

A



**Surname** \_\_\_\_\_

**Other Names** \_\_\_\_\_

**Centre Number** \_\_\_\_\_

**Candidate Number** \_\_\_\_\_

**Candidate Signature** \_\_\_\_\_

**I declare this is my own work.**

# **GCSE COMPUTER SCIENCE**

**Paper 1 Computational Thinking and Problem-Solving  
8520/1**

**Monday 11 May 2020**

**Morning**

**Time allowed: 1 hour 30 minutes**

**At the top of the page, write your surname and other names, your centre number, your candidate number and add your signature.**

**[Turn over]**



J U N 2 0 8 5 2 0 1 0 1

**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.**

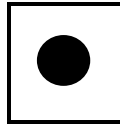
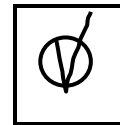
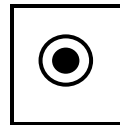
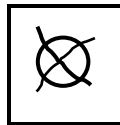
## **INFORMATION**

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

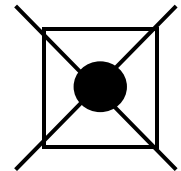


**ADVICE**

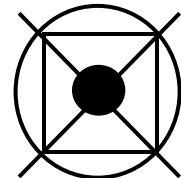
For the multiple-choice questions, completely fill in the lozenge alongside the appropriate answer.

**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 TURN OVER UNTIL TOLD TO DO SO**



**Answer ALL questions.**

**0 1 . 1** A bitmap image is represented as a grid of pixels.

**State what is meant by the term pixel.  
[1 mark]**

---

---

---

**0 1 . 2** State the maximum number of different colours that can be used if a bitmap image has a colour depth of six bits. [1 mark]

---

---

---



**BLANK PAGE**

**[Turn over]**





---

---

---

---

**Answer** \_\_\_\_\_ **kB**

**[Turn over]**



- 01.4** The algorithm shown in FIGURE 1 converts binary data entered as a string by the user into a representation of a black and white image.

The algorithm uses the + operator to concatenate two strings.

Characters in the string are indexed starting at zero. For example `bdata[2]` would access the third character of the string stored in the variable `bdata`

The MOD operator calculates the remainder after integer division, for example

$$17 \text{ MOD } 5 = 2$$

### FIGURE 1

```

bdata ← USERINPUT
image ← ''
FOR i ← 0 TO LEN(bdata) - 1
    IF bdata[i] = '0' THEN
        image ← image + '*'
    ELSE
        image ← image + '/'
    ENDIF
    IF i MOD 3 = 2 THEN
        OUTPUT image
        image ← ''
    ENDIF
ENDFOR

```





Complete the trace table for the algorithm shown in FIGURE 1 when the variable `bdata` is given the following value from the user:

110101

You may not need to use every row in the table. The algorithm output is not required. [3 marks]

i	image

[Turn over]





- 03.1** State the name of the logic gate represented by the following truth table. [1 mark]

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

Logic gate \_\_\_\_\_

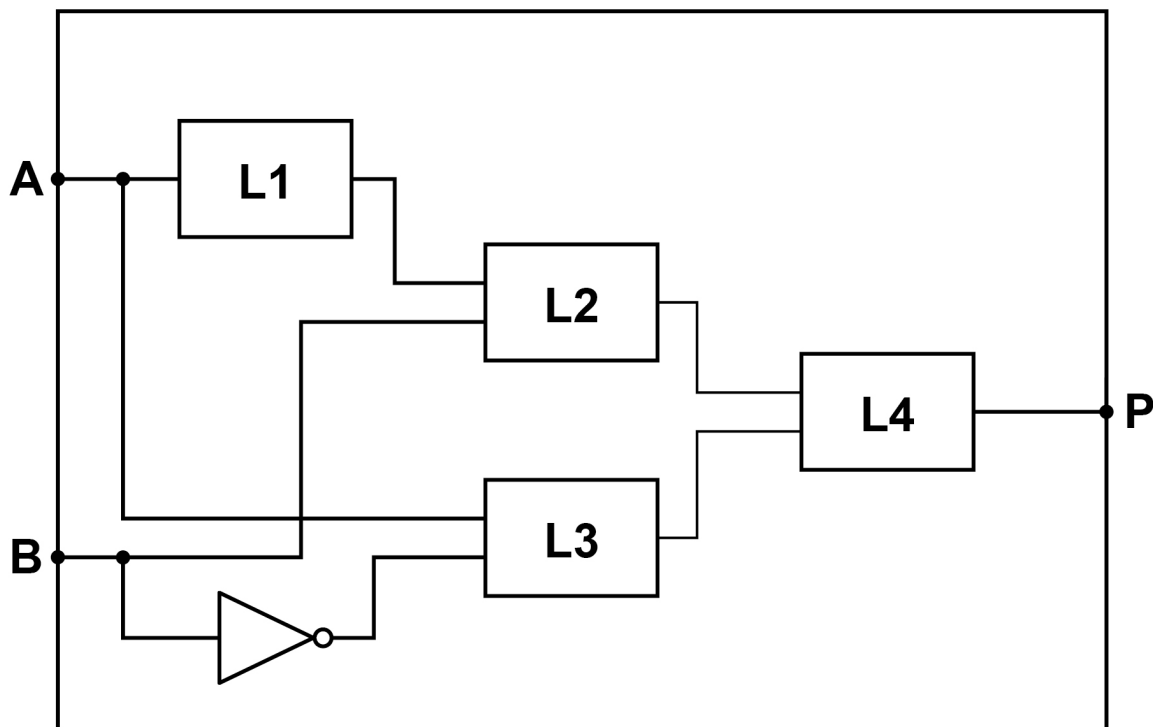
[Turn over]



A partially complete logic circuit is shown in FIGURE 2 that detects if a computer system has been set up correctly. There are two keyboard input devices, keyboard A and keyboard B, and either one can be connected to the computer system. However, if they are both connected then the computer system will not work.

Output P has the value 1 if either keyboard A or keyboard B, but not both, is connected to the computer system and 0 otherwise.

FIGURE 2



- 03.2** State the name of the logic gates that should be placed in the positions indicated by the labels L1, L2, L3 and L4 in FIGURE 2. [3 marks]

LABEL	LOGIC GATE
L1	
L2	
L3	
L4	

[Turn over]

4



04

The algorithm shown in FIGURE 3 is used to check if the start of an instruction for a particular assembly language is valid.

The string representation of the assembly language instruction is stored in the variable `instr`

Characters in the string are indexed starting at zero. For example `instr[2]` would access the third character of the string stored in the variable `instr`

FIGURE 3

```
code ← ''
i ← 0
WHILE instr[i] ≠ ':' AND i < 4
    code ← code + instr[i]
    i ← i + 1
ENDWHILE
valid ← False
IF code = 'ADD' OR code = 'SUB' OR code = 'HALT' THEN
    valid ← True
ENDIF
```



04.1

Shade ONE lozenge to show the most appropriate data type of the variable `i` in the algorithm in FIGURE 3. [1 mark]

A Character

B Integer

C Real

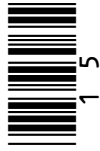
D String

04.2

State the data type of the variable `valid` in the algorithm in FIGURE 3. [1 mark]

---

[Turn over]



## REPEAT OF FIGURE 3

```
code ← ''
i ← 0
WHILE instr[i] ≠ ':' AND i < 4
    code ← code + instr[i]
    i ← i + 1
ENDWHILE
valid ← False
IF code = 'ADD' OR code = 'SUB' OR code = 'HALT' THEN
    valid ← True
ENDIF
```





04.3

State the final value of the variable `valid` in the algorithm in FIGURE 3 for the following different starting values of `instr` [3 marks]

Value of <code>instr</code>	Final value of <code>valid</code>
ADD R0, R1	
ADD: R0, R1	
HALT	

04.4

State what an assembly language program must be translated into before it can be executed by a computer. [1 mark]

---

---

[Turn over]



**04.5** State TWO reasons why a programmer, who can program in both high-level and low-level languages, would usually choose to develop in a high-level language rather than a low-level language. [2 marks]

**Reason 1** \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

**Reason 2** \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

**BLANK PAGE**

**[Turn over]**



**04.6** Develop an algorithm, using either pseudo-code OR a flowchart, that:

- **initialises a variable called `regValid` to `False`**
- **sets a variable called `regValid` to `True` if the string contained in the variable `reg` is an uppercase `R` followed by the character representation of a single numeric digit.**

**Examples:**

- **if the value of `reg` is `R0` or `R9` then `regValid` should be `True`**
- **if the value of `reg` is `r6` or `Rh` then `regValid` should be `False`**

**You may wish to use the subroutine `isDigit(ch)` in your answer. The subroutine `isDigit` returns `True` if the character parameter `ch` is a string representation of a digit and `False` otherwise.**  
**[3 marks]**

---

---

---

---



05

The algorithms shown in FIGURE 4 and FIGURE 5 both have the same purpose.

The operator LEFTSHIFT performs a binary shift to the left by the number indicated.

For example, 6 LEFTSHIFT 1 will left shift the number 6 by one place, which has the effect of multiplying the number 6 by two giving a result of 12

#### FIGURE 4

```
result ← number LEFTSHIFT 2
result ← result - number
```

#### FIGURE 5

```
result ← 0
FOR x ← 1 TO 3
    result ← result + number
ENDFOR
```



- 05.1** Complete the trace table for the algorithm shown in FIGURE 4 when the initial value of `number` is 4

You may not need to use all rows of the trace table. [2 marks]

result

[Turn over]



- 05.2** Complete the trace table for the algorithm shown in **FIGURE 5**, on page 22, when the initial value of `number` is 4

You may not need to use all rows of the trace table. [2 marks]

x	result





- 05.3** The algorithms in FIGURE 4 and FIGURE 5 have the same purpose.

State this purpose. [1 mark]

---

---

---

- 05.4** Explain why the algorithm shown in FIGURE 4 can be considered to be a more efficient algorithm than the algorithm shown in FIGURE 5. [1 mark]

---

---

---

[Turn over]

6



0	6
---	---

Show the steps involved, on the opposite page, for either the bubble sort algorithm OR the merge sort algorithm, to sort the array shown in **FIGURE 6** so the result is

[1, 4, 5, 8]

**FIGURE 6**

[8, 4, 1, 5]

**CIRCLE** the algorithm you have chosen:

**Bubble sort**

**Merge sort**

**[4 marks]**



**Steps:**

**[Turn over]**



**07.1** Four subroutines are shown in FIGURE 7.**FIGURE 7**

```
SUBROUTINE main(k)
  OUTPUT k
  WHILE k > 1
    IF isEven(k) = True THEN
      k ← decrease(k)
    ELSE
      k ← increase(k)
    ENDIF
  OUTPUT k
  ENDWHILE
ENDSUBROUTINE
```

```
SUBROUTINE decrease(n)
  result ← n DIV 2
  RETURN result
ENDSUBROUTINE
```

```
SUBROUTINE increase(n)
  result ← (3 * n) + 1
  RETURN result
ENDSUBROUTINE
```

```
SUBROUTINE isEven(n)
  IF (n MOD 2) = 0 THEN
    RETURN True
  ELSE
    RETURN False
  ENDIF
ENDSUBROUTINE
```



Complete the table showing **ALL** of the outputs from the subroutine call `main(3)`

The first output has already been written in the trace table. You may not need to use all rows of the table. [4 marks]

Output
3

[Turn over]



**REPEAT OF FIGURE 7**

```
SUBROUTINE main(k)
  OUTPUT k
  WHILE k > 1
    IF isEven(k) = True THEN
      k ← decrease(k)
    ELSE
      k ← increase(k)
    ENDIF
    OUTPUT k
  ENDWHILE
ENDSUBROUTINE
```

```
SUBROUTINE decrease(n)
  result ← n DIV 2
  RETURN result
ENDSUBROUTINE
```

```
SUBROUTINE increase(n)
  result ← (3 * n) + 1
  RETURN result
ENDSUBROUTINE
```

```
SUBROUTINE isEven(n)
  IF (n MOD 2) = 0 THEN
    RETURN True
  ELSE
    RETURN False
  ENDIF
ENDSUBROUTINE
```



**07.2** Describe how the developer has used the structured approach to programming in FIGURE 7. [2 marks]

---

---

---

---

---

---

---

---

[Turn over]

10



**08**

**The subroutine `CODE_TO_CHAR` can be used to convert a character code into the corresponding Unicode character. For example:**

`CODE_TO_CHAR(97)` **will return the character 'a'**

`CODE_TO_CHAR(65)` **will return the character 'A'**

**The subroutine `CHAR_TO_CODE` can be used to convert a Unicode character into the corresponding character code. For example:**

`CHAR_TO_CODE('a')` **will return the integer 97**

`CHAR_TO_CODE('A')` **will return the integer 65**





**08** . **1** Shade ONE lozenge to show what value would be returned from the subroutine call `CODE_TO_CHAR(100)` [1 mark]

**A** 'c'

**B** 'd'

**C** 'e'

**D** 'f'

**08** . **2** State the value that will be returned from the subroutine call:

`CODE_TO_CHAR(CHAR_TO_CODE('E') + 2)`

[1 mark]

Value returned \_\_\_\_\_

[Turn over]



**08.3** Write a subroutine `TO_LOWER`, using either pseudo-code OR a flowchart, that takes an upper case character as a parameter and returns the corresponding lower case character.

For example, if the subroutine `TO_LOWER` is passed the character 'A' as a parameter, the subroutine should return the character 'a'.

You should make use of the subroutines `CODE_TO_CHAR` and `CHAR_TO_CODE` in your answer.

You can assume that the parameter passed to the subroutine will be in upper case. [5 marks]

---

---

---

---

---

---

---

---

---

---





**0 9**

**A developer needs to store data about thousands of songs in a program. She needs to be able to hold information on every song's title, singer and year of release.**

**Explain how the developer could use a combination of an array and records to store this information.**

**In your answer you should refer to the data types that would be used by the developer.  
[4 marks]**

---

---

---

---

---

---

---

---

---

---

**[Turn over]**





**BLANK PAGE**

**[Turn over]**



1	0
---	---

An application allows only two users to log in. Their usernames are stated in TABLE 1 along with their passwords.

TABLE 1

username	password
gower	9Fdg3
tuff	888rG

Develop an algorithm, using either pseudo-code OR a flowchart, that authenticates the user. The algorithm should:

- get the user to enter their username and password
- check that the combination of username and password is correct and, if so, output the string 'access granted'
- get the user to keep re-entering their username and password until the combination is correct.

[6 marks]

---

---

---













**BLANK PAGE**

**[Turn over]**



1	1
---	---

**Develop an algorithm, using either pseudo-code OR a flowchart, that helps an ice cream seller in a hot country calculate how many ice creams they are likely to sell on a particular day. Your algorithm should:**

- **get the user to enter whether it is the weekend or a weekday**
- **get the user to enter the temperature forecast in degrees Celsius (they should enter a number between 20 and 45 inclusive; if the number falls outside of this range then they should be made to re-enter another number until they enter a valid temperature)**
- **calculate the number of ice creams that are likely to be sold using the following information:**
  - **100 ice creams are likely to be sold if the temperature is between 20 and 30 degrees inclusive,**
  - **150 ice creams are likely to be sold if the temperature is between 31 and 38 degrees inclusive,**
  - **and 120 ice creams are likely to be sold if the temperature is higher than 38 degrees**
- **double the estimate if it is a weekend**
- **output the estimated number of ice creams that are likely to be sold.**

**[9 marks]**















1 2

A developer has written a set of subroutines to control an array of lights. The lights are indexed from zero. They are controlled using the subroutines in TABLE 2.

TABLE 2

SUBROUTINE	EXPLANATION
SWITCH ( $n$ )	<p>If the light at index <math>n</math> is on it is set to off.</p> <p>If the light at index <math>n</math> is off it is set to on.</p>
NEIGHBOUR ( $n$ )	<p>If the light at index <math>(n+1)</math> is on, the light at index <math>n</math> is also set to on.</p> <p>If the light at index <math>(n+1)</math> is off, the light at index <math>n</math> is also set to off.</p>
RANGE OFF ( $m, n$ )	<p>All the lights between index <math>m</math> and index <math>n</math> (but NOT including <math>m</math> and <math>n</math>) are set to off.</p>



**Array indices are shown above the array of lights.**

**For example, if the starting array of the lights is**

0	1	2	3
off	on	off	on

**Then after the subroutine call SWITCH(2) the array of lights will become**

0	1	2	3
off	on	on	on

**And then after the subroutine call NEIGHBOUR(0) the array of lights will become**

0	1	2	3
on	on	on	on

**Finally, after the subroutine call RANGEOFF(0, 3) the array of lights will become**

0	1	2	3
on	off	off	on

**[Turn over]**



**BLANK PAGE**



**1 2 . 1** If the starting array of lights is

0	1	2	3	4	5	6
on	off	off	on	off	off	on

**What will the array of lights become after the following algorithm has been followed?**

```

a ← 2
SWITCH (a)
SWITCH (a + 1)
NEIGHBOUR (a - 2)

```

**Write your final answer in the following array  
[3 marks]**

0	1	2	3	4	5	6

**[Turn over]**



**1 2 . 2** If the starting array of lights is

0	1	2	3	4	5	6
off	off	on	off	on	on	on

**What will the array of lights become after the following algorithm has been followed?**

```

FOR a ← 0 TO 2
    SWITCH (a)
ENDFOR
b ← 8
RANGEOFF ((b / 2), 6)
NEIGHBOUR (b - 4)

```

**Write your final answer in the following array**  
**[3 marks]**

0	1	2	3	4	5	6



**1 2 . 3** If the starting array of lights is

0	1	2	3	4	5	6
off	on	off	on	off	on	off

**What will the array of lights become after the following algorithm has been followed?**

```

a ← 0
WHILE a < 3
  SWITCH(a)
  b ← 5
  WHILE b ≤ 6
    SWITCH(b)
    b ← b + 1
  ENDWHILE
  a ← a + 1
ENDWHILE

```

**Write your final answer in the following array  
[3 marks]**

0	1	2	3	4	5	6

**[Turn over]**



**BLANK PAGE**



**1 2 . 4** If the starting array of lights is

0	1	2	3	4	5	6
on	on	on	on	on	on	on

Write an algorithm, using **EXACTLY THREE** subroutine calls, that means the final array of lights will be

0	1	2	3	4	5	6
off	off	off	off	off	off	off

**You must use each of the subroutines SWITCH, NEIGHBOUR and RANGE OFF EXACTLY ONCE in your answer. If you do not do this you may still be able to get some marks. [3 marks]**

---



---



---



---



---

**END OF QUESTIONS**

12







**BLANK PAGE**

For Examiner's Use	
Question	Mark
1–2	
3	
4	
5	
6–7	
8	
9	
10	
11	
12	
<b>TOTAL</b>	

**Copyright information**

For confidentiality purposes, all acknowledgements of third-party copyright material are published in a separate booklet. This booklet is published after each live examination series and is available for free download from [www.aqa.org.uk](http://www.aqa.org.uk).

Permission to reproduce all copyright material has been applied for. In some cases, efforts to contact copyright-holders may have been unsuccessful and AQA will be happy to rectify any omissions of acknowledgements. If you have any queries please contact the Copyright Team.

Copyright © 2020 AQA and its licensors. All rights reserved.

**IB/M/CD/Jun20/8520/1/E3**

