AQA

Please write clearly in block capitals.

Centre number		Candidate number	

Surname ______________________

Forename(s) ______________________

Candidate signature ______________________
I declare this is my own work.

GCSE
COMPUTER SCIENCE

Paper 1 Computational Thinking and Problem-Solving

Time allowed: 1 hour 30 minutes

Materials
There are no additional materials required for this paper.

Instructions
- Use black ink or black ball-point pen. Use pencil only for drawing.
- Answer **all** questions.
- You must answer the questions in the spaces provided.
- If you need extra space for your answer(s), use the lined pages at the end of this book. Write the question number against your answer(s).
- Do all rough work in this book. Cross through any work you do not want to be marked.
- Unless the question states otherwise, you are free to answer questions that require a coded solution in whatever format you prefer as long as your meaning is clear and unambiguous.
- You must **not** use a calculator.

Information
The total number of marks available for this paper is 80.

For Examiner's Use	
Question	Mark
1	
2	
3	
4	
5	
6	
7	
8	
9	
TOTAL	

Advice

For the multiple-choice questions, completely fill in the lozenge alongside the appropriate answer.

CORRECT METHOD   WRONG METHODS

If you want to change your answer you must cross out your original answer as shown.

If you wish to return to an answer previously crossed out, ring the answer you now wish to select as shown.

Do not write outside the box

There are no questions printed on this page

DO NOT WRITE ON THIS PAGE
ANSWER IN THE SPACES PROVIDED

Do not write outside the box

Answer **all** questions.

0 1 Match the computer science process to each correct label.

You should write a label **A–F** next to each process.

You should **not** use the same label more than once.

[3 marks]

A Abstraction

B Data validation

C Decomposition

D Efficiency

E Random number generation

F Variable assignment

Process	**Label (A–F)**
Breaking down a problem into sub-problems.	
Removing unimportant details.	
Ensuring the user enters data that is allowed, for example within a correct range.	

3

Turn over for the next question

Do not write outside the box

0 2 The algorithm shown in **Figure 1** is designed to help an athlete with their training. It uses two subroutines `getBPM` and `wait`:

- `getBPM()` returns the athlete's heart rate in beats per minute from an external input device
- `wait(n)` pauses the execution of the algorithm for `n` seconds, so `wait(60)` would pause the algorithm for 60 seconds.

Line numbers have been included but are not part of the algorithm.

Figure 1

```
seconds ← 0
rest ← 50
REPEAT
   bpm ← getBPM()
   effort ← bpm – rest
   IF effort ≤ 30 THEN
      OUTPUT 'faster'
   ELSE
      IF effort ≤ 50 THEN
         OUTPUT 'steady'
      ELSE
         OUTPUT 'slower'
      ENDIF
   ENDIF
   wait(60)
   seconds ← seconds + 60
UNTIL seconds > 200
```

0 2 . 1 State the most appropriate data type of the variable `seconds` in the algorithm shown in **Figure 1**.

[1 mark]

____

____

0 2 . 2 Explain why `rest` could have been defined as a constant in the algorithm shown in **Figure 1**.

[1 mark]

____

____

Do not write outside the box

0 2 . 3 State the line number where iteration is first used in the algorithm shown in **Figure 1**.

[1 mark]

0 2 . 4 Complete the trace table for the algorithm shown in **Figure 1**.

Some values have already been entered in the trace table:

- the first value of `seconds`
- the values returned by the subroutine `getBPM` that are assigned to the variable `bpm`.

You may not need to use all rows of the trace table.

[4 marks]

seconds	bpm	effort	OUTPUT
0	70		
	80		
	100		
	120		

7

Turn over for the next question

0 3 A developer is writing a program to convert a sequence of integers that represent playing cards to Unicode text.

The developer has identified that they need to create the subroutines shown in **Figure 2** to complete the program.

Figure 2

Subroutine	**Purpose**
`getSuit(n)`	Returns: • the string '`hearts`' if `n` is `0` • the string '`diamonds`' if `n` is `1` • the string '`spades`' if `n` is `2` • the string '`clubs`' if `n` is `3`.
`getRank(n)`	Returns the number value of the card as a string, for example: • if `n` is `1` then '`ace`' is returned • if `n` is `2` then '`two`' is returned • if `n` is `10` then '`ten`' is returned • if `n` is `11` then '`jack`' is returned.
`convert(cards)`	Returns the complete string representation of the array `cards`. For example: • if `cards` is `[3, 1]`, the string returned would be '`three of diamonds `' • if `cards` is `[1, 0, 5, 2, 7, 0]`, the string returned would be '`ace of hearts five of spades seven of hearts `'.

0 3 . 1 Explain how the developer has used the structured approach to programming.

[2 marks]

Do not write outside the box

0 3 . 2 State **two** benefits to the developer of using the three separate subroutines described in **Figure 2** instead of writing the program without using subroutines.

[2 marks]

0 3 . 3 **Figure 3** shows the subroutine `convert` described in **Figure 2**.

Some parts of the subroutine have been replaced with the labels **L1** to **L5**.

Figure 3

```
SUBROUTINE convert(cards)
   result ← ''
   max ← LEN(cards)
   index ← 0
   WHILE index < L1
      rank ← L2(cards[index])
      suit ← getSuit(cards[L3 + 1])
      c ← rank + ' of ' + suit + ' '
      result ← result + L4
      index ← index + 2
   ENDWHILE
   RETURN L5
ENDSUBROUTINE
```

State the pseudo-code that should be written in place of the labels in the subroutine written in **Figure 3**.

[5 marks]

L1

L2

L3

L4

L5

Question 03 continues on the next page

Turn over ►

Do not write outside the box

0 3 . 4 Shade **one** lozenge that states why Unicode is now commonly used in preference to ASCII.

[1 mark]

A Unicode can be represented in hexadecimal. ⬭

B Unicode includes characters from many different alphabets. ⬭

C Unicode is a sequential character set. ⬭

D Unicode is easier to remember than ASCII. ⬭

E Unicode takes up less space in memory than ASCII. ⬭

10

0 4 . 1 A student has written the following statements about representing images. Two are correct and two are incorrect:

Statement 1
"Bitmap images are made up of pixels."

Statement 2
"A 2 pixel by 4 pixel bitmap image contains 16 pixels."

Statement 3
"A pixel is a single point in a graphical image."

Statement 4
"Black and white images have a minimum colour depth of two."

Write the correct versions of the **two** incorrect statements that the student has made.

[2 marks]

First corrected statement ______________________________

____

____

Second corrected statement ______________________________

____

____

0 4 . 2 Calculate the minimum file size in **bits** of a 10 pixel by 10 pixel image with a colour depth of 3 bits.

[1 mark]

0 4 . 3 Calculate the minimum file size in **bytes** of a 10 pixel by 10 pixel image with 12 different colours.

You should show your working.

[3 marks]

Question 04 continues on the next page

Do not write outside the box

0 4 . 4 This is one row of a bitmap image that uses different shades of grey:

This row is stored using the following numbers to represent the different shades of grey:

56	34	0	99	72	23

The algorithm shown in **Figure 4** uses this row.

Figure 4

```
row ← [56, 34, 0, 99, 72, 23]
newRow ← [0, 0, 0, 0, 0, 0]
FOR i ← 0 TO 5
   IF row[i] > 50 THEN
      newRow[i] ← 99
   ENDIF
ENDFOR
```

Complete the trace table for the algorithm shown in **Figure 4**. The first values have already been entered. You may not need to use all rows of the trace table.

[3 marks]

`i`	`newRow`					
	0	**1**	**2**	**3**	**4**	**5**
	0	0	0	0	0	0
0						

Do not write outside the box

0 4 . 5 State the purpose of the algorithm shown in **Figure 4**.

[1 mark]

____

____

10

Turn over for the next question

Do not write outside the box

0 5 . 1 The following are three types of program translator:

A Assembler
B Compiler
C Interpreter

Write the label (**A**–**C**) for the type of translator next to the description.

[2 marks]

Description	**Label (A–C)**
Converts a low-level language designed to be human-readable into machine code.	
Reads a high-level program line-by-line and calls corresponding subroutines.	
Takes the entire high-level program as input and produces machine code.	

0 5 . 2 State **two** advantages of programming using a high-level language compared with programming using a low-level language.

[2 marks]

Advantage 1 ______________________________

____

Advantage 2 ______________________________

____

Do not write outside the box

0 5 . 3 Develop an algorithm, using either pseudo-code or a flowchart, that checks if the user has entered a string that represents a valid machine code instruction.

The machine code instruction is valid if it contains exactly eight characters **and** all of those characters are either `'0'` or `'1'`.

The algorithm should:

- prompt the user to enter an 8-bit machine code instruction and store it in a variable
- check that the instruction only contains the characters `'0'` or `'1'`
- check that the instruction is exactly eight characters long
- output `'ok'` when the instruction is valid, otherwise it should output `'wrong'`.

For example:

- if the user enters the string `'00101110'` it should output `'ok'`
- if the user enters the string `'11110'` it should output `'wrong'`
- if the user enters the string `'1x011001'` it should output `'wrong'`.

[9 marks]

13

Do not write outside the box

0 6 . 1 State the comparisons that would be made when the **linear search algorithm** is used to search for the value 8 in the following array (array indices have been included above the array).

0	1	2	3	4	5	6
4	7	8	13	14	15	17

[3 marks]

0 6 . 2 State the comparisons that would be made when the **binary search algorithm** is used to search for the value 8 in the following array (array indices have been included above the array).

0	1	2	3	4	5	6
4	7	8	13	14	15	17

[3 marks]

0 6 . 3 State why binary search is considered a better algorithm than linear search.

[1 mark]

Do not write outside the box

0 6 . 4 The algorithm in **Figure 5** is a new search algorithm.

Figure 5

```
arr ← [3, 4, 6, 7, 11, 14, 17, 18, 34, 42]
value ← 21
found ← False
finished ← False
i ← 0
down ← False
WHILE (found = False) AND (finished = False)
   IF arr[i] = value THEN
      found ← True
   ELSE
      IF arr[i] > value THEN
         down ← True
         i ← i - 1
      ELSE
         IF (arr[i] < value) AND (down = True) THEN
            finished ← True
         ELSE
            i ← i + 4
         ENDIF
      ENDIF
   ENDIF
ENDWHILE
```

Complete the trace table for the algorithm in **Figure 5**. The first row has been completed for you. You may not need to use all rows of the trace table.

[4 marks]

`found`	`finished`	`i`	`down`
False	False	0	False

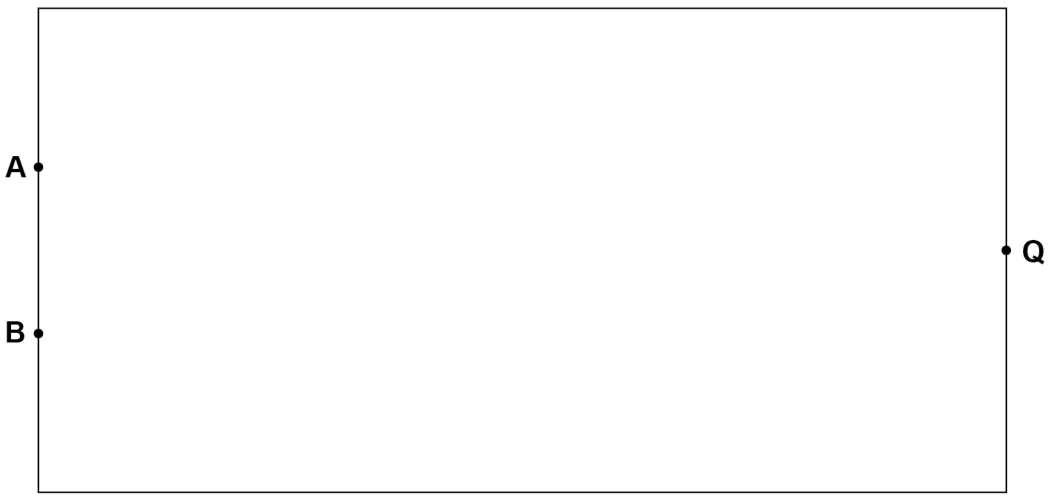
11

Do not write outside the box

0 7 . 1 Draw the logic circuit, using only one logic gate, that is represented by the following truth table:

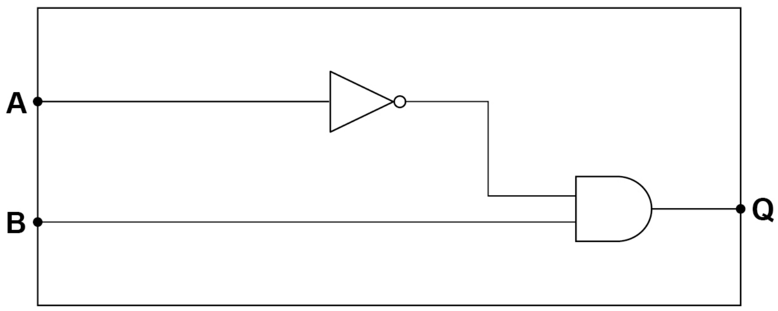
Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	1

[1 mark]



0 7 . 2 Shade **one** lozenge to show the Boolean expression that is equivalent to the logic circuit shown in **Figure 6**.

Figure 6



[1 mark]

A `A AND NOT B` ⬭

B `NOT (A AND B)` ⬭

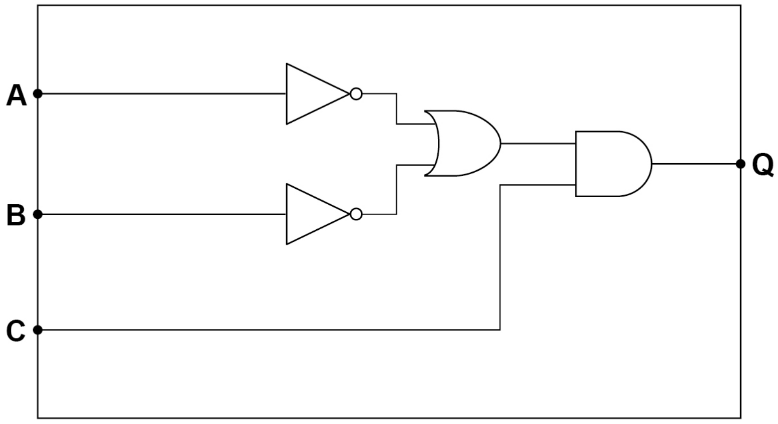
C `(NOT A) AND B` ⬭

D `(NOT A) AND (NOT B)` ⬭

Do not write outside the box

0 7 . 3 Shade **one** lozenge to show the Boolean expression which is equivalent to the logic circuit shown in **Figure 7**.

Figure 7



[1 mark]

A `NOT ((A OR B) AND C)` ⬭

B `(NOT A) OR ((NOT B) AND C)` ⬭

C `(NOT (A OR B)) AND C` ⬭

D `((NOT A) OR (NOT B)) AND C` ⬭

Question 07 continues on the next page

Do not write outside the box

0 7 . 4 **Figure 8** shows an algorithm.

Figure 8

```
x ← True
y ← False
IF NOT (x AND y) THEN
   OUTPUT 'A'
   IF NOT((NOT x) OR (NOT y)) THEN
      OUTPUT 'B'
   ELSE
      OUTPUT 'C'
   ENDIF
ELSE
   OUTPUT 'D'
   IF (NOT x) AND (NOT y) THEN
      OUTPUT 'E'
   ELSE
      OUTPUT 'F'
   ENDIF
ENDIF
```

State the output from the algorithm shown in **Figure 8**.

[2 marks]

Do not write outside the box

0 7 . 5 Draw a logic circuit in the box below for the following scenario.

A sewing machine is running (R) if either the foot pedal is on (F) or the hand dial is on (H) but not both.

You should use **only** the gates AND, OR and NOT in your answer.

[3 marks]

Question 07 continues on the next page

Do not write outside the box

0 7 . 6 Develop an algorithm, using either pseudo-code or a flowchart, that prompts the user to enter three values. It should output `'duplicate'` if at least two of these values are the same.

The start of the algorithm has been written for you.

[3 marks]

```
v1 ← USERINPUT
v2 ← USERINPUT
v3 ← USERINPUT
```

11

Do not write outside the box

0 8 Number the following lines of code in order (**1–4**) so that they create an algorithm where the final value of the variable `n` is 13.

The `LEFTSHIFT` operator performs a binary left shift.
For example, `4 LEFTSHIFT 2` would left shift the value 4 twice.

[3 marks]

Line of code	**Position (1–4 where 1 is the first line)**
`t ← t - 1`	
`n ← t – n`	
`n ← 2`	
`t ← n LEFTSHIFT 3`	

3

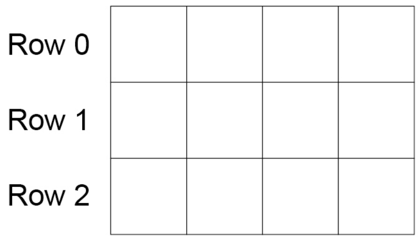
Turn over for the next question

Do not write outside the box

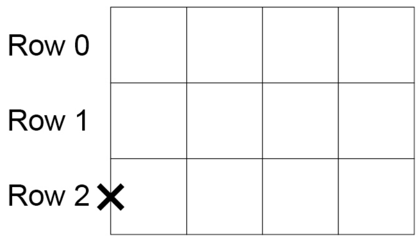
0 9 The **Algebraic Patent Sewing Machine** is a programmable sewing machine that creates patterns on rows of cloth. It is controlled by writing programs that use the following subroutines:

Subroutine	**Description**
`gotoRow(n)`	start the sewing machine needle at the left-hand side of row `n`
`move(n)`	move the needle forward by `n` cells without producing a pattern
`shape(s)`	produce shape `s` where `s` can be `'square'` or `'circle'` and move the needle to the next cell
`atEnd()`	returns `True` if the needle is at the end of the row or `False` otherwise

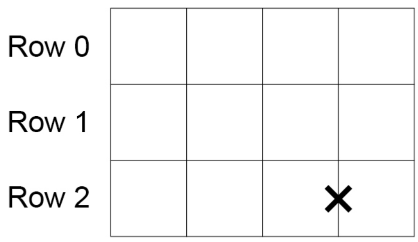
For example, if the cloth looks like this to begin with:



The subroutine call `gotoRow(2)` will place the sewing machine needle at the point shown by the black cross:

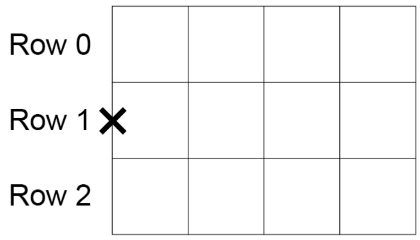


The subroutine call `move(3)` will move the sewing machine needle to the point shown by the black cross:

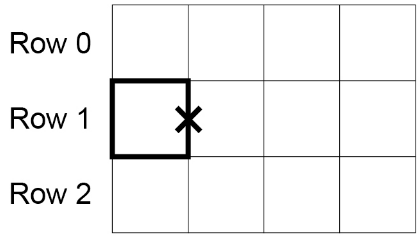


Do not write outside the box

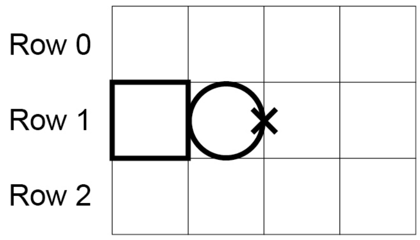
The subroutine call `gotoRow(1)` will move the sewing machine needle to the point shown by the black cross:



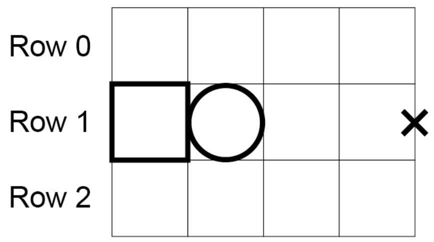
The subroutine call `shape('square')` will draw the following pattern and move the sewing machine needle to the point shown by the black cross:



And finally, the subroutine call `shape('circle')` will draw the following pattern and move the sewing machine needle to the point shown by the black cross:



All of the previous positions of the sewing machine needle would result in the subroutine call `atEnd()` returning `False`, however in the following example `atEnd()` would return `True`:



Question 09 continues on the next page

0 9 . 1 Draw the final pattern after the following algorithm has executed.

```
gotoRow(0)
WHILE atEnd() = False
   shape('square')
   move(1)
ENDWHILE
gotoRow(1)
shape('circle')
move(1)
IF atEnd() = True THEN
   gotoRow(2)
ELSE
   move(1)
ENDIF
shape('square')
```

You should draw your answer on the following grid.

You do not need to show the position(s) of the needle in your answer.

[4 marks]

Row 0				
Row 1				
Row 2				

Do not write outside the box

0 9 . 2 Draw the final pattern after the following algorithm has executed.

This question uses the `MOD` operator. `MOD` calculates the remainder after integer division, for example `7 MOD 5 = 2`.

```
patterns ← ['circle', 'square', 'square', 'circle']
r ← 2
FOR k ← 0 TO 3
   gotoRow(k MOD r)
   move(k + 1)
   shape(patterns[k])
ENDFOR
```

You should draw your answer on the following grid.

You do not need to show the position(s) of the needle in your answer.

[4 marks]

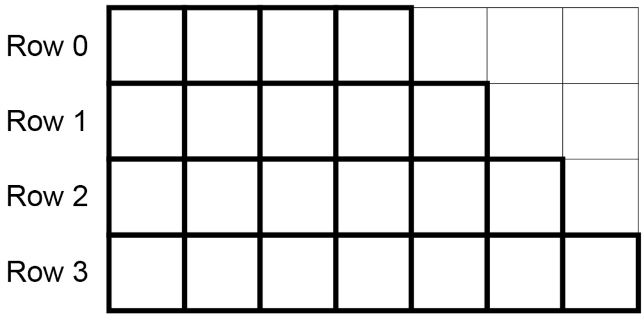
Row 0							
Row 1							
Row 2							
Row 3							

Question 09 continues on the next page

0 9 . 3 Develop an algorithm, using either pseudo-code or a flowchart, to produce the pattern shown in **Figure 9**.

To gain full marks your answer must make appropriate use of iteration.

Figure 9



[4 marks]

12

END OF QUESTIONS

Do not write outside the box

There are no questions printed on this page

DO NOT WRITE ON THIS PAGE
ANSWER IN THE SPACES PROVIDED

Do not write outside the box

Question number	Additional page, if required. Write the question numbers in the left-hand margin.

Do not write outside the box

Question number	**Additional page, if required. Write the question numbers in the left-hand margin.**

Do not write outside the box

Question number	**Additional page, if required. Write the question numbers in the left-hand margin.**

Do not write outside the box

Question number	**Additional page, if required.** **Write the question numbers in the left-hand margin.**

Do not write outside the box

There are no questions printed on this page

DO NOT WRITE ON THIS PAGE
ANSWER IN THE SPACES PROVIDED