AQA

GCSE
Computer Science

8520/1-Paper 1 Computational thinking and problem-solving
Mark scheme

June 2018

Version/Stage: 1.0 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 Assessment Writer.

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>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>01</td><td>1</td><td>2 marks for AO1 (recall)

A sequence/number/set of steps/instructions;
that can be followed to complete a task/to solve a problem;

A. Different wording with similar meaning</td><td>2</td></tr>
<tr><td>01</td><td>2</td><td>3 marks for AO1 (recall)

One mark for each correct distinct label.

If the answers given were, for example, C, C, B then award only 1 mark for the B as the C is duplicated. Likewise if C, C, C was the answer then no marks would be given. The correct table is:

<table><tr><th></th><th>Label</th></tr><tr><td>Breaking a problem down into a number of sub-problems.</td><td>C</td></tr><tr><td>The process of removing unnecessary detail from a problem.</td><td>A</td></tr><tr><td>Defines the sort of values a variable may take.</td><td>B</td></tr></table>
A. If actual terms are written out instead of labels
R. All instances of duplicate labels</td><td>3</td></tr>
<tr><td>02</td><td>1</td><td>Mark is for AO1 (understanding)

A (Line number 2) only;
If more than one lozenge shaded then mark is not awarded</td><td>1</td></tr>
<tr><td>02</td><td>2</td><td>Mark is for AO1 (understanding)

C (Line number 11) only;
If more than one lozenge shaded then mark is not awarded</td><td>1</td></tr>
<tr><td>02</td><td>3</td><td>Mark is for AO2 (apply)

A (1 subroutine call) only;
If more than one lozenge shaded then mark is not awarded</td><td>1</td></tr>
<tr><td>02</td><td>4</td><td>Mark is for AO2 (apply)
B (String) only;
If more than one lozenge shaded then mark is not awarded;</td><td>1</td></tr>
</table>

Question	Part	Marking guidance	Total marks
02	5	**Mark is for AO2 (apply)** 2//twice//two; **I.** Minor spelling errors	1
02	6	**Mark is for AO2 (apply)** 2//two; **A.** true and false (or other possible indicators for true and false) **R.** Boolean	1
02	7	**Mark is for AO2 (apply)** 7; **A.** All of 3, 5 **and** 11 **A.** If instruction written out (a ← 2)	1
02	8	**Mark is for AO3 (program)** `q ← 2;` **A**. `a ← 1, a ← 4` **and** `FOR n ← 1 TO a` (only if all given)	1
03	1	**Mark is for AO2 (apply)** 11000; **I.** leading zeroes	1
03	2	**Mark is for AO2 (apply)** It multiplies the number by 8//by two three times// by 2^3// multiply by 1000_2; **NE** It multiplies the number	1
03	3	**Mark is for AO2 (apply)** It will halve the number//Divide the number by two;	1

Question	Part	Marking guidance	Total marks
04	1	**2 marks for AO1 (understanding)** Maximum of 2 from: Computer systems use binary/ones and zeros/voltage on or off; Sound is analogue/continuous/wave; Computers use discrete values;	2
04	2	**4 marks for AO2 (apply)** 4 marks if answer is correct 5,000 bytes/5,000B/5 kB;;;; **A.** 5,000 If answer given is not 5,000 bytes then award working marks as follows: Mark A for multiplying any two of 2,000, 4 and 5 even if the result is incorrect; Mark B for multiplying all of 2,000, 4 and 5 even if the result is incorrect; Mark C for attempting to divide the result of a calculation by 8; Partially correct examples: Example 1 2,000 * 4 = 8,000; (Mark A) 8,000 / 8 = 1,000; (Mark C) Example 2 2,000 * 4 * 5 = 20,000;; (Mark A and Mark B, note result is incorrect) 20,000 / 8 = 2,000; (Mark C, note result is incorrect)	4
04	3	**Mark is for AO2 (apply)** **B** (5 bits) only; **If more than one lozenge shaded then mark is not awarded**	1
04	4	**Mark is for AO2 (apply)** **D** (improves the quality of the recording and increases the file size.) only; **If more than one lozenge shaded then mark is not awarded**	1

Question	Part	Marking guidance	Total marks

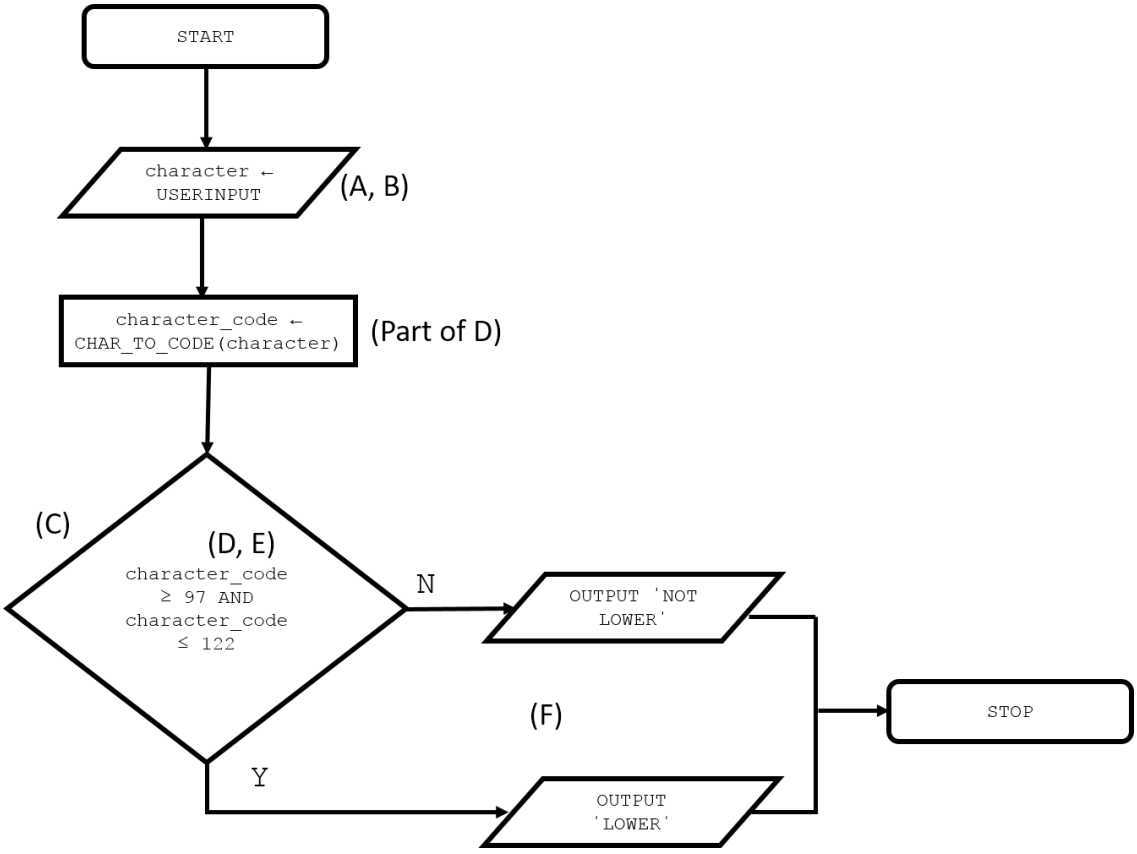
05		**7 marks for AO3 (program)**
		
		If CHAR_TO_CODE is not used then a maximum of 6 marks.
		
		Mark A for using user input;
		Mark B for storing the result of user input in a variable or using the user input directly as a parameter to CHAR_TO_CODE;
		Mark C for using selection to determine if character is lowercase or otherwise;
		Mark D for using a Boolean expression that uses CHAR_TO_CODE with the input parameter being the user input (either directly or when stored in a variable);
		Mark E for a Boolean expression that checks if the character code is between 97 and 122 (97+25) **inclusive**;
		Mark F for outputting LOWER and NOT LOWER in logically separate places such as the IF and ELSE part of selection;
		
		Mark G if the algorithm is completely correct;
		
		A. LOWER and NOT LOWER stated in lower case for Mark F.
		A. Any logically equivalent Boolean expression for Mark E.
		A. Minor errors in spelling if the meaning is clear.
		
		Example 1 (fully correct)
		
		`character ← USERINPUT` (A, B)
		`character_code ← CHAR_TO_CODE(character)` (Part of D)
		`IF character_code ≥ 97 AND character_code ≤ 122 THEN` (C, D, E)
		`  OUTPUT 'LOWER'` (Part of F)
		`ELSE`
		`  OUTPUT 'NOT LOWER'` (Part of F)
		`ENDIF`
		(G awarded as completely correct)
		
		Example 2 (fully correct)
		
		`character_code ← CHAR_TO_CODE(USERINPUT)` (A, B, Part of D)
		`IF character_code < 97 OR character_code > 122 THEN` (C, D, E)
		`  OUTPUT 'NOT LOWER'` (Part of F)
		`ELSE`
		`  OUTPUT 'LOWER'` (Part of F)
		`ENDIF`
		(G awarded as completely correct)
			7

Example 3 (fully correct)

```
character ← USERINPUT                          (A, B)
character_code ← CHAR_TO_CODE(character)       (Part of D)
IF 97 ≤ character_code ≤ 122 THEN              (C, D, E)
   OUTPUT 'LOWER'                              (Part of F)
ELSE
   OUTPUT 'NOT LOWER'                          (Part of F)
ENDIF
```

(G awarded as completely correct)

Example 4 (fully correct)



(G awarded as completely correct)

Example 5 (6 marks)

```
IF CHAR_TO_CODE(USERINPUT) ≥ 97 AND
   CHAR_TO_CODE(USERINPUT) ≤ 122 THE            (A, B, C, D, E)
   OUTPUT 'LOWER'                               (Part of F)
ELSE
   OUTPUT 'NOT LOWER'                           (Part of F)
ENDIF
```

(G not awarded as `USERINPUT` used twice)

Example 6 (6 marks)

```
character_code ← CHAR_TO_CODE(USERINPUT)        (A, B, Part of D)
IF character_code < 97 OR character_code > 122 THEN (C, D, E)
   OUTPUT 'LOWER'                               (Part of F)
ELSE
   OUTPUT 'NOT LOWER'                           (Part of F)
ENDIF
```

(G not awarded as `LOWER` and `NOT LOWER` are in the wrong places)

Question	Part	Marking guidance	Total marks
06	1	**Mark is for AO2 (apply)** Boolean//bool; **I.** Minor spelling mistakes	1
06	2	**2 marks for AO2 (apply)** (The identifier) `sorted` describes the purpose//role//meaning of the variable; this makes the algorithm easier to understand//maintain//follow; or (The identifier) `s` does not describe the purpose//role//meaning of the variable; this makes the algorithm harder to understand//maintain//follow;	2
06	3	**Mark is for AO2 (apply)** **A** (The algorithm uses a named constant.) only; **If more than one lozenge shaded then mark is not awarded**	1
06	4	**6 marks for AO2 (apply)** 1 mark for column `arr[0]` correct; 1 mark for column `arr[1]` correct; 1 mark for column `arr[2]` correct **only if** `arr[0]` and `arr[1]` are correct; 1 mark for `sorted` column correct; 1 mark for `i` column correct; 1 mark for `t` column correct; **I.** different rows used as long as the order within columns is clear **I.** duplicate values on consecutive rows within a column	6

Arr			sorted	i	t
0	1	2			
4	1	6	false		
			true	0	
1	4		false		4
				1	
				2	
			true	0	
				1	
				2	

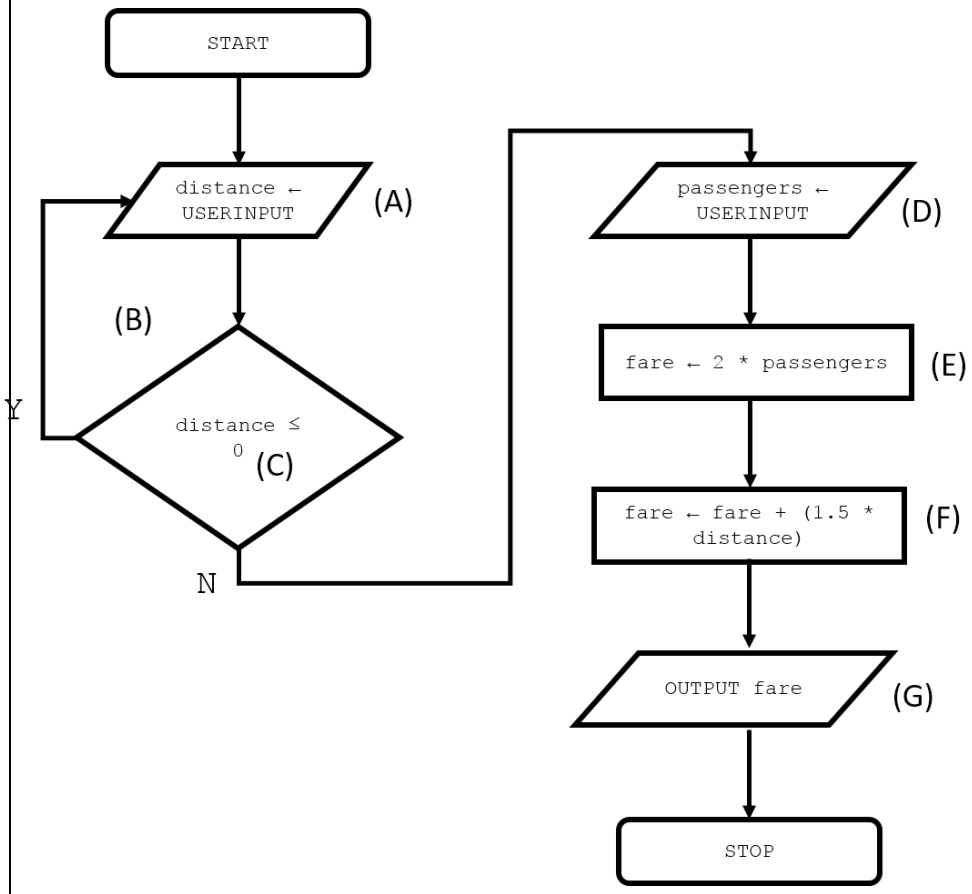
Question	Part	Marking guidance	Total marks
06	5	**3 marks for AO2 (apply)** 1 mark if pairwise comparisons are made in the second row but allow for one pairwise comparison error; 1 mark if pairwise comparisons are made in the third row but allow for one pairwise comparison error (allow follow through from previous row); 1 mark if all correct; 7 3 4 1 2 8 5 6 **3 7 1 4 2 8 5 6** **1 3 4 7 2 5 6 8** 1 2 3 4 5 6 7 8	3

06	6	**Mark is for AO1 (understanding)** It is more (time) efficient// It will usually take fewer steps; **A.** quicker//it will take less time as long as the answer has been qualified.	1

06	7	**2 marks for AO2 (apply)** Maximum of 2 from: It allows the code to be (more easily) reused; It can be used to sort any array (not just the one on line 1); It would be easier to test; The code could be changed//updated without affecting the overall program; Makes the program easier to read//understand; **A.** Any other creditable answer as long as they are clearly distinct from the other responses.	2

Question	Part	Marking guidance	Total marks
07		**8 marks for AO3 (program)** **DPT.** For repeated errors in user input and variable assignment. Mark A for getting user input for the distance and storing in a variable; Mark B for using a WHILE loop or similar to re-prompt for and re-assign the user input; Mark C for using a correct Boolean condition with the validation structure; Mark D for getting user input for the passengers; Mark E for a fare that charges £2 per passenger; Mark F for a fare that charges £1.50 for every kilometre; Mark G for outputting the fare based on E and F (Even if E and/or F have been calculated incorrectly); Mark H if the algorithm is completely correct; **Example 1 (fully correct)** `distance ← USERINPUT` (A) `WHILE distance ≤ 0` (Part of B, C) `  distance ← USERINPUT` (Part of B) `ENDWHILE` `passengers ← USERINPUT` (D) `fare ← 2 * passengers` (E) `fare ← fare + (1.5 * distance)` (F) `OUTPUT fare` (G) (Mark H as completely correct) **Example 2 (fully correct)** `REPEAT` (Part of B) `  distance ← USERINPUT` (A, Part of B) `UNTIL distance > 0` (C) `fare ← (2 * USERINPUT) + (1.5 * distance)` (D, E, F) `OUTPUT fare` (G) (Mark H as completely correct) **Example 3 (fully correct)** `DO` (Part of B) `  distance ← USERINPUT` (A, Part of B) `WHILE NOT (distance > 0)` (C) `fare ← (2 * USERINPUT) + (1.5 * distance)` (D, E, F) `OUTPUT fare` (G) (Mark H as completely correct)	8

Example 4 (fully correct)



(Mark H as completely correct)

Example 5 (7 marks)

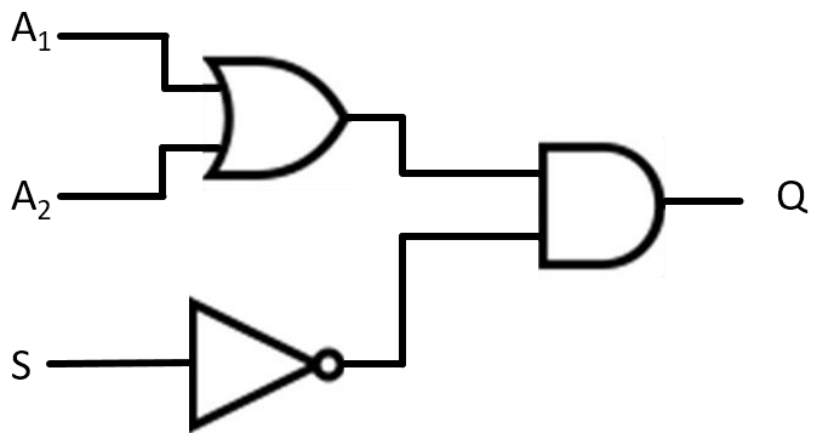
```
distance ← USERINPUT                    (A)
WHILE distance ≤ 0                      (C)
   distance ← USERINPUT                 (Part of B)
ENDWHILE
passengers ← USERINPUT                  (D)
fare ← 2 * passengers                   (E)
fare ← 1.5 * distance                   (F)
OUTPUT fare                             (G)
```

(Mark H not awarded as the final fare does not include the cost of `2 * passengers`)

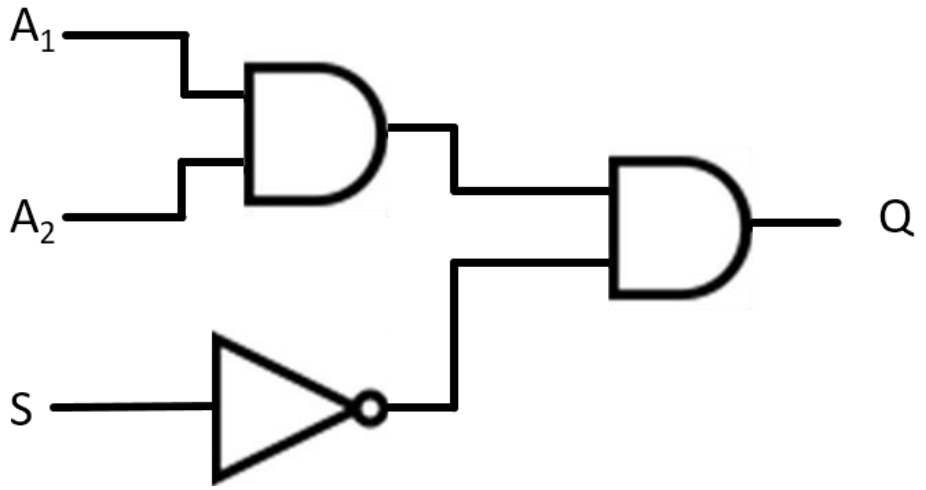
		Example 6 (5 marks)
		
		`distance ← USERINPUT` (A)
		`IF distance ≤ 0` (C)
		`  distance ← USERINPUT`
		`ENDIF`
		`passengers ← USERINPUT` (D)
		`fare ← 2 * passengers` (E)
		`fare ← fare + (1.5 * distance)` (F)
		`OUTPUT fare` (G)
		(Mark B not awarded as `IF` used instead of iteration and mark H not awarded as not completely correct)	

08	1	**Mark is for AO1 (understanding)** Only reward if column **A AND B** is completely correct; **A.** F,F,F,T//false, false, false, true//off, off, off, on **I.** Case and minor spelling mistakes	1

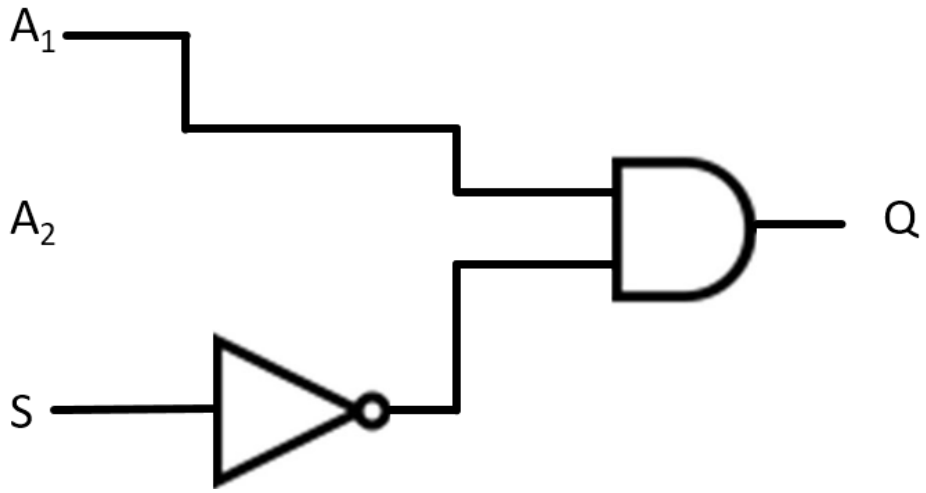
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

Question	Part	Marking guidance	Total marks
08	2	**3 marks for AO2 (apply)** Max 2 marks if not fully correct (the fully correct answer is given in example 1). Mark A if A_1 and A_2 are the inputs to an OR gate; Mark B if S is the only input into a NOT gate; Mark C if Q has a single output connection from an AND gate; **Example 1 (Fully correct answer)**  **R.** Incorrect symbols **See next page for partially correct answers**	3

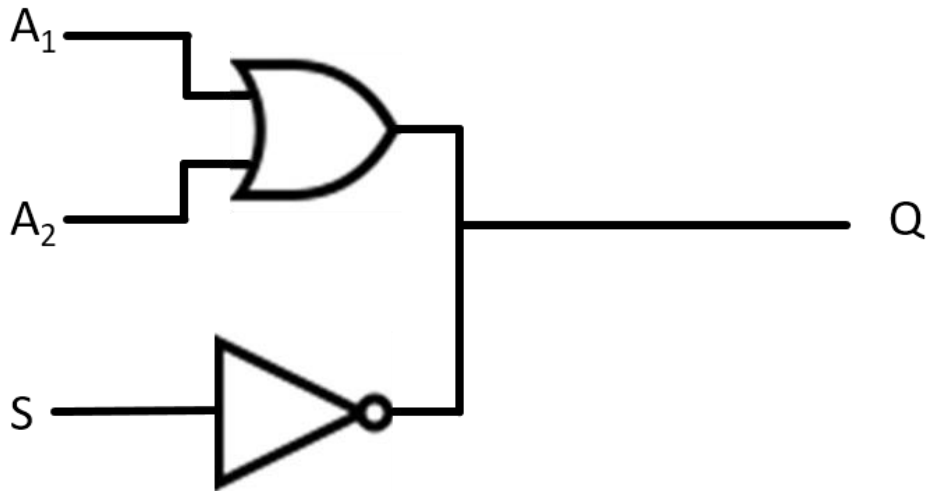
Example 2 (Marks B and C)



Example 3 (Marks B and C)



Example 4 (Marks A and B)



Question	Part	Marking guidance	Total marks
09	1	**3 marks for AO2 (apply)** 1 mark for C written **once** and in column 1; 1 mark for A and B written **once** and both in column 2; 1 mark for A and B written **once** and in correct positions in column 2; Column 0 Column 1 Column 2 A ____ C B	3
09	2	**3 marks for AO2 (apply)** 1 mark for A written **once** and in correct column (0); 1 mark for B written **once** and in correct column (2); 1 mark for C written **once** and in correct column (1); Column 0 Column 1 Column 2 A C B	3
09	3	**3 marks for AO2 (apply)** If any value is written more than once no marks for that value. 3 marks if A, B and C are all written **once**, in correct columns and in correct position (see diagram below). If not fully correct then a maximum of 2 from: 1 mark for A column 1 (even if not only value present); 2 marks for column 2 correct; 2 marks if B is above C in column 2 with A in column 2 as well in any position (assuming A, B and C are only written once); 1 mark if either one or both of B or C are present in column 2 (possibly with A as well and assuming B and C are only written once); 1 mark if A is in an incorrect column and B and C are in another incorrect column but are in the correct order and all are only written once; Column 0 Column 1 Column 2 B ____ A C	3

Question	Part	Marking guidance	Total marks
09	4	**5 marks for AO3 (program)** **Note for mark C – DPT for same logical error in the Boolean condition** Maximum of 5 marks; Mark A for using a `WHILE` loop or similar to move from column 0 to column 2; Mark B for a Boolean condition that detects when the column 0 is empty; Mark C for using a second WHILE loop or similar to move the result from A and B into column 1 (both the loop and the associated Boolean condition need to be correct to gain this mark); **or** Mark A for using a `FOR` loop or similar to move from column 0 to column 2; Mark B for ascertaining the terminating value for the `FOR` loop; Mark C for using a second `FOR` loop or similar to move the result from A and B into column 1 (both the loop and the associated terminating value need to be correct to gain this mark); **and** Mark D for using the subroutines correctly throughout, i.e. called with appropriate parameters and return values handled correctly; Mark E if algorithm is completely correct; **A.** Minor spelling errors such as `HIEGHT` for `HEIGHT` **Example 1** `WHILE HEIGHT(0) > 0` (Part of A, B) `  MOVE(0, 2)` (Part of A) `ENDWHILE` `WHILE HEIGHT(2) > 0` (Part of C) `  MOVE(2, 1)` (Part of C) `ENDWHILE` (`MOVE` and `HEIGHT` are used correctly throughout so D and completely correct so also E.)	5

Example 2

```
DO                          (Part of A)
   MOVE(0, 2)               (Part of A)
WHILE HEIGHT(0) > 0         (Part of A, B)
DO                          (Part of C)
   MOVE(2, 1)               (Part of C)
WHILE HEIGHT(2) > 0         (Part of C)
```

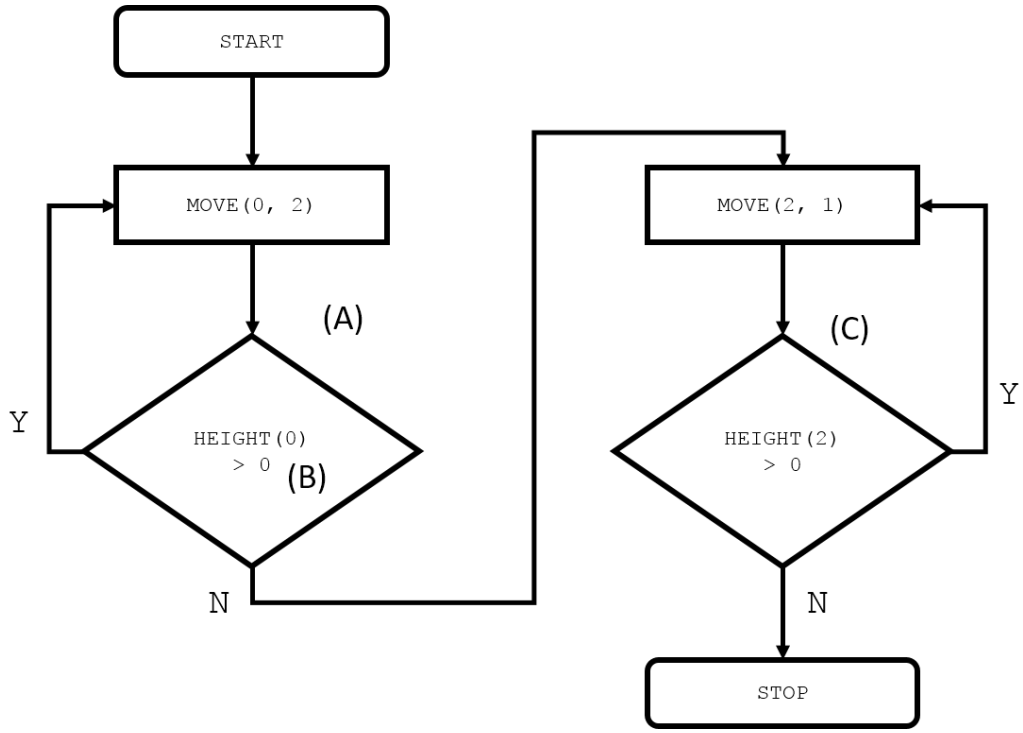
(`MOVE` and `HEIGHT` are used correctly throughout so D and completely correct so also E.)

Example 3

```
REPEAT                      (Part of A)
   MOVE(0, 2)               (Part of A)
UNTIL HEIGHT(0) = 0         (Part of A, B)
REPEAT                      (Part of C)
   MOVE(2, 1)               (Part of C)
WHILE HEIGHT(2) = 0         (Part of C)
```

(`MOVE` and `HEIGHT` are used correctly throughout so D and completely correct so also E.)

Example 4



(`MOVE` and `HEIGHT` are used correctly throughout so D and completely correct so also E.)

		Example 5 `number_of_blocks ← HEIGHT(0)` (Part of B) `FOR x ← 0 TO number_of_blocks` (Part of A, Part of B) `  MOVE(0, 2)` (Part of A) `ENDFOR` `FOR x ← 0 TO number_of_blocks` (Part of C) `  MOVE(2, 1)` (Part of C) `ENDFOR` (Part of C) (`MOVE` and `HEIGHT` are used correctly throughout so D and completely correct so also E.)	

10	1	**3 marks for AO2 (apply)** 1 mark if column `z` increments by 1 and starts at 0; 1 mark if column `z` has the final value 3; 1 mark if `correct` column is correct; **z** | **correct** 0 | false 1 | true 2 | 3 | | |	3

10	2	**Mark is for AO2 (apply)** false; **I.** Case	1

10	3	**Mark is for AO2 (apply)** Second row only; **New Line** | **Tick one box** `IF user = us[z] OR pass = ps[z] THEN` | `IF user = us[z] AND pass = ps[z] THEN` | Tick `IF NOT (user = us[z] AND pass = ps[z]) THEN` |	1

Question	Part	Marking guidance	Total marks
10	4	**Mark is for AO2 (apply)** Maximum 2 marks from: The program will return true as soon as a match (between username and password) is found; So there is no need to (always) iterate over the complete array(s)/list of usernames; (If a match is found and is not last in the list) the algorithm will complete in fewer steps/less time; **A**. the programmer has used fewer variables	2