2210_s23_qp_22

Wednesday, 17 April 2024 11:31 PM

2210_s23_qp_22

Cambridge Assessment International Education

Cambridge O Level

CANDIDATE NAME: ZAK

CENTRE NUMBER: Home

CANDIDATE NUMBER:

COMPUTER SCIENCE **2210/22**

Paper 2 Algorithms, Programming and Logic **May/June 2023**

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.

[Turn over

1 Tick (✓) **one** box to identify the first stage of the program development life cycle.

A Analysis ✓

B Coding

C Design

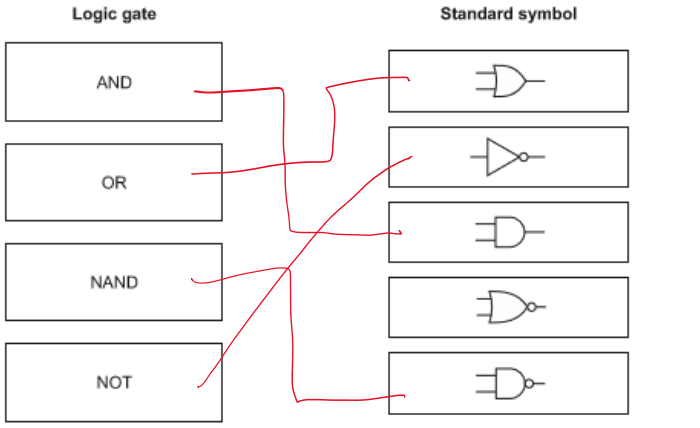
D Testing

PDLC
1. Analysis
2. Design
3. Coding
4. Testing

[1]

2 **Four** logic gates and **five** standard symbols for logic gates are shown.

Draw **one** line to link each logic gate to its standard symbol. **Not** all standard symbols will be used.



[4]

3 Identify **three** different ways that the design of a solution to a problem can be presented.

1 Pseudocode.

2 Flowchart.

3 Structure chart.

[3]

4 A program needs to make sure the value input for a measurement meets the following rules:
- the value is a positive number
- a value is always input
- the value is less than 1000.

(a) Describe the validation checks that the programmer would need to use.

1. Presence check: To ensure value has been input.

2. Limit Check: To ensure the value is greater than zero, or value is less than 1000.

3. Type: To ensure a number only is input

[3]

(b) The program needs editing to include a double entry check for the value input.

(i) State why this check needs to be included.

To ensure the accuracy and consistency of the critical data entered. (Verification). [1]

(ii) The input value needs to be stored in the variable `Measurement`
Write pseudocode to perform the double entry check until a successful input is made.

```
REPEAT
    OUTPUT "Enter the measurement value:"
    INPUT Measurement
    OUTPUT "Re-enter the measurement value to confirm:"
    INPUT Measurement2
    IF Measurement <> Measurement2 THEN
        OUTPUT 'Error: The values don't match, please try again"
    ENDIF
UNTIL Measurement = Measurement2.
```
[3]


```
REPEAT
    INPUT "Enter a number (1-999)", Num
    IF Num <= 0 THEN
        OUTPUT "Error: Please enter a +ve nb"
    ELSEIF Num >= 1000 THEN
        OUTPUT "Error: Please enter <1000"
    ELSEIF Type(Num) <> INT OR
           Type(Num) <> REAL THEN
        OUTPUT "Only value in numbers is allowed".
    ENDIF
UNTIL Num > 0 AND Num < 1000
```

5 Due to an issue with Question 5, the question has been removed from the question paper.

6 State **three** different features of a high-level programming language that a programmer could use to make sure that their program will be easier to understand by another programmer.
Give an example for each feature.

Feature 1 Comments.

Example // Calculate the Sum of two numbers.
Total ← Number1 + Number2.

Feature 2 Meaningful Identifier Names.

Example // Good practice for naming variables.
DECLARE CustomerTotalPurchases : REAL

Feature 3 Modularity. (Use of procedures and functions).

Example NetSal ← CalculateNetSalary (GrossSal, TaxRate)

[6]

BLANK PAGE

[Turn over

7 An algorithm has been written in pseudocode to calculate a check digit for a four-digit number.
The algorithm then outputs the five-digit number including the check digit.
The algorithm stops when –1 is input as the fourth digit.

```
01 Flag ← FALSE
02 REPEAT
03     Total ← 0
04     FOR Counter ← 1 TO 4
05       OUTPUT "Enter a digit ", Counter
06       INPUT Number[Counter]
07       Total ← Total + Number * Counter
08       IF Number[Counter] = 0
09         THEN
10           Flag ← TRUE
11       ENDIF
12     NEXT Counter
13     IF NOT Flag
14       THEN
15         Number[5] ← MOD(Total, 10)
16         FOR Counter ← 0 TO 5
17           OUTPUT Number[Counter]
18         NEXT
19     ENDIF
20 UNTIL Flag
```

Count controlled loop

Constructs:
1. Sequence
2. Iteration/Loops
3. Decision
4. Assignments

Number[Counter] * Counter

−1

IF FLAG = FALSE

Totalling?
X ← X + Y

(a) Give the line number(s) for the statements showing:

Totalling 07

Count-controlled loop 04–12

Post-condition loop 02–20

[3]

(b) Identify the **three** errors in the pseudocode and suggest a correction for each error.

Error 1 07

Correction .....

.....

Error 2 .....

Correction .....

.....

Error 3 .....

Correction .....

.....

[3]

(c) The algorithm does **not** check that each input is a single digit.
Identify the place in the algorithm where this check should occur.
Write pseudocode for this check.
Your pseudocode must make sure that the input is a single digit and checks for −1

Place in algorithm ..... After line 6.

Pseudocode .....

```
6     INPUT Number[Counter]
6.1   IF (Number[Counter] < -1) OR (Number[Counter] > 9) THEN
6.2   OUTPUT "Error: Enter a valid single digit (-1 To Stop)"
6.3   REPEAT
6.4       INPUT Number[Count]
6.5   UNTIL Number[Counter] >= -1 AND Number[Counter] <= 9
6.6   ENDIF.
```

[4]

8 Consider this logic expression.

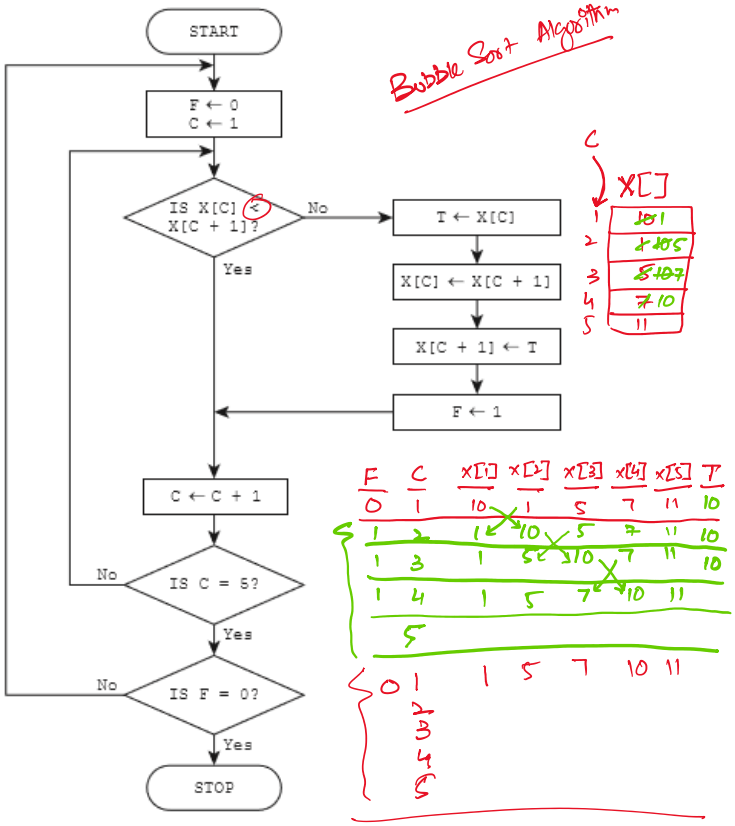
X = (**A** OR **B**) AND (NOT **B** AND **C**)

Complete the truth table for this logic expression. X=((A+B).(B'C))

A	B	B'	C	Working space (A+B)	Working space (B'C)	X
0	0	1	0	0	0	0
0	0	1	1	0	1	0
0	1	0	0	1	0	0
0	1	0	1	1	0	0
1	0	1	0	1	0	0
1	0	1	1	1	1	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0

[4]

9 This flowchart represents an algorithm.



(a) The array `X[1:5]` used in the flowchart contains this data:

X[1]	X[2]	X[3]	X[4]	X[5]
10	1	5	7	11

Complete the trace table by using the data given in the array.

F	C	X[1]	X[2]	X[3]	X[4]	X[5]	T
		10	1	5	7	11	

[5]

(b) Describe what the algorithm represented by the flowchart is doing.

- It is Sorting the X[] array in ascending order.
- It is Bubble Sort.

[2]

[Turn over

10 A music streaming service has a new database table named `Songs` to store details of songs available for streaming. The table contains the fields:

- `SongNumber` – the catalogue number, for example AG123
- `Title` – the title of the song
- `Author` – the name of the song writer(s)
- `Singer` – the name of the singer(s)
- `Genre` – the type of music, for example rock
- `Minutes` – the length of the song in minutes, for example 3.75
- `Recorded` – the date the song was recorded.

(a) Identify the field that will be the most appropriate primary key for this table.

SongNumber. [1]

(b) Complete the table to identify the most appropriate data type for the fields in `Songs`

Field	Data type
`SongNumber`	Text
`Title`	Text
`Recorded`	Date/Time
`Minutes`	Real.

[2]

(c) Explain the purpose of the structured query language (SQL) statements.

```
SUM (Minutes) FROM Songs WHERE Genre = "rock";
```
To Sum the minutes field values in records when Genre = "rock".

```
COUNT (Title) FROM Songs WHERE Genre = "rock";
```
To Count, How many, records are having rock saved in Genre field.

[3]

11 The variables `P` and `Q` are used to store data in a program. `P` stores a string. `Q` stores a character.

(a) Write pseudocode statements to declare the variables `P` and `Q`, store `"The world"` in `P` and store `'W'` in `Q`

```
DECLARE P : STRING
DECLARE Q : CHAR
P ← "The World"
Q ← 'W'
```

[2]

(b) Write a pseudocode algorithm to:
- convert `P` to upper case
- find the position of `Q` in the string `P` (the first character in this string is in position 1)
- store the position of `Q` in the variable `Position`

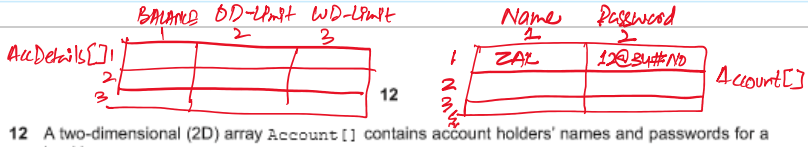
```
P ← UCASE(P)
DECLARE Position : INTEGER
Position ← 0
FOR i ← 1 TO LENGTH(P)
    THISCHAR ← SUBSTRING(P, i, 1)
    IF THISCHAR = Q THEN Position = i
NEXT i
IF Position = 0 THEN
    OUTPUT "Q NOT FOUND!!!"
ELSE
    OUTPUT "Q FOUND at: ", Position
ENDIF
```

[4]

(c) Give the value of `Position` after the algorithm has been executed with the data in **question 11(a)**.

5

[1]



12 A two-dimensional (2D) array `Account[]` contains account holders' names and passwords for a banking program.

A 2D array `AccDetails[]` has three columns containing the following details:
- column one stores the balance – the amount of money in the account, for example 250.00
- column two stores the overdraft limit – the maximum total amount an account holder can borrow from the bank after the account balance reaches 0.00, for example 100.00
- column three stores the withdrawal limit – the amount of money that can be withdrawn at one time, for example 200.00

The amount of money in a bank account can be negative (overdrawn) but **not** by more than the overdraft limit.
For example, an account with an overdraft limit of 100.00 must have a balance that is greater than or equal to –100.00

Suitable error messages must be displayed if a withdrawal cannot take place, for example if the overdraft limit or the size of withdrawal is exceeded.

The bank account ID gives the index of each account holder's data held in the two arrays.
For example, account ID 20's details would be held in:
`Account[20,1]` and `Account[20,2]`
`AccDetails[20,1]` `AccDetails[20,2]` and `AccDetails[20,3]`

The variable `Size` contains the number of accounts.

Account[20,1] and Account[20,2]
AccDetails[20,1] AccDetails[20,2] and AccDetails[20,3]

The variable `Size` contains the number of accounts.

The arrays and variable `Size` have already been set up and the data stored.

Write a program that meets the following requirements:

- checks the account ID exists and the name and password entered by the account holder match the name and password stored in `Account[]` before any action can take place
- displays a menu showing the four actions available for the account holder to choose from:
    1. display balance
    2. withdraw money
    3. deposit money
    4. exit
- allows an action to be chosen and completed. Each action is completed by a procedure with a parameter of the account ID.

You must use pseudocode or program code **and** add comments to explain how your code works. All inputs and outputs must contain suitable messages.

You only need to declare any local arrays and local variables that you use.

You do **not** need to declare and initialise the data in the global arrays `Account[]` and `AccDetails[]` and the variable `Size`

[Turn over

[15]

BLANK PAGE

BLANK PAGE