GCSE
COMPUTER SCIENCE
8520/1

Paper 1 Computational Thinking and Problem-Solving

Mark scheme

June 2021

Version: 1.0 Final

216G8520/1/MS

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.

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>1</td><td></td><td>3 marks for AO1 recall

1 mark for 1 correct label;
2 marks for 2 correct labels;
3 marks for 3 correct labels;

Correct table is:

<table>
<tr><th>Process</th><th>Label (A–F)</th></tr>
<tr><td>Breaking down a problem into sub-problems.</td><td>C</td></tr>
<tr><td>Removing unimportant details.</td><td>A</td></tr>
<tr><td>Ensuring the user enters data that is allowed, for example within a correct range.</td><td>B</td></tr>
</table>

R. all occurrences of a label entered more than once.</td><td>3</td></tr>
</table>

Qu	Pt	Marking guidance	Total marks
2	1	**Mark is for AO2** Integer/int; **A.** programming language specific data type	1

Qu	Pt	Marking guidance	Total marks
2	2	**Mark is for AO2** (The value) doesn't change/vary (after being initialised);	1

Qu	Pt	Marking guidance	Total marks
2	3	**Mark is for AO2** 3 // three; **A.** 3rd (line) // third (line);	1

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>2</td><td>4</td><td>
4 marks for AO2

1 mark for <code>seconds</code> having values 60, 120 and 180 in that order;
1 mark for the final value of <code>seconds</code> as 240;
1 mark for the first value of <code>effort</code> as 20 and the first value of <code>OUTPUT</code> as <code>'faster'</code>.
1 mark for the last three values in the <code>effort</code> column all correct and every output correct for these three values of effort;

Max 3 marks if any errors.

I. use of quote marks or minor spelling errors in the <code>OUTPUT</code> column.
I. values on different lines as long as the order is correct and no other values have been entered.

Correct table as follows:

<table>
<tr><th>seconds</th><th>bpm</th><th>effort</th><th>OUTPUT</th></tr>
<tr><td>0</td><td>70</td><td>20</td><td>faster</td></tr>
<tr><td>60</td><td>80</td><td>30</td><td>faster</td></tr>
<tr><td>120</td><td>100</td><td>50</td><td>steady</td></tr>
<tr><td>180</td><td>120</td><td>70</td><td>slower</td></tr>
<tr><td>240</td><td></td><td></td><td></td></tr>
</table>
</td><td>4</td></tr>
</table>

Qu	Pt	Marking guidance	Total marks
3	1	**2 marks for AO2** **Max two marks** from the following: (The developer has…) decomposed the problem/broken the problem down (into sub-problems); implemented sub-problems as subroutines; used interfaces (including parameters and return values);	2

Qu	Pt	Marking guidance	Total marks
3	2	**2 marks for AO1 (understanding)** **Max two marks** from the following: The subroutines will be easier to test/mistakes will be easier to find; The subroutines can be reused; The subroutines can be changed without affecting the rest of the program; The subroutines create better self-documenting code;	2

Qu	Pt	Marking guidance	Total marks
3	3	**5 marks for AO3 (program)** 1 mark for each correct label: **L1** `max ;` **L2** `getRank ;` **L3** `index ;` **L4** `c ;` **L5** `result ;`	5

Qu	Pt	Marking guidance	Total marks
3	4	**Mark is for AO1 (understanding)** **B** Unicode includes characters from many different alphabets.; **R.** if more than one lozenge is shaded.	1

Qu	Pt	Marking guidance	Total marks
4	1	**2 marks for AO2** 1 mark for each correction: (Statement 2) A 2 pixel by 4 pixel bitmap image contains 8 pixels // A *n* pixel by *m* pixel bitmap image contains 16 pixels *[where n*m=16]*; (Statement 4) Black and white images have a minimum colour depth of one // Three/Four-colour images have a minimum colour depth of two; **A.** Explanation of error that makes it clear what should have been written instead of the corrected statement.	**2**

Qu	Pt	Marking guidance	Total marks
4	2	**Mark is for AO2** 300 (b // bits);	**1**

Qu	Pt	Marking guidance	Total marks
4	3	**3 marks for AO2** 50 (B // bytes);;; If incorrect answer then award a **max of two** marks for the following working: identifying the colour depth is 4; correctly multiplying 10 x 10 x (possibly incorrect) colour depth; attempt at dividing the calculated size in bits by 8;	**3**

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>4</td><td>4</td><td>3 marks for AO2

1 mark for <code>i</code> column correct;
1 mark for one of indices <code>0</code>, <code>3</code> and <code>4</code> assigned the value 99;
1 mark for all of indices <code>0</code>, <code>3</code> and <code>4</code> (and no other indices) assigned the value 99;

Max 2 marks if any errors.

Correct table as follows:
<table>
<tr><th rowspan="2">i</th><th colspan="6">newRow</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr>
<tr><td></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>99</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>3</td><td></td><td></td><td></td><td>99</td><td></td><td></td></tr>
<tr><td>4</td><td></td><td></td><td></td><td></td><td>99</td><td></td></tr>
<tr><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>3</td></tr>
</table>

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>4</td><td>5</td><td>Mark is for AO2

Converts (row/grey scale image) to black and white // the values 0 and 99 // two colours/shades;</td><td>1</td></tr>
</table>

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>5</td><td>1</td><td>2 marks for AO1 (understanding)

1 mark for one row correct;
2 marks for all rows correct;

R. any duplicated answers

Correct table as follows:

<table>
<tr><th>Description</th><th>Label (A–C)</th></tr>
<tr><td>Converts a low-level language designed to be human-readable into machine code.</td><td>A</td></tr>
<tr><td>Reads a high-level program line-by-line and calls corresponding subroutines.</td><td>C</td></tr>
<tr><td>Takes the entire high-level program as input and produces machine code.</td><td>B</td></tr>
</table>
</td><td>2</td></tr>
</table>

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>5</td><td>2</td><td>2 marks for AO1 (understanding)

Max two from:

(High-level languages) are easier to test // identify mistakes (than low-level languages);
(High-level languages) allow faster development (than low-level languages);
(High-level languages) are better documented (than low-level languages);
(High-level languages) contain complex data structures;
(High-level languages) allow code to be more portable (than low-level languages);

Refer other plausible answers to Team Leader.</td><td>2</td></tr>
</table>

Qu	Pt	Marking guidance	Total marks
5	3	**9 marks for AO3 (programming)** **[Mark A]** for getting user input and assigning it to a variable; **[Mark B]** for using selection to check for the length of user input (even if the Boolean condition is incorrect); **[Mark C]** for a correct Boolean condition to check that the length is 8 (or not 8 if opposite logic used) even if not within a selection structure; ***[Mark D]** for iterating over the instruction to check for (in)correct characters; ***[Mark E]** for the iteration structure in **Mark D** isolating every character in the string (even if the subsequent check for validity is incorrect); **[Mark F]** for using selection to check if a character is/is not `'0'` or `'1'`; **[Mark G]** for a correct Boolean condition checking the character is/is not a `'0'` and/or a `'1'`; **[Mark H]** outputting `'ok'` and `'wrong'` based on the length of the user input. **[Mark I]** outputting `'ok'` and `'wrong'` based on the characters in the user input. ***A. alternative method for obtaining Mark D and Mark E** **[Mark D]** eight selection structures instead of iteration; **[Mark E]** ensure every character is checked in **Mark D**; **Max 8 marks if any errors.**	9

An example of a completely correct solution:

```
instruction ← USERINPUT                          [A]
valid ← True                                [Part H, Part I]
IF LEN(instruction) ≠ 8 THEN                     [B, C]
   valid ← False                                [Part H]
ELSE
   FOR i ← 0 TO 7                                [D, E]
      IF instruction[i] ≠ '0' AND                [F, G]
                instruction[i] ≠ '1' THEN
         valid ← False                           [Part I]
      ENDIF
   ENDFOR
ENDIF
IF valid = True THEN
   OUTPUT 'ok'                           [Part H, Part I]
ELSE
   OUTPUT 'wrong'                         [Part H, Part I]
ENDIF
```

Another example of a completely correct solution:

```
instruction ← USERINPUT                          [A]
IF LEN(instruction) = 8 THEN                     [B, C]
   i ← 0                                        [Part E]
   valid ← True
   WHILE i < 8                                   [D, Part E]
      IF instruction[i] ≠ '0' THEN      [Part F, Part G]
         IF instruction[i] ≠ '1' THEN   [Part F, Part G]
            valid ← False
         ENDIF
      ENDIF
      i ← i + 1                                  [Part E]
   ENDWHILE
   IF valid = True THEN
      OUTPUT 'ok'                           [Part H, Part I]
   ELSE
      OUTPUT 'wrong'                             [Part I]
   ENDIF
ELSE
   OUTPUT 'wrong'                                [Part H]
ENDIF
```

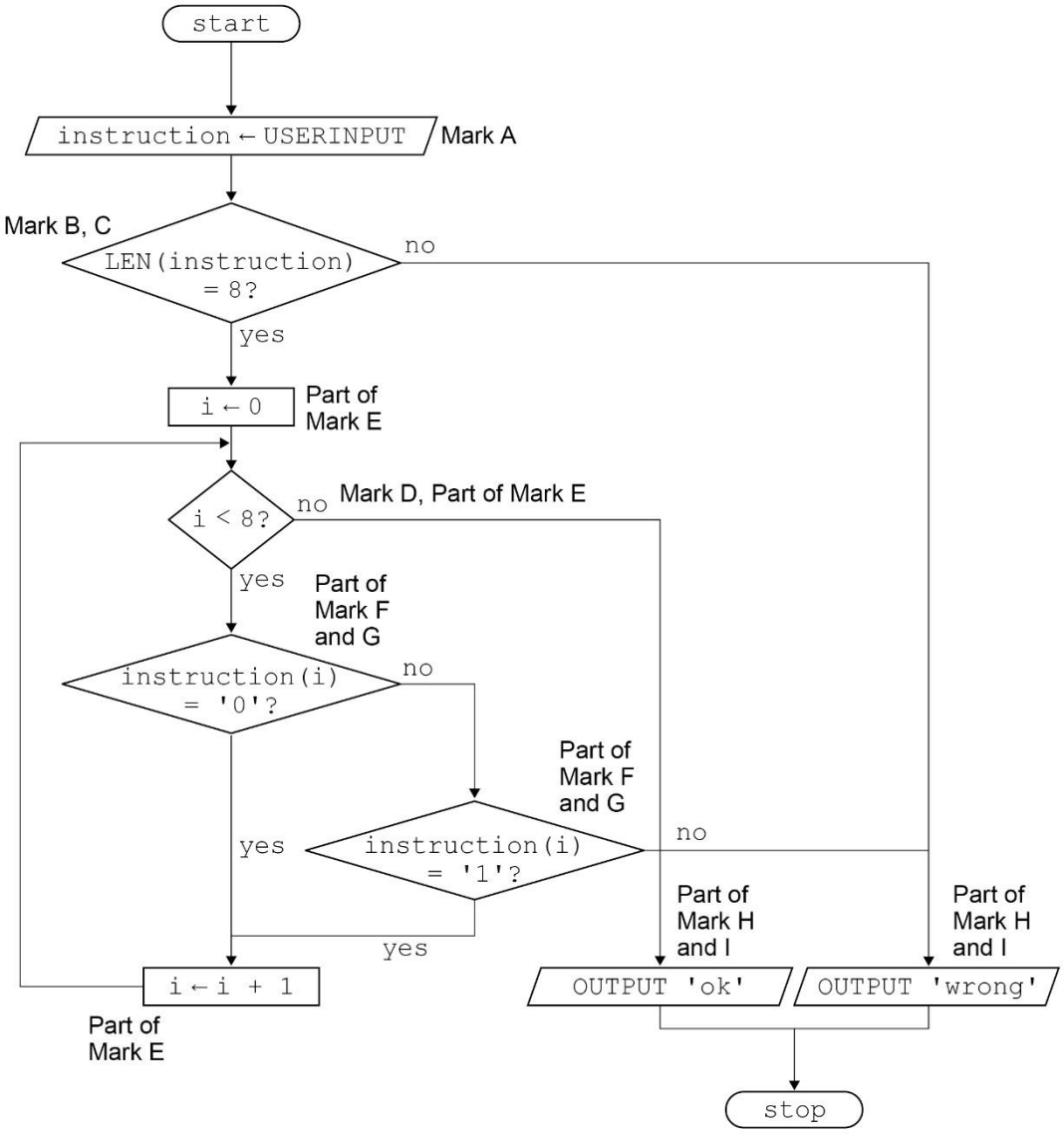
Another example of a completely correct solution:

```
instruction ← USERINPUT                          [A]
IF LEN(instruction) = 8 THEN                     [B, C]
   valid ← True
   FOR i ← 0 TO 7                               [D, E]
      IF instruction[i] ≠ '0' THEN     [Part F, Part G]
         IF instruction[i] ≠ '1' THEN  [Part F, Part G]
            valid ← False
         ENDIF
      ENDIF
   ENDFOR
   IF valid = True THEN
      OUTPUT 'ok'                          [Part H, Part I]
   ELSE
      OUTPUT 'wrong'                            [Part I]
   ENDIF
ELSE
   OUTPUT 'wrong'                               [Part H]
ENDIF
```

A final example of a completely correct solution that uses a FOR-EACH style loop to iterate over the characters of the string (note that this is not part of the AQA pseudo-code supplement but still perfectly acceptable):

```
instruction ← USERINPUT                           [A]
valid ← True                                [Part H, Part I]
IF LEN(instruction) ≠ 8 THEN                      [B, C]
   valid ← False                                 [Part H]
ELSE
   FOR ch IN instruction                          [D, E]
      IF ch ≠ '0' AND ch ≠ '1' THEN               [F, G]
         valid ← False                            [Part I]
      ENDIF
   ENDFOR
ENDIF
IF valid = True THEN
   OUTPUT 'ok'.                          [Part H, Part I]
ELSE
   OUTPUT 'wrong'                        [Part H, Part I]
ENDIF
```

An example of a fully correct flowchart solution is:



I. shape of symbols

Qu	Pt	Marking guidance	Total marks
6	1	**3 marks for AO2** (The value 8 is compared to the value) 4; **R.** if not first comparison (The value 8 is compared to the value) 7; **R.** if not second comparison (The value 8 is compared to the value) 8; **R.** if not third comparison Alternatively: (The value 8 is compared to the) first element of the array; (The value 8 is compared to) every subsequent value of the array; When the value 8 is found in the array it returns True;	3

Qu	Pt	Marking guidance	Total marks
6	2	**3 marks for AO2** (The value 8 is compared to) 13; **R.** if not first comparison (The value 8 is compared to) 7; **R.** if not second comparison (The value 8 is compared to) 8; **R.** if not third comparison Alternatively: (The value 8 is compared to the) midpoint of the array; (The value 8 is compared to the) midpoint of the left subarray ([4, 7, 8]); (The value 8 is compared to the) midpoint of the right subarray ([8]);	3

Qu	Pt	Marking guidance	Total marks
6	3	**Mark is for AO1 (understanding)** It is more efficient // requires fewer steps/comparisons (on average);	1

<table>
<tr><th>Qu</th><th>Pt</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>6</td><td>4</td><td>
4 marks for AO2

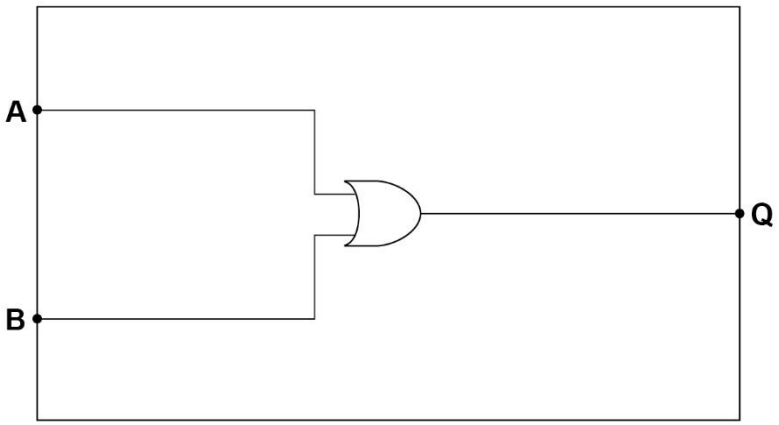
1 mark for values 4 and 8 in the <code>i</code> column (in that order);
1 mark for value 7 as the last value in the <code>i</code> column;
1 mark for <code>down</code> being set only once to <code>True</code>;
1 mark for <code>finished</code> being set only once to <code>True</code>;

Max 3 marks if any errors.

I. repeated values written in columns
I. exact placing of values as long as the vertical order is correct

Correct table as follows:

<table>
<tr><th>found</th><th>finished</th><th>i</th><th>down</th></tr>
<tr><td>False</td><td>False</td><td>0</td><td>False</td></tr>
<tr><td></td><td></td><td>4</td><td></td></tr>
<tr><td></td><td></td><td>8</td><td>True</td></tr>
<tr><td></td><td>True</td><td>7</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td></tr>
</table>
</td><td>4</td></tr>
</table>

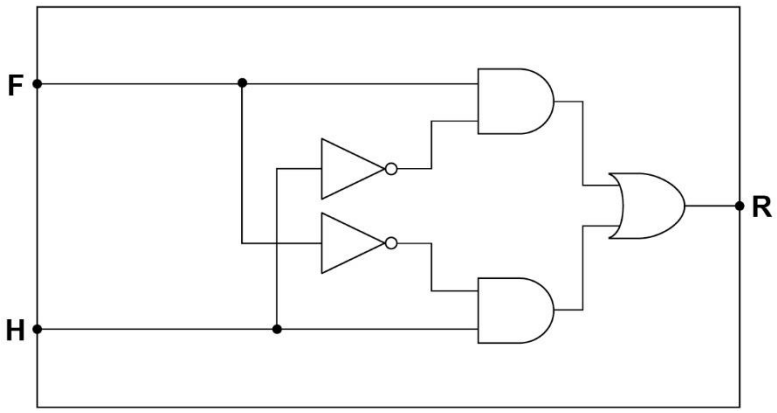
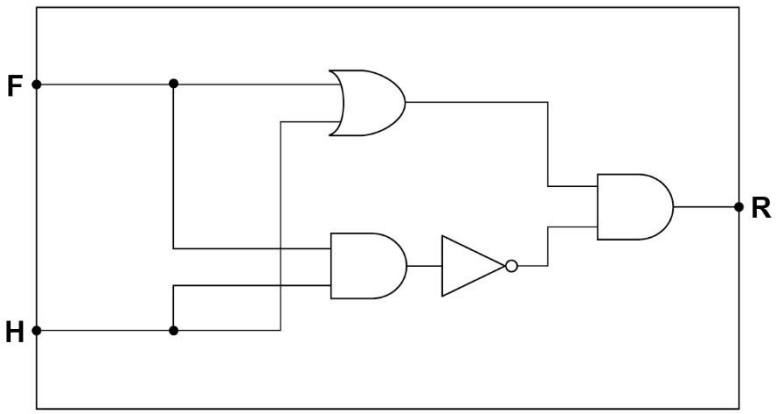
Qu	Pt	Marking guidance	Total marks
7	1	**Mark is for AO2**  Answers **must** show two inputs and one output from the gate. **I.** any labels on inputs or output.	**1**

Qu	Pt	Marking guidance	Total marks
7	2	**Mark is for AO2** **C** `(NOT A) AND B;` **R.** If more than one lozenge shaded	**1**

Qu	Pt	Marking guidance	Total marks
7	3	**Mark is for AO2** **D** `((NOT A) OR (NOT B)) AND C;` **R.** If more than one lozenge shaded	**1**

Qu	Pt	Marking guidance	Total marks

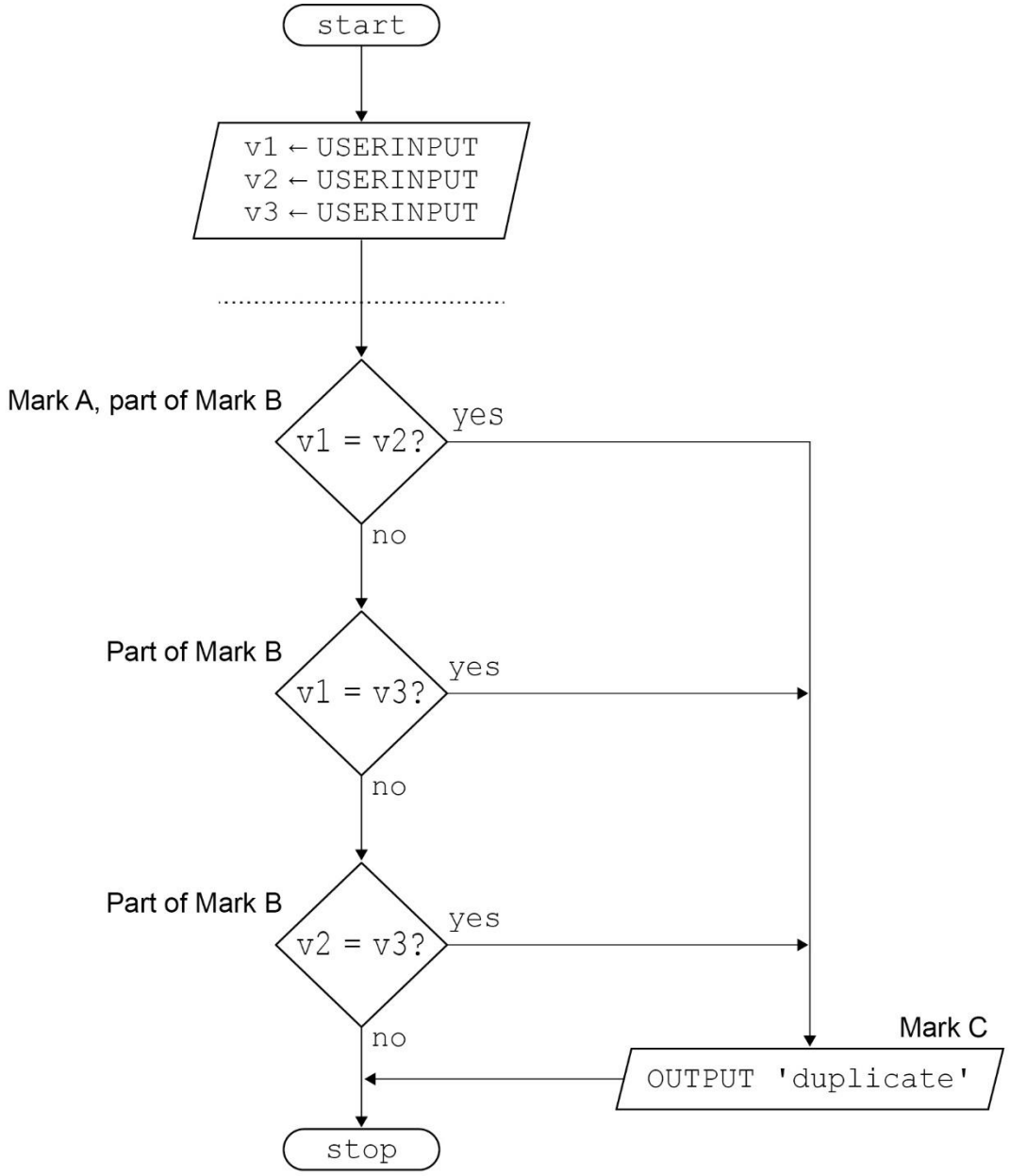
7	4	**2 marks for AO2**
		
		`A;`
		`C;`
		
		I. use of quote marks
		I. if answers are on the same line or different lines as long as order is clear
		R. if more than two characters are stated
			2

Qu	Pt	Marking guidance	Total marks
7	5	**3 marks for AO2** 1 mark for using gates to implement (F and NOT H); 1 mark for using gates to implement (H and NOT F); 1 mark for an OR gate having two inputs, whose output is R; **Max 2 marks if any errors.** **Award full marks if a different, but logically correct, solution is given using only AND, OR or NOT gates.** **Max 1 mark if gates other than AND, OR or NOT are used**. This is an example of a correct logic circuit:  Another example of a correct logic circuit:  	**3**

Qu	Pt	Marking guidance	Total marks

7	6	**3 marks for AO3 (programming)**
		
		[Mark A] for using selection to compare at least two of the three values;
		[Mark B] for using selection to compare all pairwise combinations of the three values;
		[Mark C] for output of `'duplicate'` when their conditions are met, even if the conditions contain errors;
		
		Max 2 marks if any errors.
		
		I. if student has re-written the start of the algorithm using the same or different variable identifiers to the question stem, however, their answer must be consistent.
		
		An example of a fully correct answer:
		v1 ← *USERINPUT*
		v2 ← *USERINPUT*
		v3 ← *USERINPUT*
		IF v1 = v2 THEN [A, part B]
		OUTPUT 'duplicate' [part C]
		ELSE
		IF v1 = v3 THEN [part B]
		OUTPUT 'duplicate' [part C]
		ELSE
		IF v2 = v3 THEN [part B]
		OUTPUT 'duplicate' [part C]
		ENDIF
		ENDIF
		ENDIF
		
		An alternative fully correct answer is:
		
		v1 ← *USERINPUT*
		v2 ← *USERINPUT*
		v3 ← *USERINPUT*
		IF v1 = v2 OR v1 = v3 OR v2 = v3 THEN[A, B]
		OUTPUT 'duplicate' [C]
		ENDIF
			3

An example of a fully correct flowchart answer is:



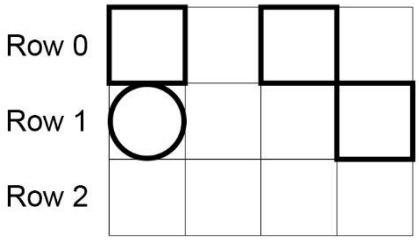
I. shape of symbols

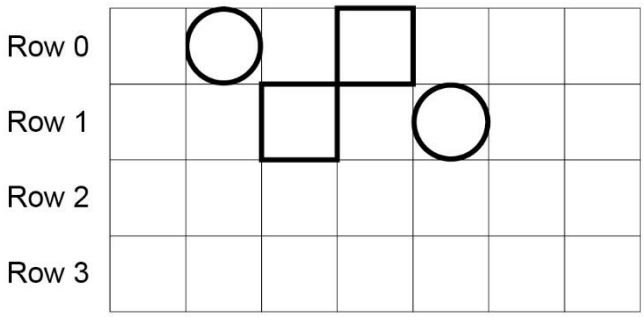
<table>
<tr><th>Qu</th><th>Pt</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>8</td><td></td><td>
3 marks for AO3 (programming)

1 mark for 1 correct position;
2 marks for 2 correct positions;
3 marks for 4 correct positions;

R. Any position which is used more than once

<table>
<tr><th>Line of code</th><th>Position (1–4 where 1 is the first line)</th></tr>
<tr><td>t ← t - 1</td><td>3</td></tr>
<tr><td>n ← t - n</td><td>4</td></tr>
<tr><td>n ← 2</td><td>1</td></tr>
<tr><td>t ← n LEFTSHIFT 3</td><td>2</td></tr>
</table>
</td><td>3</td></tr>
</table>

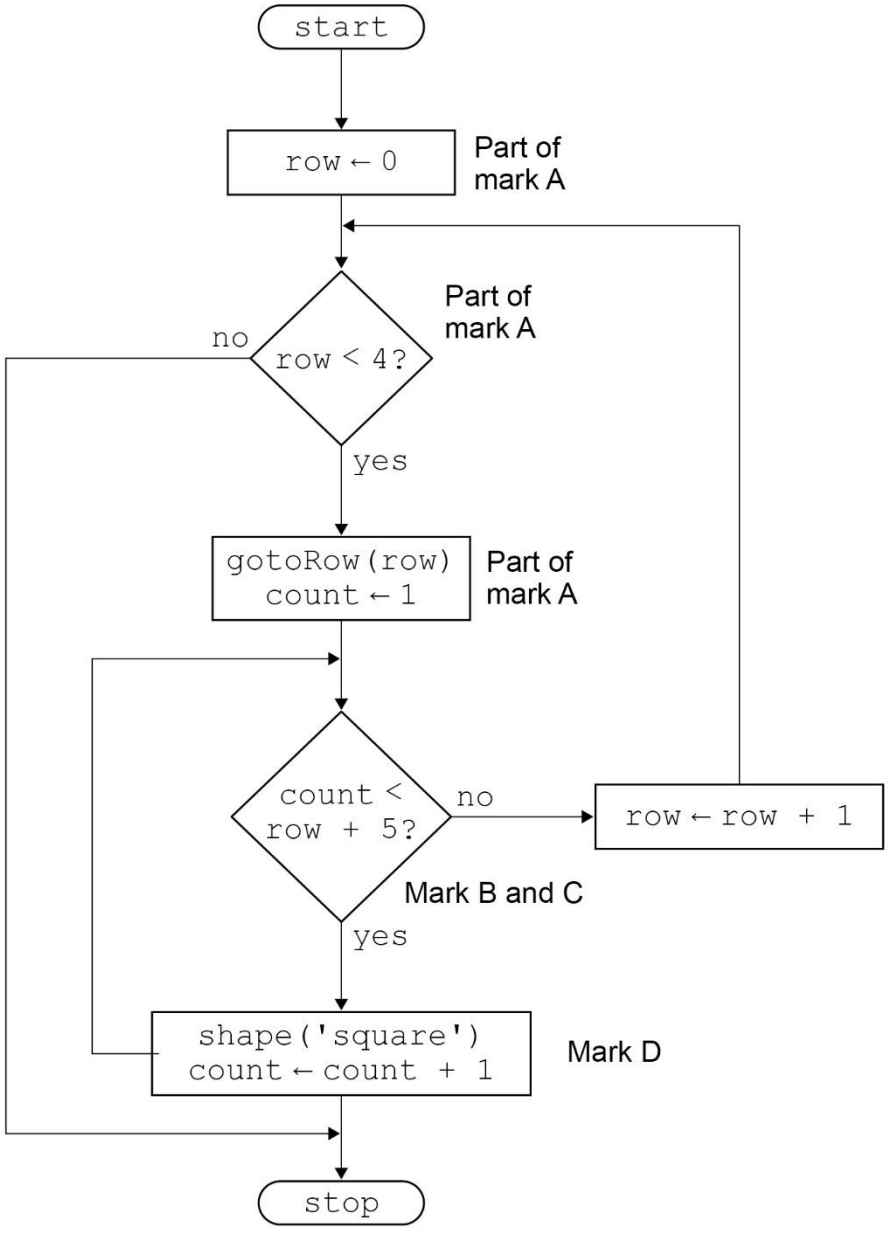
Qu	Pt	Marking guidance	Total marks
9	1	**4 marks for AO2** 1 mark for drawing one square at the start of row 0; 1 mark for the remaining 3 cells of row 0 correct; 1 mark for drawing a circle at the start of row 1; 1 mark for the remaining 3 cells of row 1 correct; **I.** any marks that indicate the position of the needle. **Max 3 marks if any errors**. The completed pattern is as follows:  	4

Qu	Pt	Marking guidance	Total marks
9	2	**4 marks for AO2** 1 mark for drawing a circle in the second cell of row 0 and having no shape in the first cell of row 0; 1 mark for drawing exactly four shapes; (If more than 4 shapes are drawn, stop marking) 1 mark for drawing a square in the third column of row 1; 1 mark for drawing a square in the fourth column and a circle in the fifth column; **I.** any marks that indicate the position of the needle. **Max 3 marks if any errors.** The completed pattern is:  	4

Qu	Pt	Marking guidance	Total marks

9	3	**4 marks for AO3 (programming)**
		
		[Mark A] use of the `gotoRow` subroutine with parameters 0, 1, 2 and 3;
		
		[Mark B] use of `shape('square')` to draw four squares in row 0;
		
		[**Mark C]** use of iteration to repeatedly draw the squares;
		
		[Mark D] correct squares drawn in rows 1, 2 and 3;
		
		Max 3 marks if any errors.
		
		An example of a fully correct answer:
		
		`squares ← 4 [Part B, Part D]`
		`FOR row ← 0 TO 3 [Part D]`
		`  gotoRow(row) [A]`
		`  FOR x ← 1 TO squares [Part B, C]`
		`  shape('square') [Part B, Part D]`
		`  ENDFOR`
		`  squares ← squares + 1 [Part D]`
		`ENDFOR`
		
		Another example of a fully correct answer:
		
		`gotoRow(0) [Part A]`
		`FOR x ← 1 TO 4 [Part B, C]`
		`  shape('square') [Part B]`
		`ENDFOR`
		`gotoRow(1) [Part A]`
		`FOR x ← 1 TO 5 [Part D]`
		`  shape('square') [Part D]`
		`ENDFOR`
		`gotoRow(2) [Part A]`
		`FOR x ← 1 TO 6 [Part D]`
		`  shape('square') [Part D]`
		`ENDFOR`
		`gotoRow(3) [Part A]`
		`FOR x ← 1 TO 7 [Part D]`
		`  shape('square') [Part D]`
		`ENDFOR`
		
		Another example of a fully correct answer:
		
		`FOR row ← 0 TO 3 [Part A]`
		`  gotoRow(row) [Part A]`
		`  FOR count ← 1 TO row + 4. [Part B, C]`
		`  shape('square') [Part B, D]`
		`  ENDFOR`
			4

		ENDFOR	

		An example of a fully correct flowchart answer:  **I.** shape of symbols	

An example of a partially correct solution is:

```
gotoRow(0)
shape('square')
shape('square')
shape('square')
shape('square')
gotoRow(1)
shape('square')
shape('square')
shape('square')
shape('square')
shape('square')
gotoRow(2)
shape('square')
shape('square')
shape('square')
shape('square')
shape('square')
shape('square')
gotoRow(3)
shape('square')
shape('square')
shape('square')
shape('square')
shape('square')
shape('square')
shape('square')
```

This solution gets marks A, B and D but not mark C as there is no use of iteration.