Cambridge Assessment
International Education

Example Responses – Paper 2

Cambridge IGCSE™ / IGCSE (9–1)
Computer Science 0478 / 0984

Cambridge O Level
Computer Science 2210

For examination from 2023

Cambridge Pathway

Contents

Introduction

The main aim of this booklet is to exemplify standards for those teaching Cambridge IGCSE / IGCSE (9-1) / O Level Information and Communication Technology 0478 / 0984 / 2210.

This booklet contains responses to all questions from June 2023 Paper 22, which have been written by a Cambridge examiner. Responses are accompanied by a brief commentary highlighting common errors and misconceptions where they are relevant.

The question papers and mark schemes are available to download from the School Support Hub

0478 / 0984 / 2210 June 2023 Question Paper 22

0478 / 0984 / 2210 June 2023 Mark Scheme 22

Past exam resources and other teaching and learning resources are available from the School Support Hub

Question 1

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

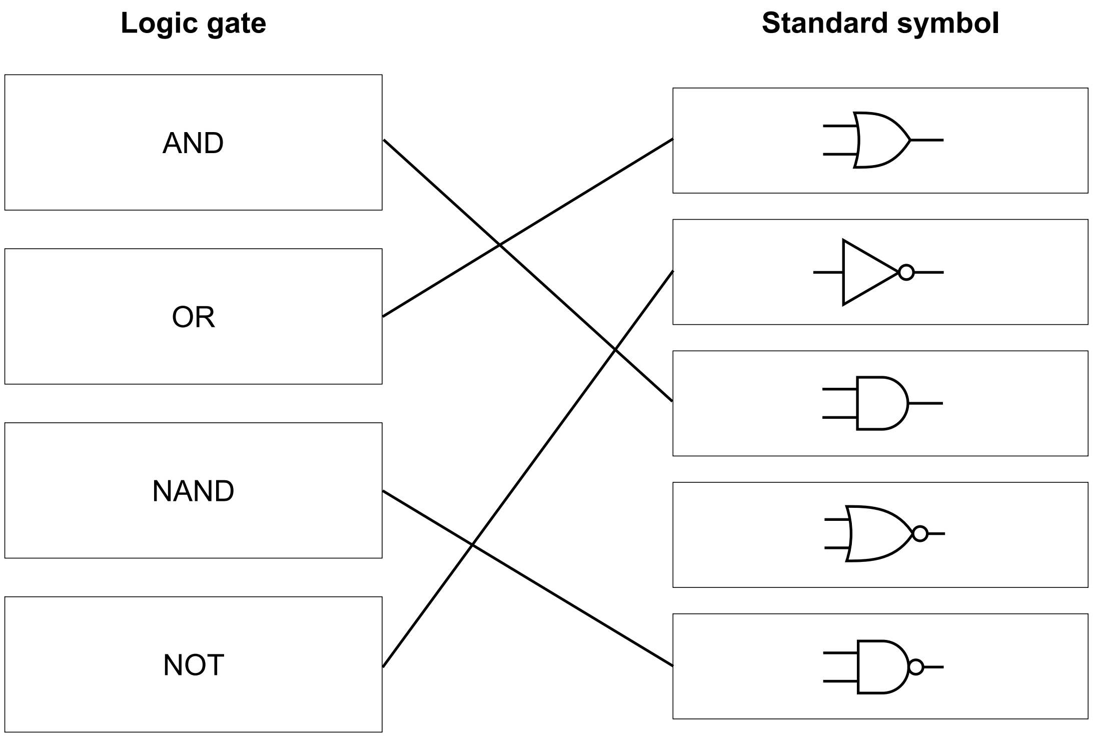
A	Analysis	✓
B	Coding	
C	Design	
D	Testing	

[1]

Question 2

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]

Examiner comment

Most candidates drew one line form each logic gate to the most appropriate standard symbol as required.

Question 3

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

1 structure diagram

2 flowchart

3 pseudocode

[3]

Examiner comment

Candidates listed three different ways to present the design of a solution to a problem. Some candidates misunderstood and listed other aspects of designing a solution, such as analysis, design, coding or testing.

Question 4

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.

A range check could be used to make sure values greater than 0 and less than 1000 are entered. A presence check could be used to make sure that a value has been entered. A type check could be used to make sure that a number has been entered.

[3]

Examiner comment

- Some common issues with this question included candidates not giving both ends of the range check, for example, they stated that the range check makes sure that the number entered is less than 1000. To be awarded the mark, they had to give the full range.
- Some candidates stated that a type check would make sure that a positive number was entered, which isn't true. The type check would only test the data type, for example whether it is a number.

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

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

this check is needed to verify the data

[1]

Examiner comment

- Some candidates incorrectly believed that the double entry check named in the question would make sure that the data entry was correct, which it cannot do. It only checks that both entries are the same, so that it verifies the data.
- Some candidates mixed up verification and validation, so they gave an incorrect reason such as checking that the data entered is reasonable.

(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 "Please enter measurement "
    INPUT Measurement
    OUTPUT "Please re-enter measurement "
    INPUT MeasurementCheck
UNTIL Measurement = MeasurementCheck
```

[3]

Examiner comment

- Many candidates answered this question well, but the question asked for a pseudocode answer, so candidates were required to use the pseudocode defined in the syllabus.
- Up to 3 marks were awarded for including two inputs, checking that the inputs were the same use of iteration to allow re-input if the inputs were not the same, correct use of the variable `Measurement` as directed in the question.

Question 5

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

Question 6

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 all identifiers should have meaningful names

Example use Sum for a variable to store the result of addition

Feature 2 use comments to explain how different sections of the program works

Example // this part of the program performs a validation check for the input data

Feature 3 use procedures and functions for repeated tasks within a program

Example `CalculateInterest(Deposit, Rate)`

[6]

Examiner comment

Candidates were awarded 1 mark for each correct feature to make a program easier to understand and 1 mark for each corresponding example. Some candidates did not give an example of the feature in the second part, even though the answer space on the paper was divided into features and examples. Instead, they put an explanation of the feature in the example space.

Question 7

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
```

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

Totalling 07

Count-controlled loop 04 to 12

Post-condition loop 02 to 20

[3]

Examiner comment

Candidates had to identify the line numbers in a given piece of pseudocode that represented each of the listed programming features and many candidates answered this well.

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

Error 1 `Line 07 Total ← Total + Number * Counter`

Correction `Total ← Total + Number[Counter] * Counter`

Error 2 `Line 08 IF Number[Counter] = 0`

Correction `IF Number[Counter] = -1`

Error 3 `Line 16 FOR Counter ← 0 TO 5`

Correction `FOR Counter ← 1 TO 5`

[3]

Examiner comment

Candidates had to identify the three lines in the given pseudocode that had errors by stating the line number, or by writing the incorrect pseudocode. They also had to give the correction to the pseudocode. Both parts, the identification and the correction, are needed for each mark.

(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 05

Pseudocode

```
REPEAT
    OUTPUT "Enter a digit "
    INPUT Number[Counter]
UNTIL Number[Counter] = Round(Number[Counter],0) AND
      ((Number[Counter] = -1) OR
      (Number[Counter] > 0 AND Number[Counter] < 10))
```

[4]

Examiner comment

Candidates were awarded 1 mark for correctly identifying where in the algorithm they would place their pseudocode. They were awarded up to 3 marks for correct aspects of their pseudocode, which could include use of a loop, checking that the number input is greater than 0, less than 10, that it is a whole number, that it is equal to −1 or that the length of the number entered is a single digit. Not all of these points were required to be awarded full marks.

Question 8

8 Consider this logic expression.

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

Complete the truth table for this logic expression.

A	**B**	**C**	**Working space** A OR B	NOT B	NOT B AND C	**X**
0	0	0	0	1	0	0
0	0	1	0	1	1	0
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	1	1	0	0
1	0	1	1	1	1	1
1	1	0	1	0	0	0
1	1	1	1	0	0	0

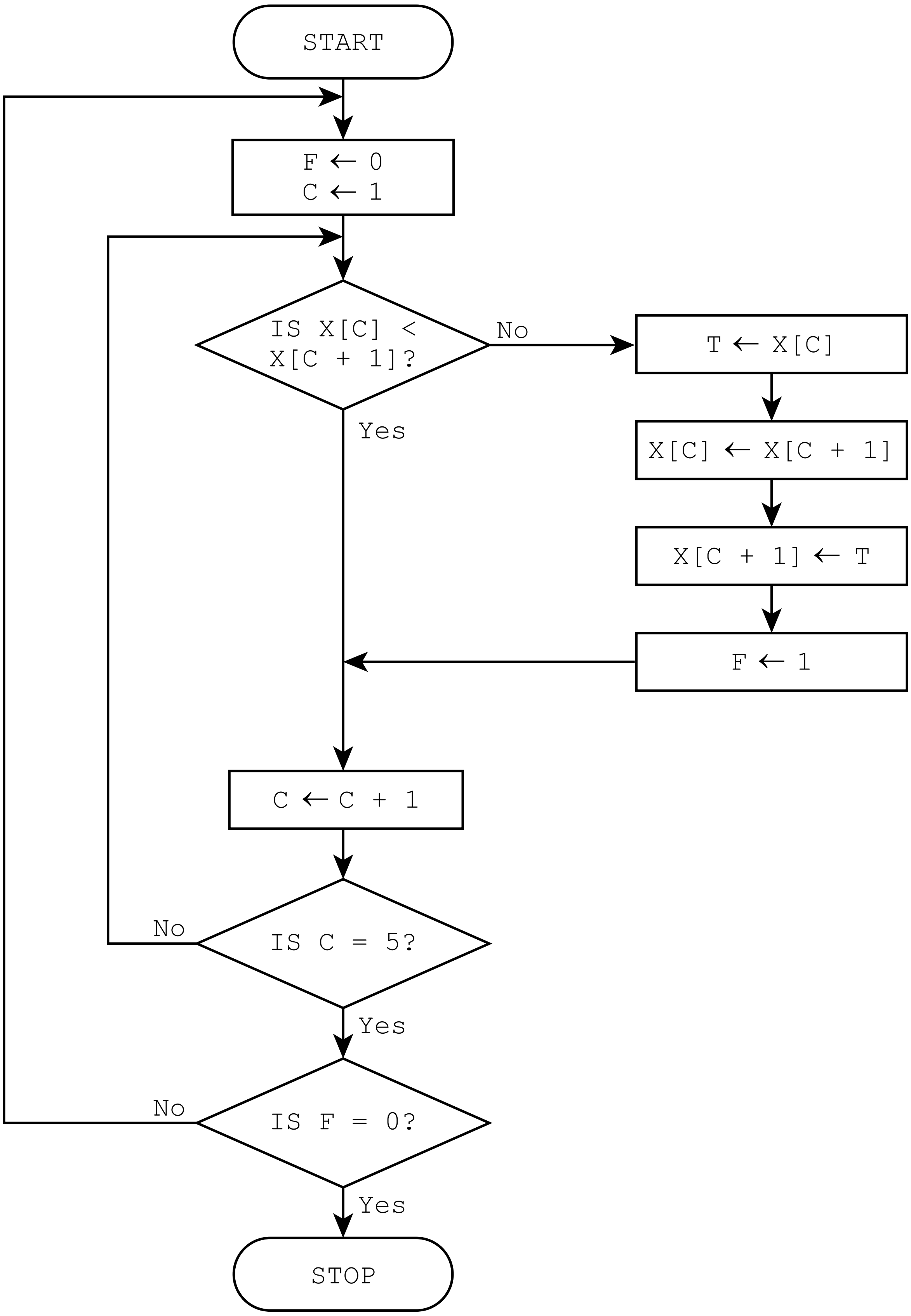
[4]

Examiner comment

Candidates were awarded 1 mark for every two correct digits in the X column, up to a maximum of 4. Candidates did not need to use the working space on this question, or they could use it in a different way to that illustrated. However, the example response shows how it can be used to achieve the final answer, X. The areas shaded grey were given in the question.

Question 9

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	
0	1						10
1	2	1	10				10
1	3		5	10			10
1	4			7	10		
	5						
0	1						
	2						
	3						
	4						
	5						

[5]

Examiner comment

Candidates needed to complete the trace table as shown and were awarded 1 mark for each of the columns F, C and T being correct. They were awarded 2 extra marks if the remaining five columns were also correct, or 1 extra mark if four of the remaining columns were correct. The grey shaded areas were given in the question.

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

the algorithm represents a bubble sort that arranges the data in the array in ascending order [2]

Examiner comment

Candidates were awarded 1 mark for identifying the purpose of the algorithm as being to sort the data in the array, or to use the bubble sort on the data in the array. The second mark was awarded for identifying the fact that the data was sorted in ascending order.

Question 10

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]

Examiner comment

Candidates had to name the field from the given list that would be the most appropriate primary key.

(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
`Minutes`	real

[2]

Examiner comment

- Candidates were awarded 1 mark for identifying two or three correct data types, or 2 marks for identifying all four. The acceptable data types from which to choose are identified in the syllabus in the database section.
- For text, candidates could have alternatively said text / alphanumeric, or just alphanumeric, and for date, they could have written date / time, as these were the options on the syllabus.

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

```
SUM (Minutes) FROM Songs WHERE Genre = "rock";

COUNT (Title) FROM Songs WHERE Genre = "rock";
```

the first SQL statement finds the total number of minutes of rock music

and the second SQL statement finds the total number of rock songs

[3]

Examiner comment

Candidates were awarded 2 marks for explaining the purposes of the given SQL statements as finding the total number of minutes of music and for finding the total number of songs. They were awarded the third mark for noting that in both cases the code was looking at the rock genre.

Question 11

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]

Examiner comment

- Candidates were awarded 1 mark for every two lines of correct pseudocode up to a maximum of 2 marks.
- Candidates were required to use the published version of pseudocode from the syllabus for this question.

(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)
Counter ← 1
Position ← 0
REPEAT
  IF SUBSTRING(P, Counter, 1) = Q
  THEN
  Position ← Counter
  ENDIF
  Counter ← Counter + 1
UNTIL Position <> 0 OR Counter = LENGTH(P)
```

[4]

Examiner comment

- This was a new topic that was added to the syllabus for first testing in 2023 and candidates generally found this question difficult.
- A pseudocode algorithm was required to perform some string handling functions, with 4 marks available for the question and 5 marking points.

- Candidates were awarded marks for:
  - converting the contents of the variable P to upper case using the function UCASE(P)
  - finding the length of the value stored in the variable P using LENGTH(P)
  - using a loop to check for position of the value stored in the variable Q, which is the letter w from part 11a. Any type of loop that works was acceptable, but a condition-controlled loop would be most efficient
  - using the string operation SUBSTRING(P, Counter, 1) to break down the main string into its component letters
  - storing the values in the loop counter in the variable Position if 'w' is found.

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

5

[1]

Examiner comment

Candidates were awarded 1 mark for correctly stating the position of the letter ‘w’ in the given string ‘The world’, as identified by the algorithm.

Question 12

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.

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`

```
// Procedure to check if the details entered are correct
PROCEDURE CheckDetails(AccID : INTEGER) // AccID passed as a parameter
   DECLARE Name, Password : STRING // local variables
```

```
    Valid ← FALSE
    IF AccID <0 OR AccID > Size
      THEN
        OUTPUT "Invalid Account Number"
      ELSE
        OUTPUT "Please Enter Name "
        INPUT Name
        OUTPUT "Please Enter Password "
        INPUT Password
        IF Name <> Account[AccID,1] OR Password <> Account[AccID,2]
          THEN
            OUTPUT "Invalid name or password"
          ELSE
            Valid ← True
        ENDIF
    ENDIF
ENDPROCEDURE

// Procedure to display the current account balance
PROCEDURE Balance(AccID : INTEGER) // AccID passed as a parameter
    OUTPUT "Your balance is ", AccDetails[AccID,1]
ENDPROCEDURE

// Procedure to enable an account withdrawal
PROCEDURE WithDrawal(AccID : INTEGER) // AccID passed as a parameter
    DECLARE Amount : REAL // local variable
    REPEAT
        OUTPUT "Please enter amount to withdraw "
        INPUT Amount
        IF Amount > AccDetails[AccID,3]
          THEN
            OUTPUT "Amount greater than withdrawal limit"
        ENDIF
        IF Amount > AccDetails[AccID,2] + AccDetails[AccID,1]
          THEN
            OUTPUT "Amount greater than cash available"
        ENDIF
        // Withdrawal only allowed if sufficient funds (including overdraft) and
        // withdrawal is within withdrawal limit
        IF Amount <= AccDetails[AccID,3] AND Amount <= AccDetails[AccID,2] +
          AccDetails[AccID,1]
          THEN
            AccDetails[AccID,1] ← AccDetails[AccID,1] - Amount
        ENDIF
    UNTIL Amount <= AccDetails[AccID,3] AND Amount <= AccDetails[AccID,2] +
          AccDetails[AccID,1] AND Amount > 0
ENDPROCEDURE

// Procedure to enable an account deposit
PROCEDURE Deposit(AccID : INTEGER) // AccID passed as a parameter
    DECLARE Amount : REAL // local variable
    REPEAT
        OUTPUT "Please enter a positive amount to deposit "
        INPUT Amount
    UNTIL Amount > 0
    AccDetails[AccID,1] ← AccDetails[AccID,1] + Amount
ENDPROCEDURE

// Declarations of global variables - not required by candidates,
// but part of a perfect solution for information only
DECLARE AccountNumber, Choice : INTEGER
DECLARE Valid, Exit : BOOLEAN

// Input of account number
OUTPUT "Please enter your account number "
INPUT AccountNumber
CheckDetails(AccountNumber)

IF Valid // Valid set in CheckDetails procedure
  THEN
    REPEAT
        // Menu displayed
        OUTPUT "Menu"
        OUTPUT "1. display balance"
        OUTPUT "2. withdraw money"
        OUTPUT "3. deposit money"
        OUTPUT "4. exit"
        OUTPUT "please choose 1, 2, 3 or 4"
        INPUT Choice // menu choice
        // Relevant procedure called based on choice input or program exited
        CASE OF Choice
          1 : Balance(AccountNumber)
          2 : Withdrawal(AccountNumber)
          3 : Deposit(AccountNumber)
          4 : Exit ← TRUE
          OTHERWISE OUTPUT "Invalid choice"
        ENDCASE
    UNTIL Choice = 4
  ELSE
    OUTPUT "Invalid account number "
ENDIF
```

[15]

Examiner comment

- Candidates were given a scenario for which they had to write a program. This was new for 2023 and replaced the pre-release scenarios that were given in the old specification as the programming exercise for this qualification.
- Candidates were required to write their response using pseudocode or one of the three programming languages named in the syllabus: Java, Python or VB.NET (Visual Basic). The example response given above is written in Cambridge IGCSE Pseudocode and is very likely to be different from all the candidates' responses. It is an example of a correct solution.
- Most candidates answered using either pseudocode or Python, but a few responses used Java and VB.
- Candidate responses were marked using a Level of Response grid, which was also a new concept for this qualification. For this question there are two response grids:
  - AO2: apply knowledge and understanding of the principles and concepts of computer science to a given context, including analysis and design of computational or programming problems. Maximum 9 marks.
  - AO3: provide solutions to problems by evaluating computer systems, making reasoned judgements and presenting conclusions. Maximum 6 marks.
- Each grid has a lower, middle- and upper-mark band, with a range of marks:

	Lower	**Middle**	**Upper**
AO2	1-3	4-6	7-9
AO3	1-2	3-4	5-6

  In each band, 0 marks can be awarded if no creditable response is seen.

- The solution to be solved in this question was summarised on the mark scheme as three requirements, labelled R1, R2 and R3:

  R1 Check account number and password (iteration and validation, selection, input, output)

  R2 Display menu and make a selection (output, input and selection)

  R3 Perform actions selected (use of arrays and procedures with parameters)

- Candidate responses were compared to these requirements and the scripts were annotated for R1, R2 and R3 individually as 'SEEN' if all of that requirement was present, 'NE' if it had been attempted and 'X' if there was no attempt at that requirement.
- A mark was awarded for each of AO2 and AO3, using a best fit approach from what was seen in the student's response against the text written in the response grids. The total mark (AO2 + AO3) was then recorded as a mark out of 15.