AQA

GCSE

COMPUTER SCIENCE

8520/1

Paper 1 Computational thinking and problem-solving

Mark scheme

June 2019

Version1.0 Final

Mark schemes are prepared by the Lead Assessment Writer and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation events which all associates participate in and is the scheme which was used by them in this examination. The standardisation process ensures that the mark scheme covers the students' responses to questions and that every associate understands and applies it in the same correct way. As preparation for standardisation each associate analyses a number of students' scripts. Alternative answers not already covered by the mark scheme are discussed and legislated for. If, after the standardisation process, associates encounter unusual answers which have not been raised they are required to refer these to the Lead Examiner.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of students' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this mark scheme are available from aqa.org.uk

The following annotation is used in the mark scheme:

; - means a single mark

// - means alternative response

/ - means an alternative word or sub-phrase

A - means acceptable creditworthy answer. Also used to denote a valid answer that goes beyond the expectations of the GCSE syllabus.

R - means reject answer as not creditworthy

NE - means not enough

I - means ignore

DPT - in some questions a specific error made by a candidate, if repeated, could result in the candidate failing to gain more than one mark. The DPT label indicates that this mistake should only result in a candidate losing one mark on the first occasion that the error is made. Provided that the answer remains understandable, subsequent marks should be awarded as if the error was not being repeated.

Level of response marking instructions

Level of response mark schemes are broken down into levels, each of which has a descriptor. The descriptor for the level shows the average performance for the level. There are marks in each level.

Before you apply the mark scheme to a student's answer read through the answer and annotate it (as instructed) to show the qualities that are being looked for. You can then apply the mark scheme.

Step 1 Determine a level

Start at the lowest level of the mark scheme and use it as a ladder to see whether the answer meets the descriptor for that level. The descriptor for the level indicates the different qualities that might be seen in the student's answer for that level. If it meets the lowest level then go to the next one and decide if it meets this level, and so on, until you have a match between the level descriptor and the answer. With practice and familiarity you will find that for better answers you will be able to quickly skip through the lower levels of the mark scheme.

When assigning a level you should look at the overall quality of the answer and not look to pick holes in small and specific parts of the answer where the student has not performed quite as well as the rest. If the answer covers different aspects of different levels of the mark scheme you should use a best fit approach for defining the level and then use the variability of the response to help decide the mark within the level, ie if the response is predominantly level 3 with a small amount of level 4 material it would be placed in level 3 but be awarded a mark near the top of the level because of the level 4 content.

Step 2 Determine a mark

Once you have assigned a level you need to decide on the mark. The descriptors on how to allocate marks can help with this. The exemplar materials used during standardisation will help. There will be an answer in the standardising materials which will correspond with each level of the mark scheme. This answer will have been awarded a mark by the Lead Examiner. You can compare the student's answer with the example to determine if it is the same standard, better or worse than the example. You can then use this to allocate a mark for the answer based on the Lead Examiner's mark on the example.

You may well need to read back through the answer as you apply the mark scheme to clarify points and assure yourself that the level and the mark are appropriate.

Indicative content in the mark scheme is provided as a guide for examiners. It is not intended to be exhaustive and you must credit other valid points. Students do not have to cover all of the points mentioned in the Indicative content to reach the highest level of the mark scheme.

An answer which contains nothing of relevance to the question must be awarded no marks.

Question	Part	Marking guidance	Total marks
01	1	**Mark is for AO2 (apply)** **D** 4; **If more than one lozenge shaded then mark is not awarded**	1
01	2	**Mark is for AO2 (apply)** **D** `'computer sciencegcse'`; **If more than one lozenge shaded then mark is not awarded**	1
01	3	**Mark is for AO2 (apply)** **C** `'sci'`; **If more than one lozenge shaded then mark is not awarded**	1
01	4	**Mark is for AO2 (apply)** **C** 101; **If more than one lozenge shaded then mark is not awarded**	1
02	1	**Mark is for AO1 (understanding)** **C** Only two of the examples of code are in low-level languages; **If more than one lozenge shaded then mark is not awarded**	1
02	2	**4 marks for AO1 (understanding)** Maximum four marks from: • High-level languages have built-in functions; • High-level languages have built-in libraries; • High-level languages have more support/help; • High-level languages have structures (such as selection and iteration); • High-level languages can be less machine dependent/more portable; • It (usually) requires fewer lines of code to be written; • It is (usually) quicker to develop code in high-level languages; • It is easier to find mistakes in code; • The code is easier to maintain//understand; • It is easier to structure code in high-level languages; **NE.** references to efficiency or speed unless correctly qualified; **A.** Easier to read in place of easier to understand on this occasion; **R.** Answers relating to programmer expertise;	4

Question	Part	Marking guidance	Total marks
02	3	**2 marks for AO1 (understanding)** [Statement A:] compiler; [Statement B:] assembler;	2
03	1	**Mark is for AO2 (apply)** **C** `flourNeeded ← eggsUsed * 100;` **If more than one lozenge shaded then mark is not awarded**	1
03	2	**Mark is for AO2 (apply)** **A** Assignment; **If more than one lozenge shaded then mark is not awarded**	1
03	3	**4 marks for AO3 (program)** Max 3 marks if the answer contains any errors. **1 mark (A)** Indefinite iteration is used; **1 mark (B)** User input is used within the iteration/validation structure **and** the result is stored in the variable eggsUsed; **2 marks (C, D)** A Boolean condition checks the lower bound of eggsUsed is greater than zero/greater than or equal to one **and** the upper bound of eggsUsed is less than or equal to eight/less than nine (even if the **structure** is incorrect). This could possibly be one expression such as `0 < eggsUsed ≤ 8;;` If condition not completely correct then: **1 mark** The Boolean condition checks the lower bound of eggsUsed is greater than zero (even if the structure is incorrect) OR The Boolean condition checks the upper bound of eggsUsed is less than or equal to eight (even if the structure is incorrect) OR The Boolean conditions for the lower and upper bound are joined with the AND operator (even if the structure or the conditions themselves are incorrect); OR A method has been used that does not use a Boolean condition but is largely clear;	

Example 4 mark answer:

```
REPEAT                                          (A)
  eggsUsed ← USERINPUT                          (B)
UNTIL eggsUsed > 0 AND eggsUsed ≤ 8             (C, D)
```

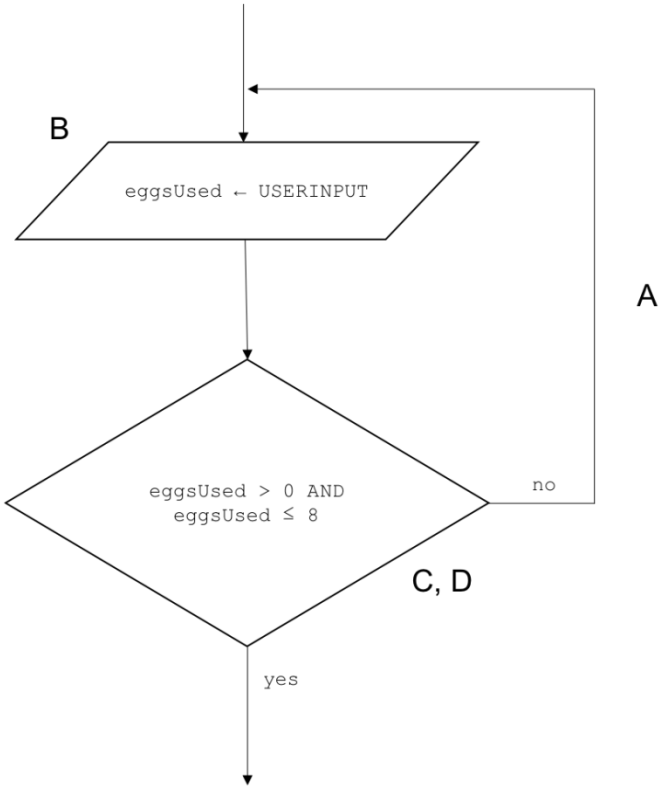
Example 4 mark answer:

```
DO                                              (A)
  eggsUsed ← USERINPUT                          (B)
WHILE eggsUsed < 1 OR eggsUsed > 8              (C, D)
```

Example 4 mark answer:

```
REPEAT                                          (A)
  eggsUsed ← USERINPUT                          (B)
UNTIL 0 < eggsUsed ≤ 8                          (C, D)
```

Example 4 mark answer:



<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>04</td><td>1</td><td>4 marks for AO2 (apply)

Mark A for <code>totalSize</code> completely correct;
Mark B for <code>dataToBeSent</code> decrementing correctly by the value given for <code>totalSize</code> until it is ≤ 0 (award even if <code>totalSize</code> is incorrect);
Mark C for <code>numberOfPackets</code> starting at 0;
Mark D for minimum of three values in the <code>numberOfPackets</code> column, incrementing by one. The number of values in the <code>dataToBeSent</code> column must match the number of values in the <code>numberOfPackets</code> column;

Correct table is:

<table>
<tr><th><code>totalSize</code></th><th><code>dataToBeSent</code></th><th><code>numberOfPackets</code></th></tr>
<tr><td>300</td><td>750</td><td>0</td></tr>
<tr><td></td><td>450</td><td>1</td></tr>
<tr><td></td><td>150</td><td>2</td></tr>
<tr><td></td><td>-150</td><td>3</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

A. follow through for incorrect <code>totalSize</code></td><td>4</td></tr>
<tr><td>04</td><td>2</td><td>Mark is for AO2 (apply)

(they are both) constants//their values do not change</td><td>1</td></tr>
<tr><td>04</td><td>3</td><td>Mark is for AO2 (apply)

A Input: <code>dataToBeSent</code>, output: <code>numberOfPackets</code>;
If more than one lozenge shaded then mark is not awarded</td><td>1</td></tr>
<tr><td>04</td><td>4</td><td>3 marks for AO3 (program)

A <code>dataToBeSent</code>;

B <code>totalSize</code>;

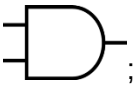
C <code>numberOfPackets + 1</code>;

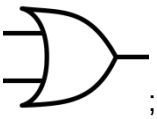
A. <code>numberOfPackets++ for C;</code>

I. case and minor spelling mistakes</td><td>3</td></tr>
</table>

Question	Part	Marking guidance	Total marks
05	1	**Mark is for AO2 (apply)** NOT;	1

05	2	**Mark is for AO2 (apply)** AND;	1

05	3	**Mark is for AO2 (apply)**  ; **I.** all labels	1

05	4	**Mark is for AO2 (apply)**  ; **I.** all labels	1

05	5	**3 marks for AO2 (apply)** 1 mark for correct column **X AND Y**; 1 mark for correct column **NOT X**; 1 mark for correct **OR** of the answers given in the column **X AND Y** and the column **NOT X** even if the answers for **X AND Y** and **NOT X** are incorrect; The correctly completed table is: **X** | **Y** | **X AND Y** | **NOT X** | **(X AND Y) OR (NOT X)** 0 | 0 | 0 | 1 | 1 0 | 1 | 0 | 1 | 1 1 | 0 | 0 | 0 | 0 1 | 1 | 1 | 0 | 1 **A.** follow through from previous columns	3

05	6	**Mark is for AO2 (apply)** **D** A1 AND A3; **If more than one lozenge shaded then mark is not awarded**	1

Question	Part	Marking guidance	Total marks
06	1	**Mark is for AO2 (apply)** **D** `USERINPUT;` **If more than one lozenge shaded then mark is not awarded**	1
06	2	**Mark is for AO2 (apply)** **B** `0;` **If more than one lozenge shaded then mark is not awarded**	1
06	3	**Mark is for AO2 (apply)** **A** `=` ; **If more than one lozenge shaded then mark is not awarded**	1
06	4	**Mark is for AO2 (apply)** **D** `OUTPUT count;` **If more than one lozenge shaded then mark is not awarded**	1
06	5	**Mark is for AO2 (apply)** **B** `i ← i + 1;` **If more than one lozenge shaded then mark is not awarded**	1
06	6	**2 marks for AO2 (apply)** **Maximum of 1 mark if Upper Case Characters given** • 1 mark for a series of more than one correct frequency/value or value/frequency pairs (ignore order of pairs); • 1 mark for all correct pairs in the correct order; Correct answer is: 2 t 2 j 3 e 2 s Other, clear ways to show frequency/value or value/frequency pairs such as '(2, t), (2, j),…' or 't2 j2…'.	2

Question	Part	Marking guidance	Total marks
06	7	**3 marks for AO2 (apply)** Maximum three marks from: • It could be tested with only 1s; • It could be tested with different lengths of input; • It could be tested with an input where the 1s and 0s vary; • It could be tested with an input where the last two numbers are different; • It could be tested with the empty string; • It could be tested with a string of length one; • It could be tested with two runs of 0s separated by a run of 1s / two runs of 1s separated by a run of 0s; • It could be tested with invalid data (such as 1010abc); Any other correct reasoning as long as clearly distinct from other mark points. **R.** not enough tests are carried out.	3
07	1	**Mark is for AO2 (apply)** **C** Selection; **If more than one lozenge shaded then mark is not awarded**	1
07	2	**Mark is for AO2 (apply)** **D** String; **If more than one lozenge shaded then mark is not awarded**	1
07	3	**Mark is for AO2 (apply)** 3//three;	1
07	4	**2 marks for AO2 (apply)** `'no'` followed by `'yes'`; any value that isn't 'no' followed by 'yes' (allow by examples such as 'yes' followed by 'yes'); **R.** if a sequence does not contain two user inputs.	2

Question	Part	Marking guidance	Total marks
07	5	**3 marks for AO2 (apply)** Maximum three marks overall. **Maximum two marks from each section.** **<u>Reason</u>** • The output message is not descriptive enough/the user is not told what word/words they should use to answer (before user input); • The Boolean expression (at lines 3, 6 and 14) only matches exact values//the program is only written for the exact words yes and no // a **clear** indication that y is not recognised as yes or n is not recognised as no; • A clear explanation of how to fix the problem; **<u>What would happen</u>** Any clear descriptions of what would happen. Line numbers may or may not be included. If the logic and explanation is clear credit the answer. This can include but is not limited to: • Line 3 will only be true if they enter 'no' // Line 3 will not be true if they enter anything other than 'no'; • Line 6/14 will only be true if they enter 'yes' // Line 6/14 will not be true if they enter anything other than 'yes'; • if they enter 'n' at line 2 the algorithm will execute an incorrect code block; • if they enter 'y' at line 5 or line 13 an incorrect message will be output;	3

08	1	**3 marks for AO2 (apply)** Stop marking at the first error. (Compare) 30 with 21/position 3; (Compare) 30 with 31/position 5; (Compare) 30 with 27/position 4;	3

08	2	**1 mark for AO1 (understanding)** (The array) must be ordered/sorted;	1

Question	Part	Marking guidance	Total marks

09		**6 marks for AO3 (program)**
		
		Any fully correct answer should get 6 marks even if it does not map exactly to the following mark points.
		
		Maximum 5 marks if the answer contains any errors.
		
		Mark A: using a selection statement in the nested `WHILE` loop;
		Mark B: using a Boolean condition that tests for equality//inequality of the `image1` and `image2` variables;
		Mark C: indexing either `image1` or `image2` using the variables `i` and `j`;
		Mark D: assigning false to inverse within the selection if logically correct throughout the code (if assigned true then check for correctness);
		Mark E: incrementing `j` in the relevant place;
		Mark F: incrementing `i` in the relevant place;
		
		Example 6 mark answer:
		
		`image1 ← [ [0, 0, 0], [0, 1, 1], [1, 1, 0] ]`
		`image2 ← [ [1, 1, 1], [1, 1, 0], [0, 0, 1] ]`
		`inverse ← true`
		`i ← 0`
		`WHILE i ≤ 2`
		`  j ← 0`
		`  WHILE j ≤ 2`
		`  IF image1[i][j] = image2[i][j] THEN (A,B,C)`
		`  inverse ← false (D)`
		`  ENDIF`
		`  j ← j + 1 (E)`
		`  ENDWHILE`
		`  i ← i + 1 (F)`
		`ENDWHILE`
			6

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>10</td><td>1</td><td>2 marks for AO1 (understanding)

Correct table is:

<table>
<tr><th>Values</th><th>Data type</th></tr>
<tr><td><code>true</code>, <code>false</code></td><td>Boolean;</td></tr>
<tr><td><code>0</code>, <code>1</code>, <code>2</code></td><td>Integer;</td></tr>
</table>

A. Bool/bool/boolean instead of Boolean
A. Int/int instead of integer</td><td>2</td></tr>
<tr><td>10</td><td>2</td><td>Mark is for AO1 (recall)

Decomposition;

A. Top-down design;</td><td>1</td></tr>
<tr><td>10</td><td>3</td><td>2 marks for AO2 (apply)

1 mark for giving a new identifier that describes this purpose, eg <code>notHit</code> (alternatively award this mark if the explanation is incorrect but the identifier describes the purpose stated in the answer);
1 mark for explaining the purpose of the subroutine is to see if a hit has been made at the specified location;</td><td>2</td></tr>
<tr><td>10</td><td>4</td><td>2 marks for AO2

(A local variable) is only accessible//declarable//within scope (in the subroutine);
(A local variable) only exists while the subroutine/program block is executing;</td><td>2</td></tr>
</table>

Question	Part	Marking guidance	Total marks

10	5	**11 marks for AO3 (program)**
		
		Any fully correct answer should get 11 marks even if it does not map exactly to the following mark points.
		
		Max 10 marks if the answer includes any errors.
		
		Mark A: for creating a subroutine with an identifier that defines its purpose;
		Mark B: for passing the board as a parameter;
		Mark C: for using iteration to loop over all (15) locations in the board;
		Mark D: for using indices (or similar) to identify the value of each cell//a FOR-EACH loop used correctly;
		Mark E: for using selection to ascertain if a cell is a hit (value 2);
		Mark F: for incrementing a variable that stores how many hits have been made;
		Mark G: for ascertaining the number of cells yet to be hit (value 1), possibly by using the subroutine F;
		Mark H: for suitable variable initialisation;
		Mark I: for outputting 'Winner' if the number yet to be hit is zero;
		Mark J: for outputting 'Almost there' if the number yet to be hit is 1–3 inclusive;
		Mark K: for outputting the Mark F variable;
		
		A. For marks I,J and K accept returning the number of hits and messages in place of outputting to the screen on this occasion only.
		
		Example of complete correct answer:
		
		`SUBROUTINE howFarAwayFromEnding(board)  [A, B]`
		`  hits ← 0  [part H]`
		`  yetToBeHit ← 0  [part H]`
		`  FOR x ← 0 TO 14  [C]`
		`  IF board[x] = 2 THEN  [D, E]`
		`  hits ← hits + 1  [F]`
		`  ELSE`
		`  IF board[x] = 1 THEN  [part G]`
		`  yetToBeHit ← yetToBeHit + 1  [part G]`
		`  ENDIF`
		`  ENDIF`
		`  ENDFOR`
		`  OUTPUT hits  [K]`
		`  IF yetToBeHit = 0 THEN  [part I]`
		`  OUTPUT 'Winner'  [part I]`
		`  ELSE IF yetToBeHit < 4 THEN  [part J]`
		`  OUTPUT 'Almost there'  [part J]`
		`  ENDIF`
		`ENDSUBROUTINE`
			11

Example of complete correct answer that uses FOREACH:

```
SUBROUTINE howFarAwayFromEnding(board)    [A, B]
   hits ← 0                               [part H]
   yetToBeHit ← 0                         [part H]
   FOREACH cell IN board                  [C, D]
      IF cell = 2 THEN.                   [E]
         hits ← hits + 1                  [F]
      ELSE
         IF cell = 1 THEN                 [part G]
            yetToBeHit ← yetToBeHit + 1   [part G]
         ENDIF
      ENDIF
   ENDFOREACH
   OUTPUT hits                            [K]
   IF yetToBeHit = 0 THEN                 [part I]
      OUTPUT 'Winner'                     [part I]
   ELSE IF yetToBeHit < 4 THEN            [part J]
      OUTPUT 'Almost there'               [part J]
   ENDIF
ENDSUBROUTINE
```

Example of complete correct answer that doesn't use Mark G variable:

```
SUBROUTINE howFarAwayFromEnding(board)    [A, B]
   hits ← 0                               [part H]
   FOR x ← 0 TO 14                        [C]
      IF board[x] = 2 THEN                [D, E]
         hits ← hits + 1                  [F]
      ENDIF
   ENDFOR
   OUTPUT hits                            [K]
   IF (6 - hits) = 0 THEN          [part G, part I]
      OUTPUT 'Winner'                     [part I]
   ELSE IF (6 - hits) < 4 THEN     [part G, part J]
      OUTPUT 'Almost there'               [part J]
   ENDIF
ENDSUBROUTINE
```