Cambridge IGCSE™

CANDIDATE NAME

CENTRE NUMBER

CANDIDATE NUMBER

COMPUTER SCIENCE **0478/22**

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.

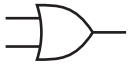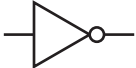This document has **16** pages. Any blank pages are indicated.

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

A Analysis ☐

B Coding ☐

C Design ☐

D 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.

Logic gate | **Standard symbol**

AND

OR

NAND

NOT

[4]

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

1 ....

....

2 ....

....

3 ....

....

[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.

....

....

....

....

....

.... [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.

....

.... [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.

....

....

....

....

....

....

....

.... [3]

5 Circle **five** file-handling operations.

calculate   close   count   create   input   open

output   print   read   sort   search

test   total   write

[5]

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 ....

....

Example ....

....

Feature 2 ....

....

Example ....

....

Feature 3 ....

....

Example ....

....

[6]

5

BLANK PAGE

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 ....

Count-controlled loop ....

Post-condition loop ....
[3]

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

Error 1 ....

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 ....

Pseudocode ....

....

....

....

....

....

[4]

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	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]

9 This flowchart represents an algorithm.

```
                        ┌─────────────┐
                        │    START    │
                        └─────────────┘
                               │
                               ▼
                        ┌─────────────┐
                        │   F ← 0     │
                        │   C ← 1     │
                        └─────────────┘
                               │
                               ▼
                         ╱─────────────╲                    ┌──────────────────┐
                        ╱  IS X[C] <    ╲      No            │    T ← X[C]      │
                        ╲  X[C + 1]?    ╱ ──────────────────▶│                  │
                         ╲─────────────╱                     └──────────────────┘
                               │                                      │
                             Yes                                      ▼
                               │                             ┌──────────────────┐
                               │                             │ X[C] ← X[C + 1]  │
                               │                             └──────────────────┘
                               │                                      │
                               │                                      ▼
                               │                             ┌──────────────────┐
                               │                             │ X[C + 1] ← T     │
                               │                             └──────────────────┘
                               │                                      │
                               │                                      ▼
                               │                             ┌──────────────────┐
                               │◀────────────────────────────│    F ← 1         │
                               │                             └──────────────────┘
                               ▼
                        ┌─────────────┐
                        │  C ← C + 1  │
                        └─────────────┘
                               │
                               ▼
                         ╱─────────────╲
                No      ╱               ╲
          ◀────────────╲   IS C = 5?    ╱
                        ╲               ╱
                         ╲─────────────╱
                               │
                             Yes
                               ▼
                         ╱─────────────╲
                No      ╱               ╲
          ◀────────────╲   IS F = 0?    ╱
                        ╲               ╱
                         ╲─────────────╱
                               │
                             Yes
                               ▼
                        ┌─────────────┐
                        │    STOP     │
                        └─────────────┘
```
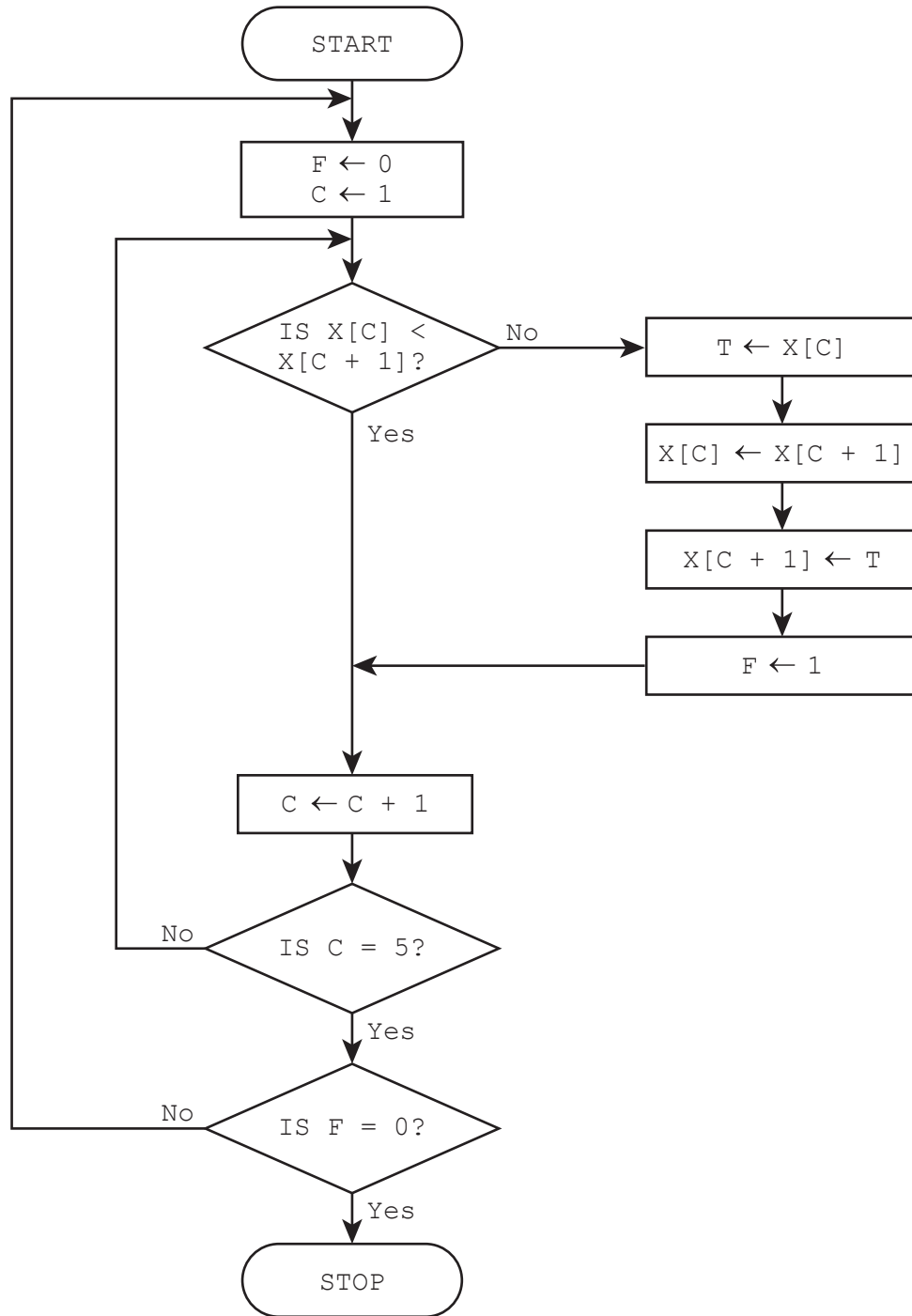
(a) The array `X[1:5]` used in the flowchart contains this data:

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.

....

....

....

.... [2]

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.

.... [1]

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

Field	Data type
SongNumber	
Title	
Recorded	
Minutes	

[2]

(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";
```

....

....

....

....

....

.... [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

....

....

....

.... [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

....

....

....

....

....

....

....

....

....

.... [4]

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

....

.... [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.

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`

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

 ....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [15]

BLANK PAGE

BLANK PAGE

Cambridge IGCSE™

COMPUTER SCIENCE   **0478/22**

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

MARK SCHEME

Maximum Mark: 50

Published

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the May/June 2023 series for most Cambridge IGCSE, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

         [Turn over

Generic Marking Principles

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1: Marks must be awarded in line with: • the specific content of the mark scheme or the generic level descriptors for the question • the specific skills defined in the mark scheme or in the generic level descriptors for the question • the standard of response required by a candidate as exemplified by the standardisation scripts.
GENERIC MARKING PRINCIPLE 2: Marks awarded are always **whole marks** (not half marks, or other fractions).
GENERIC MARKING PRINCIPLE 3: Marks must be awarded **positively**: • marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate • marks are awarded when candidates clearly demonstrate what they know and can do • marks are not deducted for errors • marks are not deducted for omissions • answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.
GENERIC MARKING PRINCIPLE 4: Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5:

Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

GENERIC MARKING PRINCIPLE 6:

Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Please note the following further points:

The words in **bold** in the mark scheme are important text that needs to be present, or some notion of it needs to be present. It does not have to be the exact word, but something close to the meaning.

If a word is underlined, this **exact** word must be present.

A single forward slash means this is an alternative word. A double forward slash means that this is an alternative mark point.

Ellipsis (…) on the end of one-mark point and the start of the next means that the candidate **cannot** get the second mark point without being awarded the first one. If a mark point has an ellipsis at the beginning, but there is no ellipsis on the mark point before it, then this is just a follow-on sentence and **can** be awarded **without** the previous mark point.

Question	Answer	Marks
1	**A**	**1**

Question	Answer	Marks
2	**One** mark for each correct line 	**4**

Question	Answer	Marks
3	**One** mark for each correct answer • structure diagram / chart • flowchart • pseudocode	**3**

Question	Answer	Marks
4(a)	**One** mark for each point (**max three**). • range check with acceptable values is (greater than) zero **and** less than 1000 • presence check to ensure the program will not continue until a value has been entered • type/character check to ensure that a number is entered • length check to ensure there are no more than 3 digits entered	**3**
4(b)(i)	To verify the data / for verification / as a verification check // to make sure that no changes are made to the data on entry	**1**

4(b)(ii)	**One** mark for each point (**max three**).
	• use of iteration
	• use of two inputs
	• to check that the two inputs are the same / different
	• use of the given variable `Measurement`
	
	For example
	<pre>REPEAT
	    OUTPUT "Please enter measurement "
	    INPUT Measurement
	    OUTPUT "Please re-enter measurement "
	    INPUT MeasurementCheck
	UNTIL Measurement = MeasurementCheck</pre>
		3

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

Question	Answer	Marks
6	**One mark** for each feature and **one mark** for corresponding example (**max six**) • ensuring that all identifiers have meaningful names … • … example using `Total` to store a running total • using comments to explain how the program works … • … example `// all values are zeroed before the next calculation` • using procedures and functions for the tasks within a program … • … example `CalculateInterest(Deposit, Rate)`	**6**

Question	Answer	Marks
7(a)	• `07` • `04/12` or `16/18` • `02/20`	**3**
7(b)	**One** mark for each error identified and correction • Line `07 Total ← Total + Number * Counter` should be `Total ← Total + Number[Counter] * Counter` • Line `08 IF Number[Counter] = 0` should be `IF Number[Counter] = -1` **//** should be `IF Number[Counter] < 0` • Line `16 FOR Counter ← 0 TO 5` should be `FOR Counter ← 1 TO 5`	**3**

Question	Answer	Marks

7(c)	**One** mark for place in algorithm (**max one**)
	• around lines 05 and 06
	• line 07
	• (immediately) after the input of the number
	
	Three marks pseudocode
	One mark for each point (**max three**)
	• Use of REPEAT … UNTIL // any working loop structure
	• check for >0 // >=0
	• check for <10 // >9
	• check for whole number
	• check for -1
	• check for length of digit <> 1
	
	Example
	`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

Question	Answer	Marks
8	4 marks for 8 correct outputs 3 marks for 6/7 correct outputs 2 marks for 4/5 correct outputs 1 mark for 2/3 correct outputs	**4**

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

Question	Answer	Marks
9(a)	**One** mark for each column `F`, `C` and `T` **Two** marks for columns `X[1]` to `X[5]` all entries correct or **One** mark for columns `X[1]` to `X[5]` with one error	5

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						

Question	Answer	Marks
9(b)	**One** mark for each point • (bubble) sort data in array • in ascending order	2

Question	Answer	Marks
10(a)	`SongNumber`	**1**
10(b)	**One** mark for every **two** correct data types <table><tr><th>Field</th><th>Data Type</th></tr><tr><td>`SongNumber`</td><td>Text/Alphanumeric</td></tr><tr><td>`Title`</td><td>Text/Alphanumeric</td></tr><tr><td>`Recorded`</td><td>Date/time</td></tr><tr><td>`Minutes`</td><td>Real</td></tr></table>	**2**
10(c)	**One** mark for each point • to find the total number of minutes of music • to find the total number of songs • available for the genre rock	**3**

Question	Answer	Marks

11(a)	**One** mark for any two correct lines
	
	`DECLARE P : STRING`
	`P ← "The world"`
	`DECLARE Q : CHAR`
	`Q ← 'W'`
		2

Question	Answer	Marks

11(b)	**One** mark for each point (**max four**)
	• converting `P` to upper case
	• finding the length of `P`
	• using a loop to check for position of `Q`
	• using the string operation substring
	• storing the loop counter in `Position` if the value is found
	
	For example:
	`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

11(c)	5	1

Question	Answer	Marks

12	Read the whole answer:
	Check if each requirement listed below has been met. Requirements may be met using a suitable built-in function from the programming language used (Python, VB.NET or Java)
	Mark SEEN on script if requirement met, cross if no attempt seen, NE if partially met (see marked scripts).
	Use the tables for A02 and A03 below to award a mark in a suitable band using a best fit approach
	Then add up the total.
	Marks are available for:
	• AO2 (maximum 9 marks)
	• AO3 (maximum 6 marks)
	
	Data Structures required names shown underlined must be used as given in the scenario
	Arrays or lists `Account`, `AccDetails`
	Variable `Size`, `AccountNumber`
	
	Requirements (techniques)
	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)
	
	Example 15 mark answer in pseudocode
	<pre>// Procedures to be called
	PROCEDURE CheckDetails(AccID : INTEGER)
	    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</pre>
		15

Question	Answer	Marks
12	(see code below)	

```
            Valid ← True
        ENDIF
    ENDIF
ENDPROCEDURE

PROCEDURE Balance(AccID : INTEGER)
    OUTPUT "Your balance is ", AccDetails[AccID,1]
ENDPROCEDURE

PROCEDURE WithDrawal(AccID : INTEGER)
    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
        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 Deposit(AccID : INTEGER)
    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
```

Question	Answer	Marks

12	<pre>ENDPROCEDURE
	
	// Declarations of global variables for information – not required in candidate responses
	DECLARE AccountNumber, Choice : INTEGER
	DECLARE Valid, Exit : BOOLEAN
	
	OUTPUT "Please enter your account number "
	INPUT AccountNumber
	CheckDetails(AccountNumber)
	
	IF Valid
	  THEN
	    REPEAT
	        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
	        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</pre>	

Marking Instructions in italics			
AO2: Apply knowledge and understanding of the principles and concepts of computer science to a given context, including the analysis and design of computational or programming problems			
0	**1–3**	**4–6**	**7–9**
No creditable response.	At least one programming technique has been used. *Any use of selection, iteration, counting, totalling, input and output.*	Some programming techniques used are appropriate to the problem. *More than one technique seen applied to the scenario, check the list of techniques needed.*	The range of programming techniques used is appropriate to the problem. *All criteria stated for the scenario have been covered by the use of appropriate programming techniques, check the list of techniques needed.*
	Some data has been stored but not appropriately. *Any **use** of variables or arrays or other language dependent data structures e.g. Python lists.*	Some of the data structures chosen are appropriate and store some of the data required. *More than one data structure **used** to store data required by the scenario.*	The data structures chosen are appropriate and store all the data required. *The data structures **used** store all the data required by the scenario.*

Marking Instructions in italics			
AO3: Provide solutions to problems by: • **evaluating computer systems** • **making reasoned judgements** • **presenting conclusions**			
0	**1–2**	**3–4**	**5–6**
No creditable response.	Program seen without relevant comments.	Program seen with some relevant comment(s).	The program has been fully commented.
	Some identifier names used are appropriate. *Some of the data structures used have meaningful names.*	The majority of identifiers used are appropriately named. *Most of the data structures used have meaningful names.*	Suitable identifiers with names meaningful to their purpose have been used throughout. *All of the data structures used have meaningful names.*
	The solution is illogical.	The solution contains parts that may be illogical.	The program is in a logical order.
	The solution is inaccurate in many places. *Solution contains few lines of code with errors that attempt to perform a task given in the scenario.*	The solution contains parts that are inaccurate. *Solution contains lines of code with some errors that logically perform tasks given in the scenario. Ignore minor syntax errors.*	The solution is accurate. *Solution logically performs all the tasks given in the scenario. Ignore minor syntax errors.*
	The solution attempts at least one of the requirements. *Solution contains lines of code that attempt at least one task given in the scenario.*	The solution meets most of the requirements. *Solution contains lines of code that perform most tasks given in the scenario.*	The solution meets all the requirements given in the question. *Solution performs all the tasks given in the scenario.*