Cambridge Assessment International Education

Cambridge IGCSE™

CANDIDATE NAME			
CENTRE NUMBER	□□□□□	CANDIDATE NUMBER	□□□□

5093781865

COMPUTER SCIENCE **0478/23**

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 complete the sentence.

A constant

A stores a value that can change at any time during the execution of a program. □

B stores a value that cannot change during the execution of a program. □

C stores values of multiple data types. □

D stores values that must be of the same data type. □

[1]

2 Explain the purpose of the library routines `MOD` and `RANDOM`

`MOD` ....

....

....

....

`RANDOM` ....

....

....

....

[4]

3 Describe what happens when a function is called during the execution of a program.

....

....

....

....

....

.... [3]

4 **(a)** Explain why verification checks are used when data is input.

....

....

....

.... [2]

(b) Give **two** types of verification check and state how each one can be used.

Verification check 1 ....

Use ....

....

Verification check 2 ....

Use ....

....

[4]

5 (a) **Four** descriptions of validation checks are shown.

Draw **one** line to link each description to the most appropriate check.

Not all checks will be used.

Description

to check that the data entered is an integer

to check that some data has been entered

to check that the data entered has an appropriate number of characters

to check that an identification number contains no errors

Check

check digit

format check

length check

presence check

type check

[4]

(b) Write an algorithm in pseudocode to make sure that an input for the variable `Length` is between 15 and 35 inclusive. The code must iterate until a valid input has been made and the code must include appropriate messages.

....

....

....

....

....

....

....

.... [3]

6 An algorithm has been written in pseudocode to allow 100 positive numbers to be input. The total and the average of the numbers are output.

```
01 Counter ← 100
02 Total ← 0
03 WHILE Counter > 100 DO
04    INPUT Number
05    IF Number > 0
06       THEN
07          Total ← Total + Counter
08          Counter ← Counter + 1
09    ENDCASE
10 ENDWHILE
11 OUTPUT "The total value of your numbers is ", Total
12 OUTPUT "The average value of your numbers is ", Total / 100
```

(a) Identify the **four** errors in the pseudocode and suggest corrections.

Error 1 ....

Correction ....

....

Error 2 ....

Correction ....

....

Error 3 ....

Correction ....

....

Error 4 ....

Correction ....

....
[4]

(b) Describe the changes you should make to the corrected algorithm so that a count-controlled loop is used to allow 100 positive numbers to be input.

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

....

....

....

....

....

....

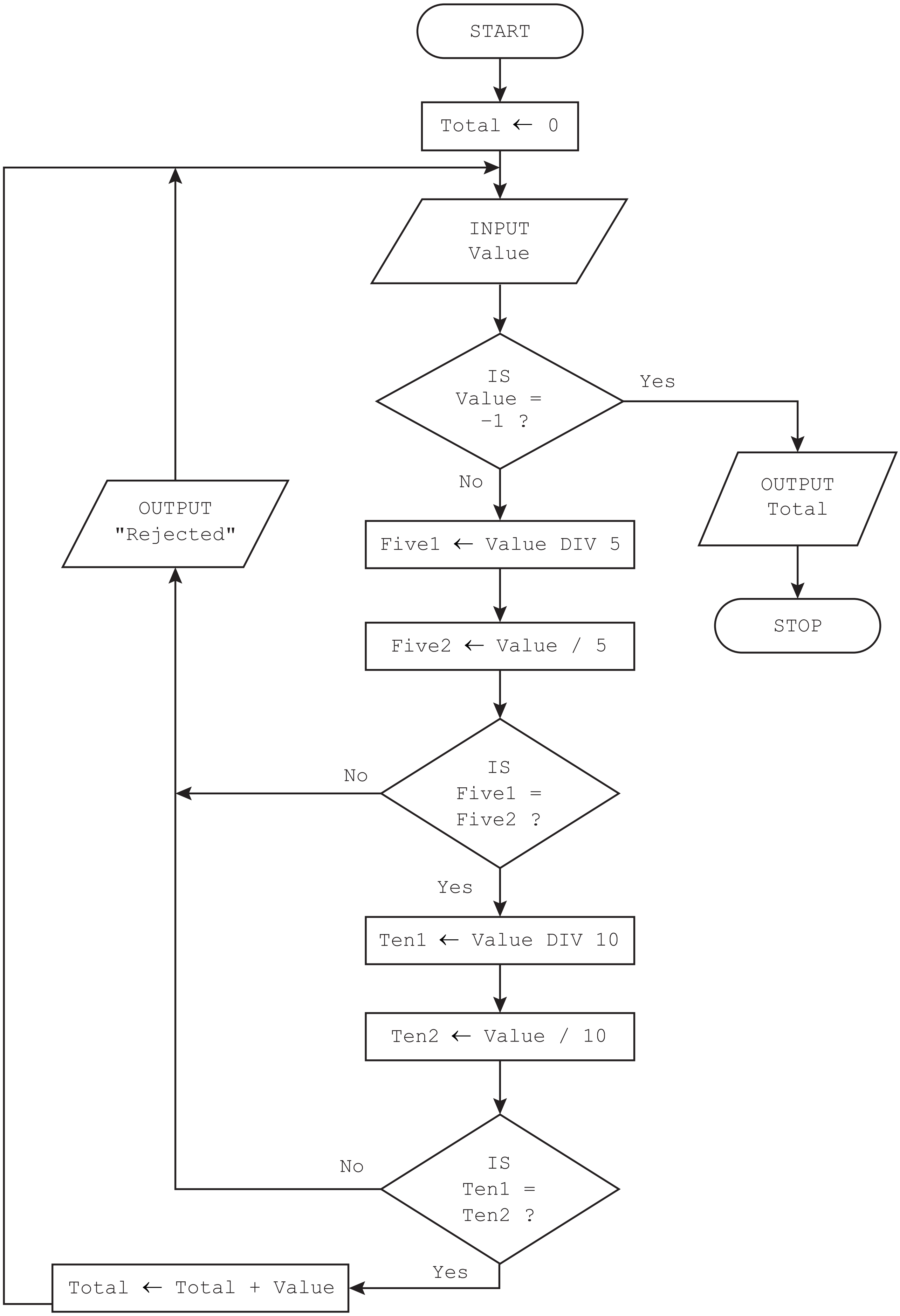
....

....

....

.... [5]

7 The flowchart represents an algorithm.
An input of –1 will terminate the algorithm.



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

5, 50, 52, 555, 57, 500, –1, 5500, 55

`Total`	`Value`	`Five1`	`Five2`	`Ten1`	`Ten2`	OUTPUT

[6]

(b) Describe the purpose of the algorithm.

....

....

....

.... [2]

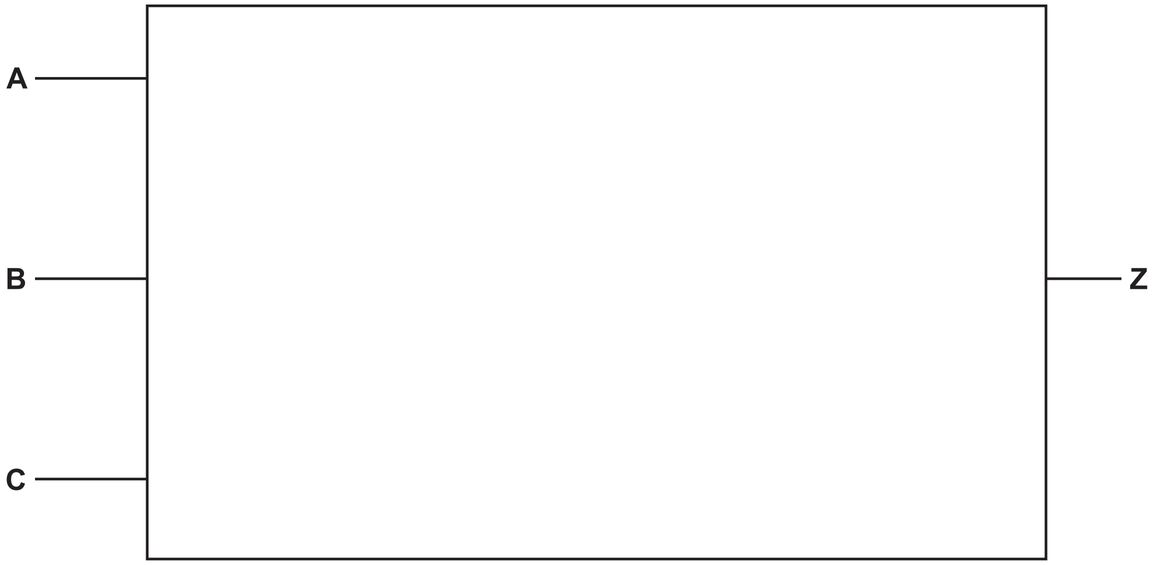
8 Consider the logic expression:

Z is 1 if (**A** = 1 AND **C** = NOT 1) AND (**B** = 1 NOR **C** = 1)

(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]

9 The variable `Saying` is used to store string data in a program.

(a) Write the pseudocode statement to declare the variable `Saying`

....

.... [1]

(b) Write the pseudocode statements to:

- allow a string to be input to the variable `Saying`
- store the content of the variable `Saying` in a text file named `"Quotations.txt"`
- make sure the text file is closed at the end of the algorithm.

....

....

....

....

....

....

....

....

....

.... [5]

10 A database table called `Site1` stores details of some holiday homes at a holiday park. The database shows the type of home, number of guests, whether it is privately owned and the weekly rate to hire it.

Name	**Type**	**Private**	**Rate$**	**NumberGuest**
Bay Lodge	Lodge	NO	1000	10
Bay View	Cabin	NO	400	4
Blue Skies	Cabin	NO	350	4
Cliff View	Cabin	NO	650	6
Coppice Lodge	Lodge	NO	1200	12
Green Lodge	Lodge	NO	1000	8
Henry	Cabin	YES	300	2
Hikers' Rest	Retreat	NO	750	6
Poppy	Cabin	NO	300	2
Summer Joy	Retreat	YES	750	6
Valley View	Cabin	NO	600	6
West Lodge	Lodge	YES	1200	12

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

Fields ....

Records ....

[2]

(b) Describe the purpose of a primary key.

....

.... [1]

(c) The database uses the data types:

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

Complete the table to show the most appropriate data type for each field.

Field	**Data type**
Type	
Private	
Rate$	
NumberGuest	

[2]

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

```
SELECT Name, NumberGuest, Rate$
FROM Site1
WHERE NumberGuest >= 10;
```

....

....

....

....

....

.... [3]

11 A two-dimensional (2D) array `Contacts[]` is used to store names and telephone numbers. All the data is stored as strings. The array must have the capacity to store 100 contacts in the form of:

- column 1 – contact names as: last name, first name
  for example: Smith, John
- column 2 – telephone numbers.

The variable `CurrentSize` shows how many contacts are in the array.

Write a program that meets the following requirements:

- display a menu of choices:
  - ○ enter new contact details
  - ○ display all the contact details
  - ○ delete all the contact details
- validate the menu input
- allow up to a maximum of five new contacts to be added to the array at any one time
- do **not** allow more than 100 contacts in total
- after new contacts have been added, sort the array by contact name, as long as there are at least two contacts in the array
- output the whole of the array
- delete the contents of the array.

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.

You do **not** need to initialise the data in the array `Contacts[]` and the variable `CurrentSize`

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [15]

BLANK PAGE

BLANK PAGE

Cambridge Assessment International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/23**

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 **19** 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.

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.

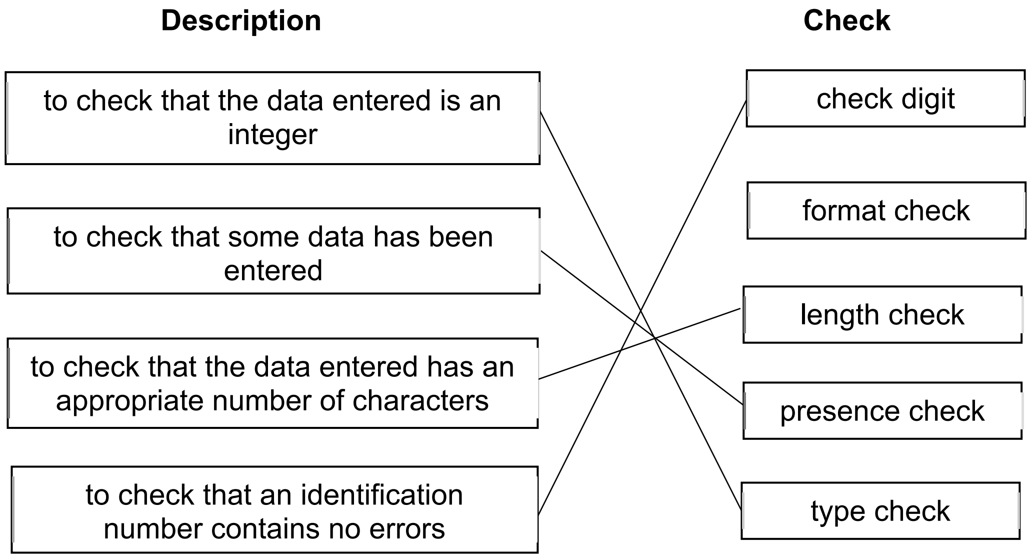
Question	Answer	Marks
1	B	**1**

Question	Answer	Marks
2	**One** mark per mark point, max **four** `MOD`, max **two** • To perform (integer) division when one number is divided by another • … and find the remainder • Allow example e.g. `7 MOD 2 = 1` `RANDOM`, max **two** • To generate (pseudo) random numbers • …(usually) within a specified range • Allow example e.g. `RANDOM() * 10` returns a random number between `0` and `10`	**4**

Question	Answer	Marks
3	**One** mark per mark point, max **three** MP1 A **call statement** is used in order to make use of a function // the function is called **using its identifier** MP2 Parameters are / may be passed (from the main program) to the function (to be used within the function) MP3 The function performs its task … MP4 … and returns a value / values to the main program	**3**

Question	Answer	Marks
4(a)	**One** mark per mark point, max **two** • To ensure that data has been accurately copied // to ensure that changes have not been made to the values originally intended when data is copied • … from one source to another	**2**

Question	Answer	Marks
4(b)	**One** mark for each appropriate verification check, max **two** **One** mark for each correct accompanying use, max **two** **For example:** Verification check 1 – Visual check Use – the user looks through the data that has been entered and confirms that no changes have been made. Verification check 2 – Double data entry Use – data is entered twice, the two entries are compared and if they do not match, a re-entry is requested.	**4**

Question	Answer	Marks
5(a)	**One** mark for each correct line.  **Description** to check that the data entered is an integer to check that some data has been entered to check that the data entered has an appropriate number of characters to check that an identification number contains no errors **Check** check digit format check length check presence check type check	**4**

Question	Answer	Marks

5(b)	**One** mark per mark point, max **three**
	• appropriate REPEAT / WHILE loop begin and end
	• input of Length
	• appropriate input prompt / error message
	• correct loop exit/entry condition / selection
	Example answers:
	WHILE Loop
	`OUTPUT "Enter a number between 15 and 35 inclusive"`
	`INPUT Length`
	`WHILE Length <15 OR Length > 35 (DO)`
	`    OUTPUT "Your number must be between 15 and 35 inclusive`
	`    INPUT Length`
	`ENDWHILE`
	REPEAT Loop
	`REPEAT`
	`    OUTPUT "Enter a number between 15 and 35 inclusive"`
	`    INPUT Length`
	`UNTIL Length >= 15 AND LENGTH <= 35`
		3

Question	Answer	Marks
6(a)	**One** mark per mark point, max **four** • Line 01 / `Counter ← 100` should be `Counter ← 0` • Line 03 / `While Counter > 100 DO` should be `While Counter < 100 DO` • Line 07 / `Total ← Total + Counter` should be `Total ← Total + Number` • Line 09 / `ENDCASE` should be `ENDIF`	**4**

Correct algorithm

```
01 Counter ← 0
02 Total ← 0
03 WHILE Counter < 100 DO
04     INPUT Number
05     IF Number > 0
06       THEN
07         Total ← Total + Number
08         Counter ← Counter + 1
09     ENDIF
10 ENDWHILE
11 OUTPUT "The total value of your numbers is ", Total
12 OUTPUT "The average value of your numbers is ", Total / 100
```

Question	Answer	Marks
6(b)	**One** mark per mark point, max **five** MP1 replace line 03 MP2 with `FOR` MP3 … with limits 0 to 99 / 1 to 100 MP4 replace line 05 to check if `Number` is not positive MP5 … (if `Number` is not positive) insert a validation and re-input routine between lines 06 and 07 … MP6 … that will repeat until a positive value is entered MP7 remove the counter update / line 08 MP8 replace line 10 / `ENDWHILE` with `NEXT`	**5**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td>

One mark per mark point, max **six**

- correct `Total` column
- correct `Value` column
- correct `Five1` column
- correct `Five2` column
- correct `Ten1` and `Ten2` columns
- correct OUTPUT column

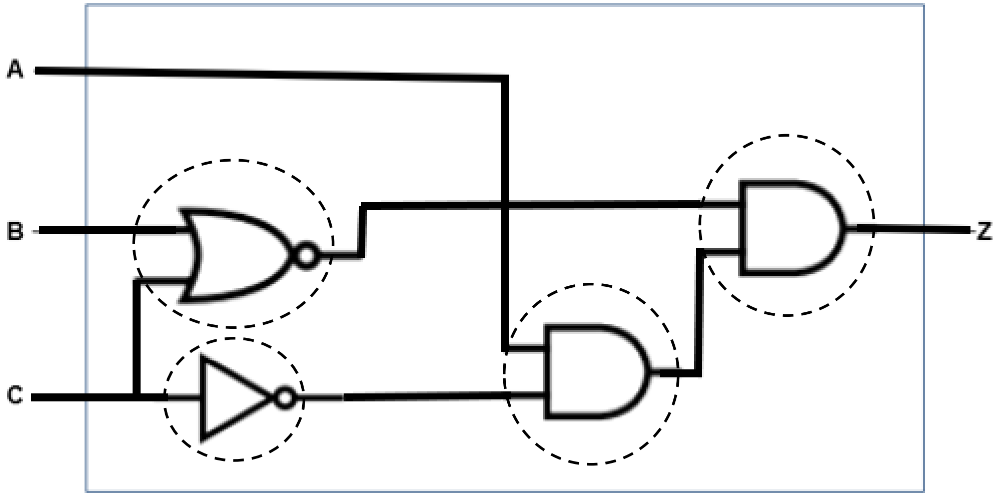
<table>
<tr><th>Total</th><th>Value</th><th>Five1</th><th>Five2</th><th>Ten1</th><th>Ten2</th><th>OUTPUT</th></tr>
<tr><td>0</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>5</td><td>1</td><td>1</td><td>0</td><td>0.5</td><td>Rejected</td></tr>
<tr><td></td><td>50</td><td>10</td><td>10</td><td>5</td><td>5</td><td></td></tr>
<tr><td>50</td><td>52</td><td>10</td><td>10.4</td><td></td><td></td><td>Rejected</td></tr>
<tr><td></td><td>555</td><td>111</td><td>111</td><td>55</td><td>55.5</td><td>Rejected</td></tr>
<tr><td></td><td>57</td><td>11</td><td>11.4</td><td></td><td></td><td>Rejected</td></tr>
<tr><td></td><td>500</td><td>100</td><td>100</td><td>50</td><td>50</td><td></td></tr>
<tr><td>550</td><td>–1</td><td></td><td></td><td></td><td></td><td>550</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

</td><td>6</td></tr>
<tr><td>7(b)</td><td>

One mark per mark point, max **two**

- to find if an input is divisible by (both 5 and) 10
- … add them together **and** output the total

</td><td>2</td></tr>
</table>

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

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>8(b)</td><td><b>Four</b> marks for eight correct outputs.
<b>Three</b> marks for six or seven correct outputs.
<b>Two</b> marks for four or five correct outputs.
<b>One</b> mark for two or three correct outputs
<table>
<tr><th>A</th><th>B</th><th>C</th><th>Z</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td></tr>
</table></td><td>4</td></tr>
</table>

Question	Answer	Marks
9(a)	`DECLARE Saying : STRING`	1

Question	Answer	Marks

9(b)	**One** mark per mark point, max **five**
	MP1 input a string into `Saying`
	MP2 correct use of `OPENFILE` to write data
	MP3 correct use of `WRITEFILE` to write `Saying`
	MP4 correct use of `CLOSEFILE`
	MP5 correct use of filename `Quotations.txt` throughout
	
	For example:
	
	`INPUT Saying`
	`OPENFILE "Quotations.txt" FOR WRITE`
	`WRITEFILE "Quotations.txt", Saying`
	`CLOSEFILE "Quotations.txt"`
		5

Question	Answer	Marks
10(a)	**One** mark for each correct answer Fields 5 Records 12	**2**
10(b)	to **uniquely identify** a record	**1**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>10(c)</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>Type</code></td><td>Alphanumeric</td></tr><tr><td><code>Private</code></td><td>Boolean</td></tr><tr><td><code>Rate$</code></td><td>Integer</td></tr><tr><td><code>NumberGuest</code></td><td>Integer</td></tr></table></td><td><b>2</b></td></tr>
<tr><td>10(d)</td><td><b>One</b> mark per mark point, max <b>three</b>
• data correctly extracted in any two rows
• data correctly extracted in third row
• data in correct order horizontally and vertically

<b>Example answer:</b>

<code>Bay Lodge 10 1000</code>
<code>Coppice Lodge 12 1200</code>
<code>West Lodge 12 1200</code></td><td><b>3</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). 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:** The names underlined must match those given in the scenario: Arrays or lists `Contacts[]` Variables `CurrentSize`, `Cont`, `Choice`, `NewContacts`, `Count`, `Count2`, `Flag` **Requirements (techniques):** **R1** Output menu and input choice, with validation (range check, output with messages, input with prompts). **R2** Input number of new entries, within limits, update current size of contacts, input new data and sort the array (range check, totalling, iteration and bubble sort). **R3** Output array whole contents and delete contents of array (iteration, output with labelling/messages, array initialisation).	**15**

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

```
// meaningful identifiers and appropriate data structures for
// all data required
DECLARE Contacts : ARRAY[1:100, 1:2] OF STRING
DECLARE CurrentSize : INTEGER
DECLARE Cont : BOOLEAN
DECLARE Choice : INTEGER
DECLARE NewContacts : INTEGER
DECLARE Count : INTEGER
DECLARE Count2 : INTEGER
DECLARE Flag : BOOLEAN
DECLARE Temp1 : STRING
DECLARE Temp2 : STRING

// the number of contacts in the array
CurrentSize ← 0

// to allow program to continue indefinitely
Cont ← TRUE
WHILE Cont DO
// display menu
    OUTPUT "Please choose one of the following: "
    OUTPUT "Press 1 to enter new contacts "
    OUTPUT "Press 2 to display your contacts "
    OUTPUT "Press 3 to delete all contacts "
    INPUT Choice
// validate choice as 1, 2 or 3
    WHILE Choice = 1 AND CurrentSize = 100 DO
        OUTPUT "Your contacts are full, please enter 2 or 3"
        INPUT Choice
    ENDWHILE
    WHILE Choice < 1 OR Choice > 3 DO
        OUTPUT "Incorrect entry – please enter 1, 2, or 3"
        INPUT Choice
    ENDWHILE
```

Question	Answer	Marks
11		

```
// enter new contacts
    IF Choice = 1
      THEN
        OUTPUT "How many contacts (1 to 5 only)?"
        INPUT NewContacts
// validates new contacts input
        WHILE NewContacts < 1 OR NewContacts > 5 DO
            OUTPUT "You may only enter between 1 and 5 contacts. Please try again"
            INPUT NewContacts
        ENDWHILE
// checks the maximum size is not exceeded
        WHILE CurrentSize + NewContacts > 100
            OUTPUT "Not enough space in your contacts"
            OUTPUT "The maximum number you may input is ", 100 - CurrentSize
            INPUT NewContacts
        ENDWHILE
        FOR Count ← CurrentSize + 1 TO CurrentSize + NewContacts
            OUTPUT "Enter the contact name as last name, first name"
            INPUT Contacts[Count, 1]
            OUTPUT "Enter the telephone number"
            INPUT Contacts[Count, 2]
        NEXT Count
        CurrentSize ← CurrentSize + NewContacts
// bubble sort to sort array if it contains 2 or more contacts
        IF CurrentSize >= 2
          THEN
            REPEAT
                Flag ← FALSE
                FOR Count ← 1 TO CurrentSize-1
                    IF Contacts[Count + 1, 1] <
                      Contacts[Count, 1]
                        THEN
                          Flag ← TRUE
                          Temp1 ← Contacts[Count, 1]
                          Temp2 ← Contacts[Count, 2]
```

Question	Answer	Marks
11		

```
                          Contacts[Count, 1] ← Contacts[Count + 1, 1]
                          Contacts[Count, 2] ← Contacts[Count + 1, 2]
                          Contacts[Count + 1, 1] ← Temp1
                          Contacts[Count + 1, 2] ← Temp2

                    ENDIF
                NEXT Count
            UNTIL NOT Flag
        ENDIF
    ENDIF
// display all contacts
    IF Choice = 2
      THEN
        IF CurrentSize > 0
          THEN
            OUTPUT "Name and Telephone Number"
            FOR Count ← 1 TO CurrentSize
                OUTPUT Contacts[Count, 1], "   ", Contacts[Count, 2]
                NEXT Count
        ENDIF
    ENDIF
// delete all contacts
    IF Choice = 3
      THEN
        FOR Count ← 1 TO 100
            FOR Count2 ← 1 TO 2
                Contacts[Count, Count2] ← ""
            NEXT Count2
        NEXT Count
    ENDIF
ENDWHILE
```

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.*