Cambridge Assessment International Education

Cambridge IGCSE™

CANDIDATE NAME			
CENTRE NUMBER		CANDIDATE NUMBER	

* 4 8 1 6 0 2 0 7 0 1 *

COMPUTER SCIENCE **0478/21**

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 (a) **Four** descriptions of stages in the program development life cycle are shown.

Draw **one** line to link each description to its most appropriate program development life cycle stage.

Not all program development life cycle stages will be used.

Program development life cycle description

develop an algorithm to solve the problem by using structure diagrams, flowcharts or pseudocode

detect and fix the errors in the program

identify the problem and its requirements

write and implement the instructions to solve the problem

Program development life cycle stage

analysis

coding

design

evaluation

testing

[4]

(b) Identify **three** of the component parts after a problem has been decomposed.

1 ....

....

2 ....

....

3 ....

....

[3]

2 Tick (✓) **one** box to show the name of the data structure used to store a collection of data of the same data type.

A Array □

B Constant □

C Function □

D Variable □

[1]

3 **(a)** Describe what is meant by data validation.

....

....

....

.... [2]

(b) A validation check is used to make sure that any value that is input is an integer between 30 and 200 inclusive.

Give **one** example of each type of test data to check that the validation check is working as intended. Each example of test data must be different.

Give a reason for each of your choices of test data.

Normal test data ....

Reason ....

....

Abnormal test data ....

Reason ....

....

Extreme test data ....

Reason ....

....

[6]

4 Explain the purpose of the library routines `DIV` and `ROUND`

`DIV` ....

....

....

....

`ROUND` ....

....

....

....

[4]

5 An algorithm has been written in pseudocode to allow some numbers to be input. All the positive numbers that are input are totalled and this total is output at the end.
An input of `0` stops the algorithm.

```
Exit ← 1
WHILE Exit <> 0 DO
    INPUT Number
    IF Number < 0
      THEN
        Total ← Total + Number
      ELSE
        IF Number = 0
          THEN
            Exit ← 1
        ENDIF
    ENDIF
ENDIF
OUTPUT "The total value of your numbers is ", Number
```

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

Error 1 ....

Correction ....

....

Error 2 ....

Correction ....

....

Error 3 ....

Correction ....

....

Error 4 ....

Correction ....

....

[4]

(b) Describe how you could change the corrected algorithm to record and output how many positive numbers have been included in the final total.

You do **not** need to rewrite the algorithm.

....

....

....

....

....

....

....

.... [4]

6 State **two** features that should be included to create a maintainable program.

Give a reason why each feature should be used.

1 ....

....

....

....

2 ....

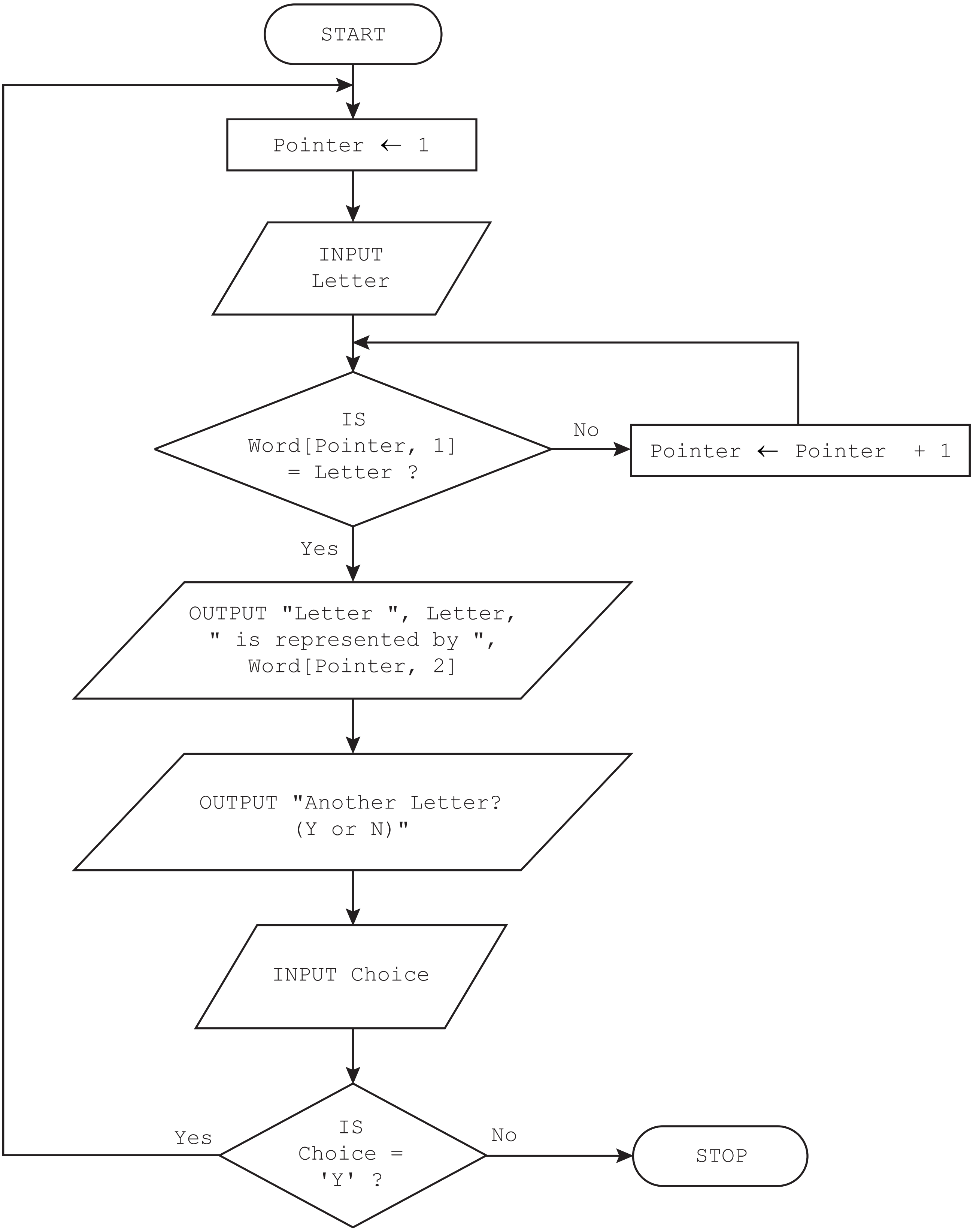
....

....

....

[4]

7 The flowchart represents an algorithm.


START
Pointer ← 1
INPUT Letter
IS Word[Pointer, 1] = Letter ?
No
Pointer ← Pointer + 1
Yes
OUTPUT "Letter ", Letter, " is represented by ", Word[Pointer, 2]
OUTPUT "Another Letter? (Y or N)"
INPUT Choice
IS Choice = 'Y' ?
Yes
No
STOP

The table represents the two-dimensional (2D) array `Word[]` which stores the first half of the phonetic alphabet used for radio transmission. For example, `Word[10,1]` is 'J'.

Index	**1**	**2**
1	A	Alpha
2	B	Bravo
3	C	Charlie
4	D	Delta
5	E	Echo
6	F	Foxtrot
7	G	Golf
8	H	Hotel
9	I	India
10	J	Juliet
11	K	Kilo
12	L	Lima
13	M	Mike

(a) Complete the trace table for the algorithm by using the input data: F, Y, D, N

`Pointer`	`Letter`	`Choice`	**OUTPUT**

[4]

(b) Identify the type of algorithm used.

....

.... [1]

(c) Describe **one** problem that could occur with this algorithm if an invalid character was input.

....

....

....

.... [2]

BLANK PAGE

[Turn over

8 The function `LENGTH(Phrase)` calculates the length of a string `Phrase`

(a) Write the pseudocode statements to:

- store the string `"The beginning is the most important part"` in `Phrase`
- calculate and output the length of the string
- output the string in upper case.

....

....

....

....

....

.... [3]

(b) Write the output your pseudocode should produce.

....

....

....

.... [2]

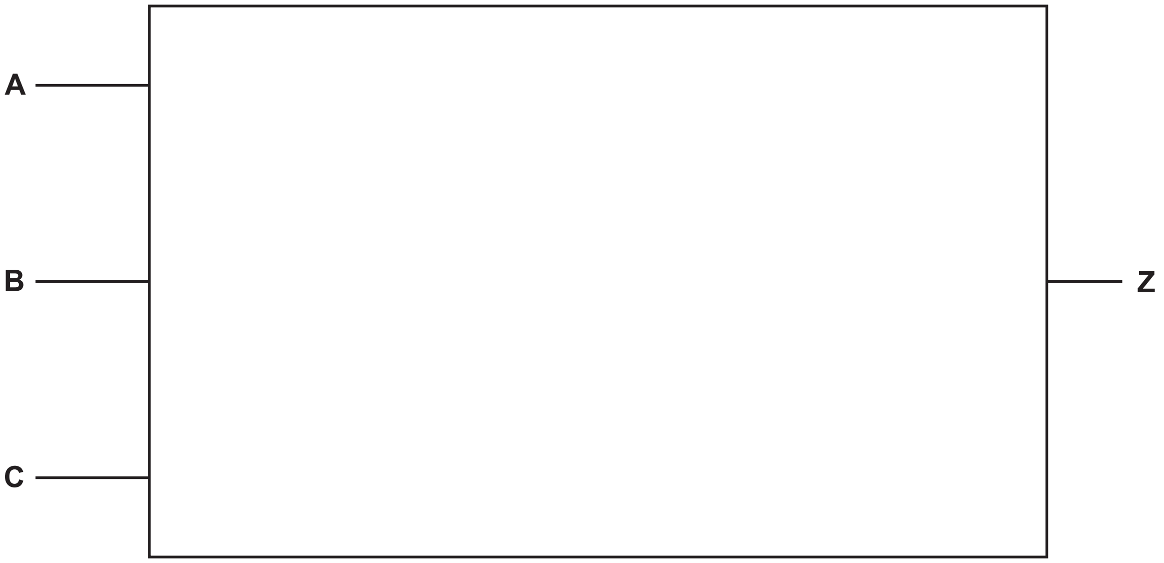
9 Consider this logic expression.

Z = (NOT **A** OR **B**) AND (**B** XOR **C**)

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

Each logic gate must have a maximum of **two** inputs.

Do **not** simplify this logic expression.



[4]

(b) Complete the truth table from the given logic expression.

A	**B**	**C**	**Working space**	**Z**
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]

10 A database table called `TVRange` shows the main features and prices of a range of televisions.

TVCode	**ScreenSize**	**Satellite**	**SmartTV**	**SoundBar**	**Price$**
TV90SaSmSd	90	YES	YES	YES	9750.00
TV75SaSmSd	75	YES	YES	YES	8500.00
TV75SaSd	75	YES	NO	YES	8000.00
TV65SaSmSd	65	YES	YES	YES	6000.00
TV65SmSd	65	NO	YES	YES	5000.00
TV65SaSd	65	YES	NO	YES	5000.00
TV55SaSmSd	55	YES	YES	YES	4000.00
TV55SaSd	55	YES	NO	YES	3500.00
TV55SmSd	55	NO	YES	YES	3500.00
TV50SaSmSd	50	YES	YES	YES	2500.00
TV50Sa	50	YES	NO	NO	1750.00
TV50Sm	50	NO	YES	NO	1750.00
TV40Sa	40	YES	NO	NO	1200.00
TV40	40	NO	NO	NO	950.00
TV32	32	NO	NO	NO	650.00

(a) Give the name of the field that is most suitable to be the primary key.

State the reason for this choice.

Field ....

Reason ....

.... [2]

(b) The database uses the data types:

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

Complete the table to show the most appropriate data type for each field.
Each data type must be different.

Field	**Data type**
`TVCode`	
`ScreenSize`	
`SmartTV`	
`Price$`	

[2]

(c) Complete the structured query language (SQL) query to return the television (TV) code, screen size and price of all Smart TVs in the database table.

```
SELECT TVCode, .................................................., ..................................................

.................................................. TVRange

WHERE SmartTV = ..................................................;
```

[4]

11 A one-dimensional (1D) array `Days[]` contains the names of the days of the week. A two-dimensional (2D) array `Readings[]` is used to store 24 temperature readings, taken once an hour, for each of the seven days of the week. A 1D array `AverageTemp[]` is used to store the average temperature for each day of the week.

The position of any day's data is the same in all three arrays. For example, if Wednesday is in index 4 of `Days[]`, Wednesday's temperature readings are in index 4 of `Readings[]` and Wednesday's average temperature is in index 4 of `AverageTemp[]`

The temperature readings are in Celsius to one decimal place. Temperatures can only be from –20.0 °C to +50.0 °C inclusive.

Write a program that meets the following requirements:

- input and validate the hourly temperatures for one week
- calculate and store the average temperature for each day of the week
- calculate the average temperature for the whole week
- convert all the average temperatures from Celsius to Fahrenheit by using the formula Fahrenheit = Celsius * 9 / 5 + 32
- output the average temperature in Celsius and in Fahrenheit for each day
- output the overall average temperature in Celsius and in Fahrenheit for the whole week.

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

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

All inputs and outputs must contain suitable messages.

All data output must be rounded to one decimal place.

You will need to initialise and populate the array `Days[]` at the start of the program.

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [15]

Cambridge Assessment International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/21**

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.

This document consists of **18** printed pages.

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:

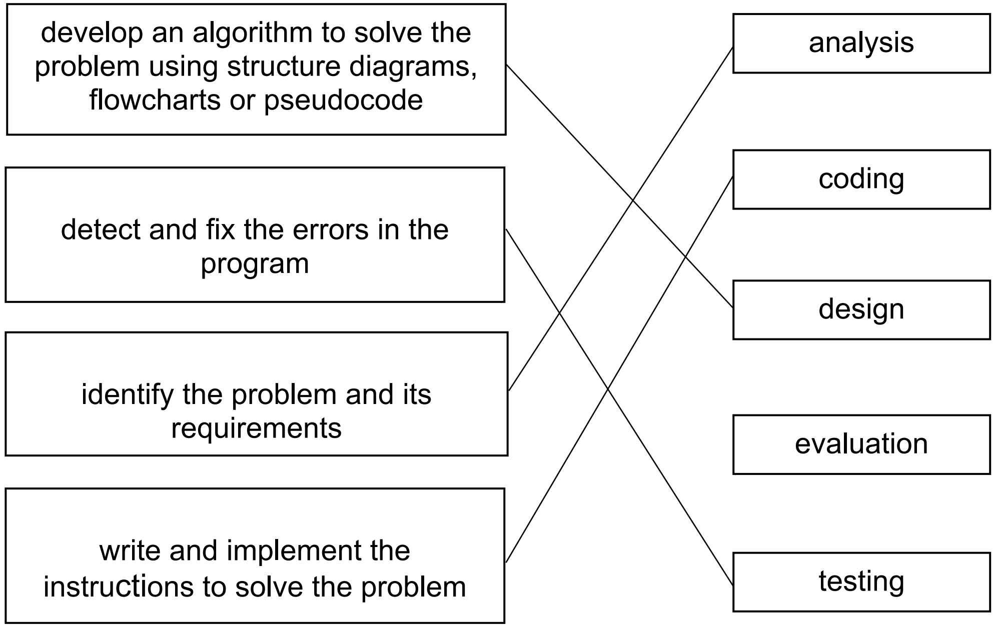
Marks must be awarded **positively**:

- 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.

Question	Answer	Marks
1(a)	**One** mark for each correct line. **Program development life cycle description** **Program development life cycle stage** develop an algorithm to solve the problem using structure diagrams, flowcharts or pseudocode detect and fix the errors in the program identify the problem and its requirements write and implement the instructions to solve the problem analysis coding design evaluation testing	**4**
1(b)	**One** mark for naming or describing each component part, max **three** **For example:** inputs // what is put into the system processes // actions taken to achieve a result outputs // what is taken out of the system storage // what needs to be kept for future use	**3**

Question	Answer	Marks
2	A	**1**

Question	Answer	Marks
3(a)	**One** mark per mark point, max **two** • Validation is an automated check carried out by a computer • … to make sure the data entered is sensible/acceptable/reasonable	**2**
3(b)	**One** mark for each appropriate test data, max **three** **One** mark for each correct accompanying reason, max **three** **For example:** Normal – 75 Reason – the data lies within the required range **and** should be accepted Abnormal – Sixty Reason – this is the wrong data type **and** should be rejected Extreme – 200 Reason – the highest value in the required range that should be accepted	**6**

Question	Answer	Marks
4	**One** mark per mark point, max **four** `DIV`, max **two** • To perform integer division • Meaning only the whole number part of the answer is retained • Example of `DIV` For example `DIV(9,4) = 2` `ROUND`, max **two** • To return a value rounded to a specified number of digits / decimal places • The result will either be rounded to the next highest or the next lowest value • … depending on whether the value of the preceding digit is >=5 or <5 • Example of `ROUND` for example, `ROUND(4.56, 1) = 4.6`	**4**

Question	Answer	Marks
5(a)	**One** mark per mark point, max **four**	**4**

- Line 04 / `IF Number < 0`
  should be `IF Number > 0`

- Line 10 / `Exit ← 1` // Line 01/ `Exit ← 1` **and** Line 02 / `WHILE Exit <> 0`
  should be `Exit ← 0` // should be `Exit ← 0` **and** `WHILE Exit = 0`

- Line 13 / `ENDIF`
  should be `ENDWHILE`

- Line 14 / `OUTPUT "The total value of your numbers is ", Number`
  should be `OUTPUT "The total value of your numbers is ", Total`

Correct algorithm:

```
01 Exit ← 1
02 WHILE Exit <> 0 DO
03   INPUT Number
04   IF Number > 0
05    THEN
06     Total ← Total + Number
07    ELSE
08     IF Number = 0
09      THEN
10       Exit ← 0
11     ENDIF
12   ENDIF
13 ENDWHILE
14 OUTPUT "The total value of your numbers is ", Total
```

Question	Answer	Marks
5(b)	**One** mark per mark point, max **four** • Initialise a new (counting) variable • … `Count ← 0` // to count the acceptable numbers • Insert a counting statement between lines 05 and 07 • … `Count ← Count + 1` • Add a new output after the loop/after line 13 / at the end (of the program) • … `OUTPUT Count`	**4**

Question	Answer	Marks
6	**One** mark for each correct feature, max **two** **One** mark for each correct accompanying reason, max **two** **For example:** Meaningful identifiers – to enable the programmer (or future programmers) to easily recognize the purpose of a variable / array / constant // to enable easy tracking of a variable / constant / array through the program Use of comments – to annotate each section of a program so that a programmer can find specific sections / so that the programmer knows the purpose of that section of code Procedures and functions – to make programs modular and easier to update / add functionality	**4**

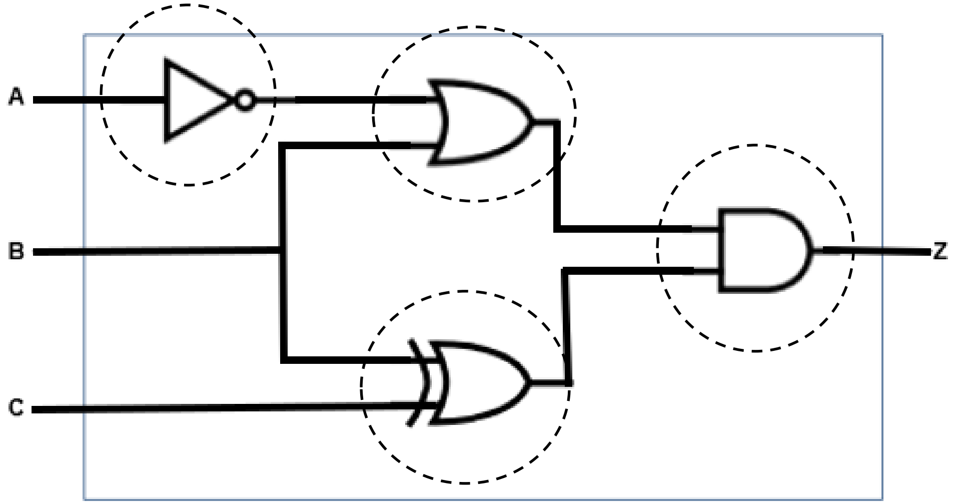
<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td><b>One</b> mark per correct column, max <b>four</b>
<table>
<tr><th><code>Pointer</code></th><th><code>Letter</code></th><th><code>Choice</code></th><th>OUTPUT</th></tr>
<tr><td>1</td><td>F</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td></tr>
<tr><td>6</td><td></td><td></td><td>Letter F is represented by Foxtrot</td></tr>
<tr><td></td><td></td><td></td><td>Another Letter? (Y or N)</td></tr>
<tr><td></td><td></td><td>Y</td><td></td></tr>
<tr><td>1</td><td>D</td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td>Letter D is represented by Delta</td></tr>
<tr><td></td><td></td><td></td><td>Another Letter? (Y or N)</td></tr>
<tr><td></td><td></td><td>N</td><td></td></tr>
</table>
</td><td><b>4</b></td></tr>
<tr><td>7(b)</td><td>(Linear) search</td><td><b>1</b></td></tr>
</table>

Question	Answer	Marks
7(c)	**One** mark per mark point, max **two** • The algorithm would not stop • … because it would not have found the item it was seeking **Or** • The array would run out of values after the pointer reached 13 • the algorithm will crash	**2**

Question	Answer	Marks

8(a)	**One** mark per mark point, max **three**
	• Storing string in `Phrase`
	• Correct use of `LENGTH` function
	• Correct use of `UCASE` function
	• Correct outputs of `LENGTH` and `UCASE`
	For example:
	`Phrase ← "The beginning is the most important part"`
	`OUTPUT LENGTH(Phrase)`
	`OUTPUT UCASE(Phrase)`
		3

8(b)	**One** mark for each correct line, max **two**
	`40`
	`THE BEGINNING IS THE MOST IMPORTANT PART`
		2

Question	Answer	Marks
9(a)	**One** mark for each correct gate, with the correct input(s) as shown.  	**4**

Question	Answer	Marks
9(b)	**Four** marks for eight correct outputs. **Three** marks for six or seven correct outputs. **Two** marks for four or five correct outputs. **One** mark for two or three correct outputs	**4**

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Question	Answer	Marks
10(a)	**One** mark for the correct field name **One** mark for the correct reason **For example:** `TVCode` Each entry in this field is a unique identifier	**2**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>10(b)</td><td><b>Two</b> marks for four correct answers.
<b>One</b> mark for two or three correct answers.
<table><tr><th>Field</th><th>Data type</th></tr><tr><td><code>TVCode</code></td><td>Text</td></tr><tr><td><code>ScreenSize</code></td><td>Integer</td></tr><tr><td><code>SmartTV</code></td><td>Boolean</td></tr><tr><td><code>Price$</code></td><td>Real</td></tr></table></td><td><b>2</b></td></tr>
<tr><td>10(c)</td><td><b>One</b> mark for each correct answer

<code>ScreenSize</code>
<code>Price$</code>
<code>FROM</code>
<code>YES</code>

<b>Correct code:</b>

<code>SELECT TVCode, ScreenSize, Price$</code>
<code>FROM TVRange</code>
<code>WHERE SmartTV = YES;</code></td><td><b>4</b></td></tr>
</table>

Question	Answer	Marks
11	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). On place a SEEN mark if requirement met, cross if no attempt seen, omission mark and/or comment if partially met (see marked scripts). Use the tables for AO2 and AO3 below to award a mark in a suitable band using a best fit approach, then add up the total: • AO2 (maximum 9 marks) • AO3 (maximum 6 marks) **Data structures required:** The names underlined must match those given in the scenario: Arrays or lists `Days[]`, `Readings[]`, `AverageTemp[]` Variables `WeekLoop`, `DayLoop`, `InTemp`, `TotalDayTemp`, `TotalWeekTemp`, `AverageWeekTemp` **Requirements (techniques):** **R1** Input and store hourly temperatures and validation of input temperatures for each day (with prompts, range check and (nested)iteration) **R2** Calculate, round to one decimal place and store daily average temperatures and calculate the weekly average temperature rounded to one decimal place (iteration, totalling and rounding) **R3** Convert all average temperatures to Fahrenheit (to one decimal place) and output the average temperatures in both Celsius and Fahrenheit. Output with appropriate messages. (output and rounding)	**15**

Question	Answer	Marks
11	**Example 15 mark answer in pseudocode**	

```
// meaningful identifiers and appropriate data structures for
// all data required
DECLARE Days : ARRAY[1:7] OF STRING
DECLARE Readings : ARRAY[1:7, 1:24] OF REAL
DECLARE AverageTemp : ARRAY[1:7] OF REAL
DECLARE WeekLoop : INTEGER
DECLARE DayLoop : INTEGER
DECLARE InTemp : REAL
DECLARE TotalDayTemp : REAL
DECLARE TotalWeekTemp : REAL
DECLARE AverageWeekTemp : REAL
// initial population of Days[] array
// input and a loop are also acceptable
Days[1] ← "Sunday"
Days[2] ← "Monday"
Days[3] ← "Tuesday"
Days[4] ← "Wednesday"
Days[5] ← "Thursday"
Days[6] ← "Friday"
Days[7] ← "Saturday"
// input temperatures inside nested loop
FOR WeekLoop ← 1 TO 7
    TotalDayTemp ← 0
    FOR DayLoop ← 1 TO 24
        OUTPUT "Enter temperature ", DayLoop, " for ", Days[WeekLoop]
```

Question	Answer	Marks
11		

```
        INPUT InTemp
// validation of input for between -20 and +50 inclusive
        WHILE InTemp < -20.0 OR InTemp > 50.0 DO
            OUTPUT "Your temperature must be between -20.0 and +50.0 inclusive. Please try
                    again"
            INPUT InTemp
        ENDWHILE
        Readings[WeekLoop, DayLoop] ← InTemp
// totalling of temperatures during the day
        TotalDayTemp ← TotalDayTemp + ROUND(InTemp, 1)
    NEXT DayLoop

// average temperature for the day
    AverageTemp[WeekLoop] ← ROUND(TotalDayTemp / 24,1)
NEXT WeekLoop
// calculate the average temperature for the week
TotalWeekTemp ← 0
FOR WeekLoop ← 1 TO 7
    TotalWeekTemp ← TotalWeekTemp + AverageTemp[WeekLoop]
NEXT WeekLoop

AverageWeekTemp ← ROUND(TotalWeekTemp / 7,1)
// outputs in Celsius and Fahrenheit
FOR WeekLoop ← 1 TO 7
    OUTPUT "The average temperature on ", Days[WeekLoop], " was ", AverageTemp[WeekLoop], "
            Celsius and ",
           ROUND(AverageWeekTemp * 9 / 5 + 32), 1, " Fahrenheit"
NEXT WeekLoop

OUTPUT "The average temperature for the week was ",
       AverageWeekTemp," Celsius and ", ROUND(AverageWeekTemp * 9 / 5 + 32, 1),"
Fahrenheit"
```

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.*