



Surname _____

Other Names _____

Centre Number _____

Candidate Number _____

Candidate Signature _____

GCSE COMPUTER SCIENCE

Paper 1 Computational thinking and problem-solving

8520/1

Monday 14 May 2018

Morning

Time allowed: 1 hour 30 minutes

- **There are no additional materials required for this paper.**

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

[Turn over]



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.
- Do all rough work in this book. Cross through any work you do not want to be marked.
- 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.

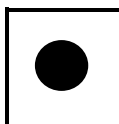
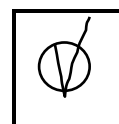
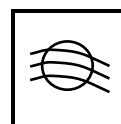
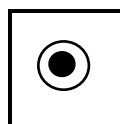
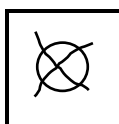
INFORMATION

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

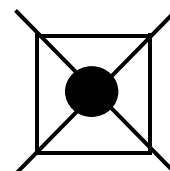


ADVICE

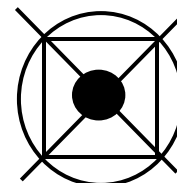
For the multiple-choice questions, completely fill in the lozenge alongside the appropriate answer.

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.

01.1 Define the term algorithm. [2 marks]



01.2 The following are computer science terms (labelled A – E).

- A abstraction**
- B data type**
- C decomposition**
- D efficiency**
- E input**

For each of the definitions in the table, write the label of the most suitable computer science term. Use a label only once. [3 marks]

	Label
Breaking a problem down into a number of sub-problems.	
The process of removing unnecessary detail from a problem.	
Defines the range of values a variable may take.	

5

[Turn over]



02 The algorithm in FIGURE 1 has been developed to automate the quantity of dog biscuits to put in a dog bowl at certain times of the day. The algorithm contains an error.

- Line numbers are included but are not part of the algorithm.

FIGURE 1

```
1      time ← USERINPUT
2      IF time = 'breakfast' THEN
3          q ← 1
4      ELSE IF time = 'lunch' THEN
5          q ← 4
6      ELSE IF time = 'dinner' THEN
7          a ← 2
8      ELSE
9          OUTPUT 'time not recognised'
10     ENDIF
11     FOR n ← 1 TO q
12         IF n < 3 THEN
13             DISPENSE_BISCUIT('chewies')
14         ELSE
15             DISPENSE_BISCUIT('crunchy')
16         ENDIF
17     ENDFOR
```



02.1 Shade ONE lozenge which shows the line number where selection is FIRST used in the algorithm shown in FIGURE 1. [1 mark]

A Line number 2

B Line number 4

C Line number 9

D Line number 12

02.2 Shade ONE lozenge which shows the line number where iteration is FIRST used in the algorithm shown in FIGURE 1. [1 mark]

A Line number 1

B Line number 8

C Line number 11

D Line number 13

[Turn over]



02.3 Shade ONE lozenge which shows how many times the subroutine `DISPENSE_BISCUIT` would be called if the user input is 'breakfast'. [1 mark]

A 1 subroutine call

B 2 subroutine calls

C 3 subroutine calls

D 4 subroutine calls

02.4 Shade ONE lozenge which shows the data type of the variable `time` in the algorithm shown in FIGURE 1. [1 mark]

A Date/Time

B String

C Integer

D Real



0 2 . 5 State how many times the subroutine `DISPENSE_BISCUIT` will be called with the parameter 'chewies' if the user input is 'lunch'. [1 mark]

0 2 . 6 State how many possible values the result of the comparison `time = 'dinner'` could have in the algorithm shown in FIGURE 1. [1 mark]

0 2 . 7 The programmer realises they have made a mistake. State the line number of the algorithm shown in FIGURE 1 where the error has been made. [1 mark]

[Turn over]



0 2 . 8 Write **ONE** line of code that would correct the error found in the algorithm in **FIGURE 1**.
[1 mark]

8



BLANK PAGE

[Turn over]



0 3

The following bit pattern represents a binary number.

00000110

0 3 . 1

What is the result of applying a left binary shift of 2 to this bit pattern? Express your answer as a bit pattern. [1 mark]

0 3 . 2

The arithmetic effect of applying a left binary shift of 1 to a binary number is to multiply that number by 2.

State the arithmetic effect of applying a left binary shift of 3 to a binary number. [1 mark]



0 3 . 3 What will be the arithmetic effect of left binary shifting a binary number by 4 and then right binary shifting the result by 5? [1 mark]

3

[Turn over]



0 4

A sound engineer is recording a singer.

0 4 . 1

Describe why the sound must be converted to a digital format before it can be stored on a computer system. [2 marks]



04.3 The sound engineer currently uses a sample resolution of 4 bits which enables a sample to be stored as one of 16 different bit patterns. She wants to increase the number of bit patterns available from 16 to 32. Shade **ONE** lozenge which shows the **MINIMUM** sample resolution (in bits) she can choose that will allow her to do this. [1 mark]

A 3 bits

B 5 bits

C 8 bits

D 16 bits



0 4 . 4 Shade ONE lozenge to show which of the following correctly states the effects of increasing the sampling rate. [1 mark]

A Decreases both the quality of the recording and the file size

B Has no effect on the quality of the recording or the file size

C Improves the quality of the recording and has no effect on file size

D Improves the quality of the recording and increases the file size

8

[Turn over]



05 The subroutine `CHAR_TO_CODE(character)` returns the integer **ASCII** value of a character. For example,

`CHAR_TO_CODE('a')` returns the value **97**
`CHAR_TO_CODE('z')` returns the value **122**
`CHAR_TO_CODE('`')` returns the value **96**
`CHAR_TO_CODE('{')` returns the value **123**

Develop an algorithm, using either pseudo-code or a flowchart, that:

- asks the user to enter a character
- outputs 'LOWER' if the user has entered a lowercase character
- outputs 'NOT LOWER' if the user has entered any other character.

You MUST use the built-in `CHAR_TO_CODE` subroutine in your answer. [7 marks]



06 The algorithm in FIGURE 2 is a sorting algorithm.

- Array indexing starts at 0.
- Line numbers are included but are not part of the algorithm.

FIGURE 2

```
1 arr ← [4, 1, 6]
2 sorted ← false
3 WHILE sorted = false
4     sorted ← true
5     i ← 0
6     WHILE i < 2
7         IF arr[i+1] < arr[i] THEN
8             t ← arr[i]
9             arr[i] ← arr[i+1]
10            arr[i+1] ← t
11            sorted ← false
12        ENDIF
13        i ← i + 1
14    ENDWHILE
15 ENDWHILE
```

06.1 State the data type of the variable `sorted` in the algorithm shown in FIGURE 2. [1 mark]



06.2 The identifier `sorted` is used in the algorithm shown in FIGURE 2.

Explain why this is a better choice than using the identifier `s`. [2 marks]

06.3 Shade ONE lozenge to show which of the following contains the FALSE statement about the algorithm in FIGURE 2. [1 mark]

A The algorithm uses a named constant

B The algorithm uses indefinite iteration

C The algorithm uses nested iteration

[Turn over]



06.4 Complete the trace table for the algorithm shown in FIGURE 2. Some values have already been entered. [6 marks]

arr			sorted	i	t
[0]	[1]	[2]			
4	1	6	false		



06.5 Fill in the values in the boxes to show how the merge part of the merge sort algorithm operates. The first and last rows have been completed for you. [3 marks]

7	3	4	1	2	8	5	6
1	2	3	4	5	6	7	8

[Turn over]



0 6 . 6 State ONE advantage of the merge sort algorithm compared to the sorting algorithm in FIGURE 2. [1 mark]

0 6 . 7 A programmer implementing the algorithm in FIGURE 2 decided to create it as a subroutine. Line 1 was removed and the array `arr` was made a parameter of the subroutine.

State TWO reasons why the programmer decided to implement the algorithm as a subroutine. [2 marks]

Reason 1: _____

Reason 2: _____



BLANK PAGE

[Turn over]



07 Develop an algorithm using either pseudo-code or a flowchart that allows a taxi company to calculate how much a taxi fare should be.

The algorithm should:

- prompt the user to enter the journey distance in kilometres
 - the distance entered must be greater than zero
 - the user should be made to re-enter the distance until the distance entered is valid
- prompt the user to enter the number of passengers (no validation is required)
- calculate the taxi fare by
 - charging £2 for every passenger regardless of the distance
 - charging a further £1.50 for every kilometre regardless of how many passengers there are
- output the final taxi fare.

[8 marks]





8



0 8 . 1 Complete the truth table for the AND logic gate.
[1 mark]

A	B	A AND B
0	0	
0	1	
1	0	
1	1	

[Turn over]

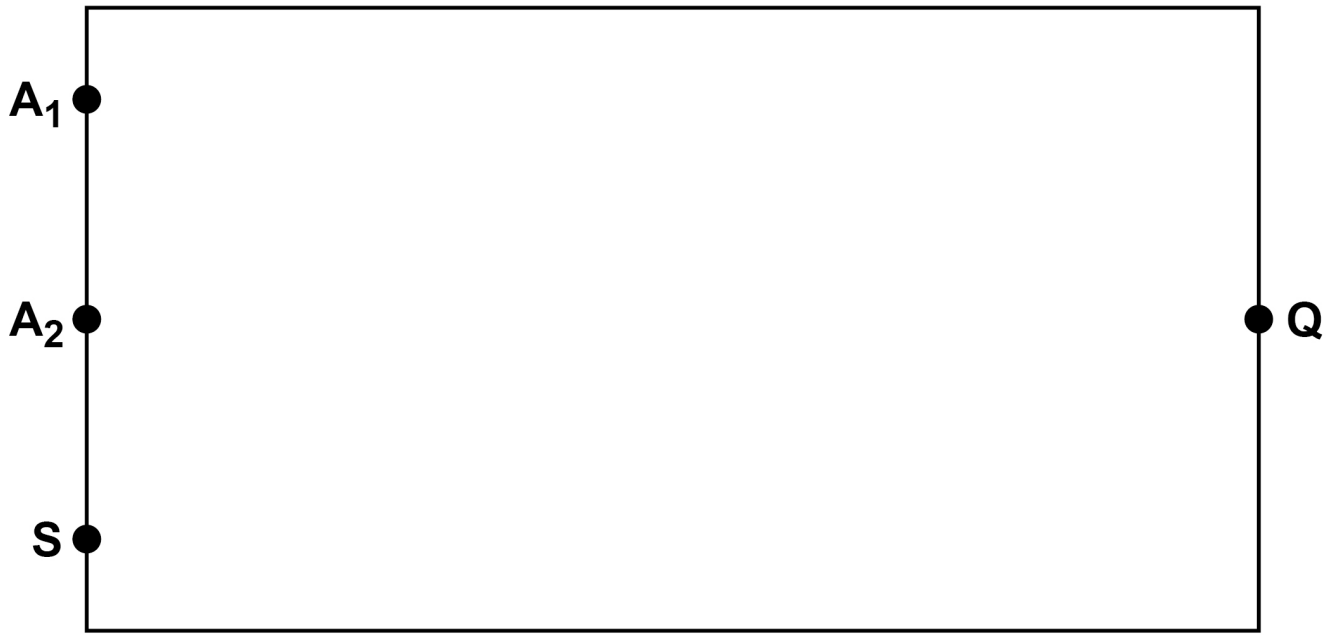


0 8 . 2 A logic circuit is being developed for an audio advert in a shop that plays automatically if a customer is detected nearby.

- The system has two sensors, A_1 and A_2 , that detect if a customer is near. The audio plays if either of these sensors is activated.
- The system should only play if another audio system, S , is not playing.
- The output from the circuit, for whether the advert should play or not, is Q .

**Complete the logic circuit for this system.
[3 marks]**





[Turn over]

4



09 The following subroutines control the way that labelled blocks are placed in different columns.

`BLOCK_ON_TOP(column)` **returns the label of the block on top of the column given as a parameter.**

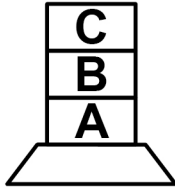
`MOVE(source, destination)` **moves the block on top of the `source` column to the top of the `destination` column.**

`HEIGHT(column)` **returns the number of blocks in the specified column.**



09.1 This is how the blocks A, B and C are arranged at the start.

Column 0



Column 1



Column 2



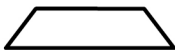
Draw the final arrangement of the blocks after the following algorithm has run.

MOVE (0, 1)

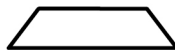
MOVE (0, 2)

MOVE (0, 2)

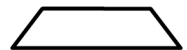
Column 0



Column 1



Column 2



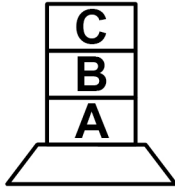
[3 marks]

[Turn over]

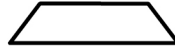


09.2 This is how the blocks A, B and C are arranged at the start.

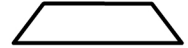
Column 0



Column 1



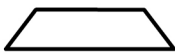
Column 2



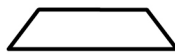
Draw the final arrangement of the blocks after the following algorithm has run.

```
WHILE HEIGHT(0) > 1
  MOVE(0, 1)
ENDWHILE
MOVE(1, 2)
```

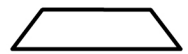
Column 0



Column 1



Column 2



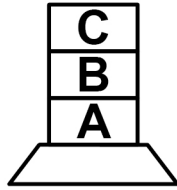
[3 marks]

[Turn over]

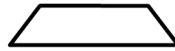


09.3 This is how the blocks A, B and C are arranged at the start.

Column 0



Column 1



Column 2

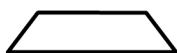


Draw the final arrangement of the blocks after the following algorithm has run.

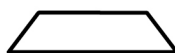
```
FOR c ← 0 TO 2
  IF BLOCK_ON_TOP(0) = 'B' THEN
    MOVE(0, (c+1) MOD 3)
  ELSE
    MOVE(0, (c+2) MOD 3)
  ENDIF
ENDFOR
```

This algorithm uses the MOD operator which calculates the remainder resulting from integer division. For example, $13 \text{ MOD } 5 = 3$.

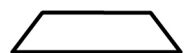
Column 0



Column 1



Column 2



[3 marks]

[Turn over]



09.4 Develop an algorithm using either pseudo-code or a flowchart that will move every block from column 0 to column 1.

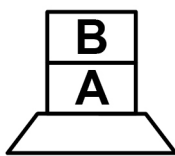
Your algorithm should work however many blocks start in column 0. You may assume there will always be at least one block in column 0 at the start and that the other columns are empty.

The order of the blocks must be preserved.

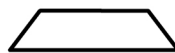
The `MOVE` subroutine must be used to move a block from one column to another. You should also use the `HEIGHT` subroutine in your answer.

For example, if the starting arrangement of the blocks is:

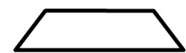
Column 0



Column 1



Column 2

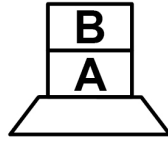
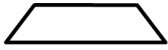


Then the final arrangement should have block B above block A:

Column 0

Column 1

Column 2



[5 marks]

[Turn over]



[Turn over]



14



BLANK PAGE

[Turn over]



10 The subroutine in FIGURE 3 is used to authenticate a username and password combination.

- Array indexing starts at 0.
- Line numbers are included but are not part of the algorithm.

FIGURE 3

```
1   SUBROUTINE Authenticate(user, pass)
2       us ← ['dave', 'alice', 'bob']
3       ps ← ['abf32', 'woof2006', '!@34E$']
4       z ← 0
5       correct ← false
6       WHILE z < 3
7           IF user = us[z] THEN
8               IF pass = ps[z] THEN
9                   correct ← true
10                  ENDIF
11              ENDIF
12              z ← z + 1
13          ENDWHILE
14          RETURN correct
15      ENDSUBROUTINE
```



10.1 Complete the trace table for the following subroutine call:

`Authenticate('alice', 'woof2006')`

[3 marks]

z	correct

[Turn over]



- 1 0 . 2** State the value that is returned by the following subroutine call:

```
Authenticate('bob', 'abf32')
```

[1 mark]

- 1 0 . 3** Lines 7 and 8 in FIGURE 3 could be replaced with a single line. Shade ONE lozenge to show which of the following corresponds to the correct new line. **[1 mark]**

A IF user = us[z] OR pass = ps[z] THEN

B IF user = us[z] AND pass = ps[z] THEN

C IF NOT (user = us[z] AND pass = ps[z]) THEN



10.4 A programmer implements the subroutine shown in FIGURE 3. He replaces line 9 with

```
RETURN true
```

He also replaces line 14 with

```
RETURN false
```

Explain how the programmer has made the subroutine more efficient. [2 marks]

7

END OF QUESTIONS



There are no questions printed on this page

For Examiner's Use	
Question	Mark
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
TOTAL	

Copyright information

For confidentiality purposes, from the November 2015 examination series, acknowledgements of third party copyright material will be published in a separate booklet rather than including them on the examination paper or support materials. This booklet is published after each examination series and is available for free download from www.aqa.org.uk after the live examination series.

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, AQA, Stag Hill House, Guildford, GU2 7XJ.

Copyright © 2018 AQA and its licensors. All rights reserved.

IB/M/Jun18/NC/8520/1/E4

