLARE Answer : INTEGER
DECLARE Value : INTEGER
REPEAT
    INPUT Value
    IF Value <> -1
      THEN
        Answer ← Value
        FOR Count ← Value - 1 TO 1 STEP -1
            Answer ← Answer * Count
        NEXT Count
        OUTPUT Answer
    ENDIF
UNTIL Value = -1
```

**(a)** Complete the trace table for the input data:

5, 6, –1, 20, 9, 4

| Value | Count | Answer | OUTPUT |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

[5]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

**(b)** State the purpose of this algorithm.

.....

..... [1]

**(c)** Describe the problem that would be caused in this algorithm if a `Value` of 1, 0 or less than –1 was input.

.....

.....

.....

..... [2]

**8** Different types of test data are used during program development to make sure a program works as intended. A program being developed takes as input whole numbers that are **not** greater than 80.

Identify **two** items of test data to test the whole number limit of 80.

Explain the reason for your choice of the data in each case.

Test data 1 .....

Reason for choice .....

.....

Test data 2 .....

Reason for choice .....

.....

[4]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

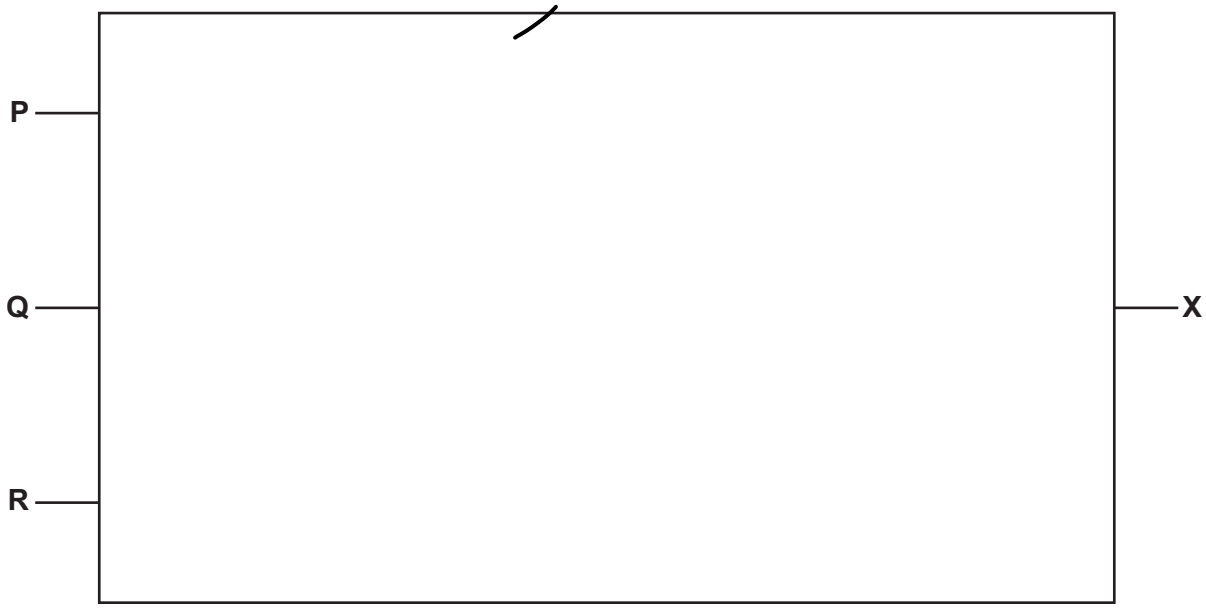
**9** Consider the logic expression:

**X** = (NOT **P** OR **Q**) NAND (**Q** XOR **R**)

**(a)** Draw a logic circuit for this 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.

| P | Q | R | 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]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

**10** The function `LENGTH(X)` calculates the length of a string `X`

Write the pseudocode statements to:

- allow a line of text to be input to an appropriate variable
- store this line of text in a text file called `Main.txt`
- calculate the length of the line of text and output the text in lower case along with its length
- store the lower-case line of text in a text file called `Lowercase.txt`

Make sure that any variables used are declared and that both text files are closed after they have been used.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [6]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

**11** A database table called `Booking28` stores details of hotel rooms and bookings for the week beginning Monday 7 July 2025.

| RoomNo | Type | Guests | Rate$ | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---|---|---|---|---|---|---|---|---|---|---|
| 101D | Double | 2 | 99.99 | T | T | T | T | F | T | T |
| 102D | Double | 2 | 99.99 | T | T | T | F | T | T | T |
| 103F | Family | 4 | 150.00 | T | T | T | T | T | T | T |
| 104S | Single | 1 | 72.50 | F | T | T | F | T | T | T |
| 105S | Single | 1 | 72.50 | F | T | T | F | T | T | T |
| 106T | Twin | 2 | 120.00 | T | T | T | T | F | T | T |
| 201F | Family | 4 | 160.00 | F | F | T | T | T | T | T |
| 202D | Double | 2 | 120.00 | T | F | T | T | T | T | T |
| 203T | Twin | 2 | 120.00 | T | F | T | T | T | T | T |
| 204T | Twin | 2 | 125.00 | T | F | T | F | T | T | T |
| 205S | Single | 1 | 79.99 | T | F | T | T | F | T | T |
| 301D | Double | 2 | 200.00 | F | T | T | F | F | T | T |
| 302T | Twin | 2 | 200.00 | T | T | T | T | F | T | T |
| 303P | Suite | 4 | 500.00 | T | T | T | T | F | T | T |
| 304P | Suite | 6 | 700.00 | F | F | F | F | T | T | T |

**(a)** State the number of fields and records in this database table.

Fields .....

Records .....
[2]

**(b)** State the reason why the `Type` field would **not** be suitable as a primary key.

.....

..... [1]

DO NOT WRITE IN THIS MARGIN

DO NOT WRITE IN THIS MARGIN

DO NOT WRITE IN THIS MARGIN

DO NOT WRITE IN THIS MARGIN

DO NOT WRITE IN THIS MARGIN

**(c)** The database uses only the data types:

- alphanumeric
- character
- Boolean
- integer
- real
- date/time.

Complete the table to show the fields that could have the given data types.

Only **one** field name is required in each box and each field name must be different.

| Field | Data type |
|---|---|
| | alphanumeric |
| | Boolean |
| | real |
| | integer |

[2]

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

```
SELECT RoomNo, Type, Guests, Rate$

FROM Booking28

WHERE Mon <> T;
```

.....

.....

.....

.....

.....

..... [3]

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

**12** A one-dimensional (1D) array `Rooms[]` contains the names of up to 20 rooms in a house.
A two-dimensional (2D) array `Dimensions[]` is used to store the length, width and area of each room.

The position of any room's data is the same in both arrays. For example, the data in index 5 of `Dimensions[]` belongs to the room in index 5 of `Rooms[]`

The variable `Number` stores the number of rooms for which data is to be input. There must be at least 3 rooms but no more than 20.

Write a program that meets the following requirements:

- allows the number of rooms for which data is required to be input, stored and validated
- allows the name of the room and the length and width of the room, in metres, to be entered and stored
- allows the area of each room to be calculated as length multiplied by width and stored as square metres rounded to two decimal places
- calculates the average size of all the rooms by area, in square metres, rounded to two decimal places
- finds the largest room and smallest room by area
- outputs the names of all rooms with their dimensions and area
- outputs the names of the largest room and smallest room by area
- outputs the total area of the house and the average size of all the rooms by area.

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

You do **not** need to declare any arrays or variables; you may assume that this has already been done.

All inputs and outputs must contain suitable messages.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

.....

.....

.....

**BLANK PAGE**

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

**BLANK PAGE**

DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN
DO NOT WRITE IN THIS MARGIN

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 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.

© UCLES 2024