GCSE
COMPUTER SCIENCE
8520/1

Paper 1 Computational Thinking and Problem-Solving

Mark scheme

June 2020

Version: 1.0 Final

206G8520/1/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.

Qu	Part	Marking guidance	Total marks
01	1	**Mark is for AO1 (recall)** (A pixel is a) single point (of colour) in an image/smallest (addressable) part of an image; **A.** Picture element **A.** alternatives to the word point eg dot, element	1
01	2	**Mark is for AO2 (apply)** 64 // 2^6;	1
01	3	**3 marks for AO2 (apply)** 500/500kB/500 kilobytes;;; If incorrect answer is given then award a maximum of 2 marks for working as follows: indicating the colour depth is 5; multiplying (800x1000) by the colour depth (even if colour depth is incorrect); correct conversion from bits to kilobytes;	3

Qu	Part	Marking guidance	Total marks

01	4	**3 marks for AO2 (apply)**
		
		the `i` column having all values 0-5 in order;
		the first three rows of the `image` column;
		the last three rows of the `image` column;
		
		Max 2 marks if any additional values given.
		
		`i` | `image`
		`0` | `/`
		`1` | `//`
		`2` | `//*`
		`3` | `/`
		`4` | `/*`
		`5` | `/*/`
			3

02		**3 marks for AO1 (understanding)** Start at the beginning (of the array/list); compare each element/item until the value being searched for is found; or the end of the array/list is reached;	3
03	1	**Mark is for AO1 (recall)** AND;	1
03	2	**3 marks for AO2 (apply)** 1 mark for one correct label; 2 marks for two correct labels; 3 marks for all correct labels; **Label** | **Logic Gate** L1 | NOT L2 | AND L3 | AND L4 | OR	3

<table>
<tr><th>Qu</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>04</td><td>1</td><td>Mark is for AO2 (apply)

B: Integer;

R. If more than one lozenge shaded.</td><td>1</td></tr>
<tr><td>04</td><td>2</td><td>1 mark for AO2 (apply)

Boolean/bool;</td><td>1</td></tr>
<tr><td>04</td><td>3</td><td>3 marks for AO2 (apply)

1 mark for each correct value of <code>valid</code>;;;

<table>
<tr><th>Value of <code>instr</code></th><th>Final value of <code>valid</code></th></tr>
<tr><td><code>ADD R0, R1</code></td><td><code>False</code></td></tr>
<tr><td><code>ADD: R0, R1</code></td><td><code>True</code></td></tr>
<tr><td><code>HALT</code></td><td><code>True</code></td></tr>
</table></td><td>3</td></tr>
<tr><td>04</td><td>4</td><td>Mark is for AO1 (understanding)

Machine code;

A. binary;
A. object code;</td><td>1</td></tr>
<tr><td>04</td><td>5</td><td>2 marks for AO1 (understanding)

Max 2 marks from:

(High-level languages) are better supported;
(High-level languages) provide built-in subroutines;
(High-level languages) provide programming structures such as iteration and selection;
(Code written in high-level languages) is normally shorter;
(High-level languages) allow creation of subroutines;
(High-level languages) provide data structures;
(High-level languages) are easier to understand/read;
(High-level languages) are easier to debug;

A. any other correct justification.</td><td>2</td></tr>
</table>

Qu	Part	Marking guidance	Total marks

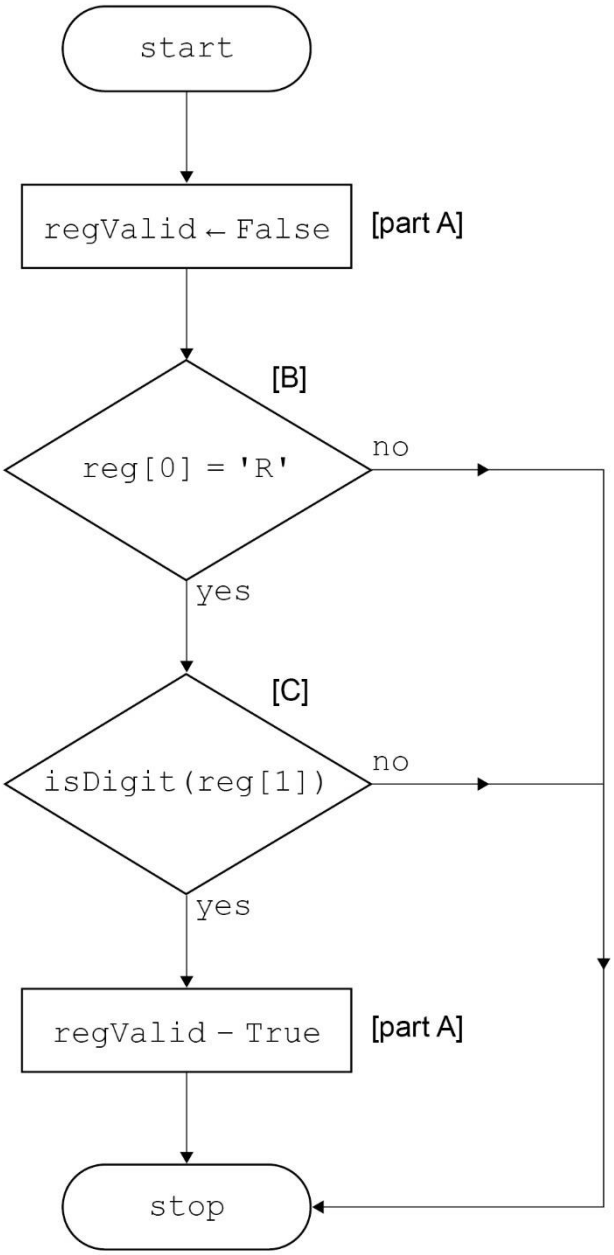
04	6	**3 marks for AO3 (program)**
		
		Mark A for setting the variable `regValid` to `True`/`False` within a selection structure;
		Mark B for using a Boolean condition that checks if the first character is an `'R'`;
		Mark C for using a Boolean condition that checks if the second character is a digit;
		
		Max 2 marks if any errors in the answer.
		
		A. minor changes to variable identifiers if the meaning is still clear.
		
		Example of fully correct answer:
		
		`regValid ← False` [part A]
		`IF reg[0] = 'R' and isDigit(reg[1]) THEN` [B,C]
		`  regValid ← True` [part A]
		`ENDIF`
		
		Example of another fully correct answer:
		
		`IF reg[0] = 'R' THEN` [B]
		`  IF isDigit(reg[1]) THEN` [C]
		`  regValid ← True` [part A]
		`  ELSE`
		`  regValid ← False` [part A]
		`  ENDIF`
		`ELSE`
		`  regValid ← False` [part A]
		`ENDIF`
		
		Example of 2 mark answer:
		
		`IF reg[0] = 'R' or isDigit(reg[1]) THEN` [B,C]
		`  regValid ← True` [part A]
		`ELSE`
		`  regValid ← True` [part A]
		`ENDIF`
		
		(only 2 marks awarded as the answer contains an error in the Boolean condition)
			3

Example of another 2 mark answer:

```
IF reg[0] = 'R' and isDigit(reg[1]) THEN [B,C]
     regValid ← True                     [part A]
ENDIF
```

(only 2 marks awarded as only part of mark A is given)

Example of a fully correct flowchart solution:

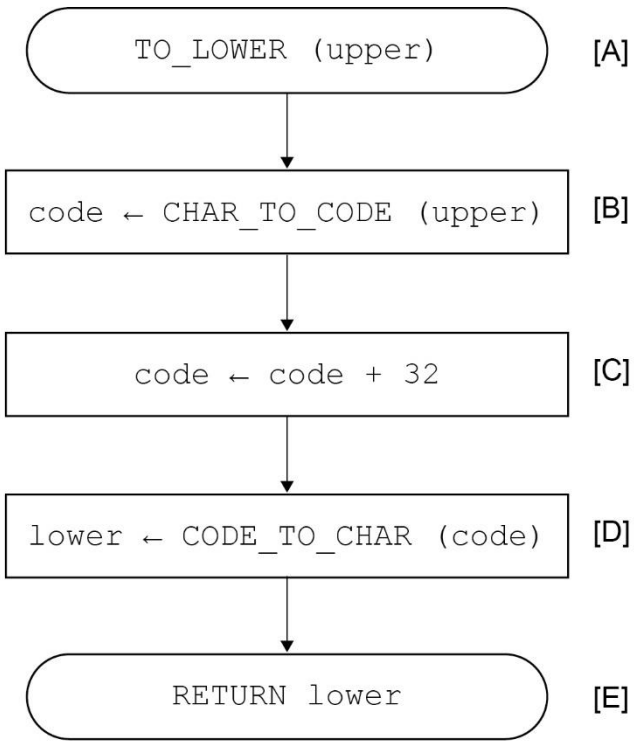


Qu	Part	Marking guidance	Total marks
05	1	**2 marks for AO2 (apply)** The first value of result 16; The last value of result 12; Max 1 mark if more than two values are given for result. The correct table is as follows: `result` 16 12 	2
05	2	**2 marks for AO2 (apply)** The `x` column fully correct; The `result` column fully correct; If more values are given in any column then max 1 mark. The correct table is as follows: `x` | `result` | 0 1 | 4 2 | 8 3 | 12 | **I.** Horizontal alignment of values as long as the vertical order of values is correct.	2
05	3	**Mark is for AO2 (apply)** (The purpose of the algorithms is) to multiply the value in `number` by 3; **A.** The value 4 instead of `number`. **NE.** Multiply two numbers.	1
05	4	**Mark is for AO2 (apply)** The algorithm in **Figure 4** uses fewer steps/instructions; **A.** The algorithm in **Figure 4** uses fewer variables; **A.** The algorithm in **Figure 4** has fewer instructions so will take up less memory; **A.** The algorithm in **Figure 4** will execute in less time; **A.** Opposite statements for **Figure 5**. **NE.** Reference to number of lines.	1

Qu	Part	Marking guidance	Total marks
06		**4 marks for AO2 (apply)** Maximum 4 marks from: **If bubble sort chosen then:** 8 & 4 are swapped; 1 & 8 are swapped; 5 & 8 are swapped; 1 & 4 are swapped; swap two consecutive numbers if the left number was greater than the right number; would repeat passes until no swaps are made/all numbers are sorted // a pass of the array `[1, 4, 5, 8]` requiring no swaps and so the algorithm stops; *or by diagram:* *8* *4* *1* *5* ; (both lines required) 4 8 1 5 4 1 8 5 ; 4 1 5 8 ; 1 4 5 8 ; **R.** the final (sorted) array if no prior arrays (excluding `[8, 4, 1, 5]`) are given. **If merge sort chosen then:** separate the array into arrays that contain only one element;; combine pairs of arrays, ordering the numbers // the values 8 and 4 combine to form the array `[4, 8]` and the value 1 and 5 combine to form the array `[1, 5]`; the arrays `[4, 8]` and `[1, 5]` combine to form the array `[1, 4, 5, 8]` / sorted array // 4 is compared with 1, 4 is compared with 5, 8 is compared with 5; Or by diagram (to a max 4 marks): `8, 4, 1, 5` ; (both lines required) `8, 4  1, 5` `8  4  1  5` ; `4, 8  1, 5` ; `1, 4, 5, 8` ; [A] **R.** mark [A] if preceding row not given.	4

Qu	Part	Marking guidance	Total marks
07	1	**4 marks for AO2 (apply)** first (calculated) value of 10; next calculated value of 5; next calculated value of 16; all values of 8, 4, 2 and 1 in that order; Stop marking at the first incorrect value. Max of 3 marks if additional outputs are given. **Output** 3 10 5 16 8 4 2 1	4
07	2	**2 marks for AO1 (understanding)** **Max 2** from: (The developer has) modularised their code // used subroutines; (The developer has) decomposed the problem // broken the problem down into sub-problems; (The developer has) created interfaces (to the subroutines); (The developer has) used parameters; (The developer has) used return values; (The developer has) used local variables;	2
08	1	**Mark is for AO1 (understanding)** **B**: `'d'`; **R.** If more than one lozenge shaded.	1
08	2	**Mark is for AO2 (apply)** G; **R.** g **I.** use of quote marks	1

Qu	Part	Marking guidance	Total marks
08	3	**5 marks for AO3 (program)** **Mark A** for defining a subroutine with the identifier TO_LOWER and one parameter; **Mark B** for using CHAR_TO_CODE with a variable parameter; **Mark C** for adding 32 to the result of mark B; **Mark D** for using the result of mark C as a parameter to the CODE_TO_CHAR subroutine; **Mark E** for returning the value of mark D; **Max 4 marks** if any errors in answer. Example of fully correct answer: `SUBROUTINE TO_LOWER(upper)` *[A]* `  code ← CHAR_TO_CODE(upper)` *[B]* `  code ← code + 32` *[C]* `  lower ← CODE_TO_CHAR(code)` *[D]* `  RETURN lower` *[E]* `ENDSUBROUTINE` Another example of a fully correct answer: `SUBROUTINE TO_LOWER(upper)` *[A]* `  code ← CHAR_TO_CODE(upper)` *[B]* `  RETURN CODE_TO_CHAR(code + 32)` *[C,D,E]* `ENDSUBROUTINE` Another example of a fully correct answer: `SUBROUTINE TO_LOWER(upper)` *[A]* `  RETURN CODE_TO_CHAR(CHAR_TO_CODE(upper) + 32)` *[B,C,D,E]* `ENDSUBROUTINE` Example of a 4 mark answer: `SUBROUTINE TO_LOWER(upper)` *[A]* `  code ← CHAR_TO_CODE(character)` *[B]* `  code ← code + 32` *[C]* `  lower ← CODE_TO_CHAR(code)` *[D]* `  RETURN lower` *[E]* `ENDSUBROUTINE` (only 4 marks awarded as answer contains an error where parameter to CHAR_TO_CODE is different to parameter for TO_LOWER)	5

		Example of a fully correct flowchart solution:  	

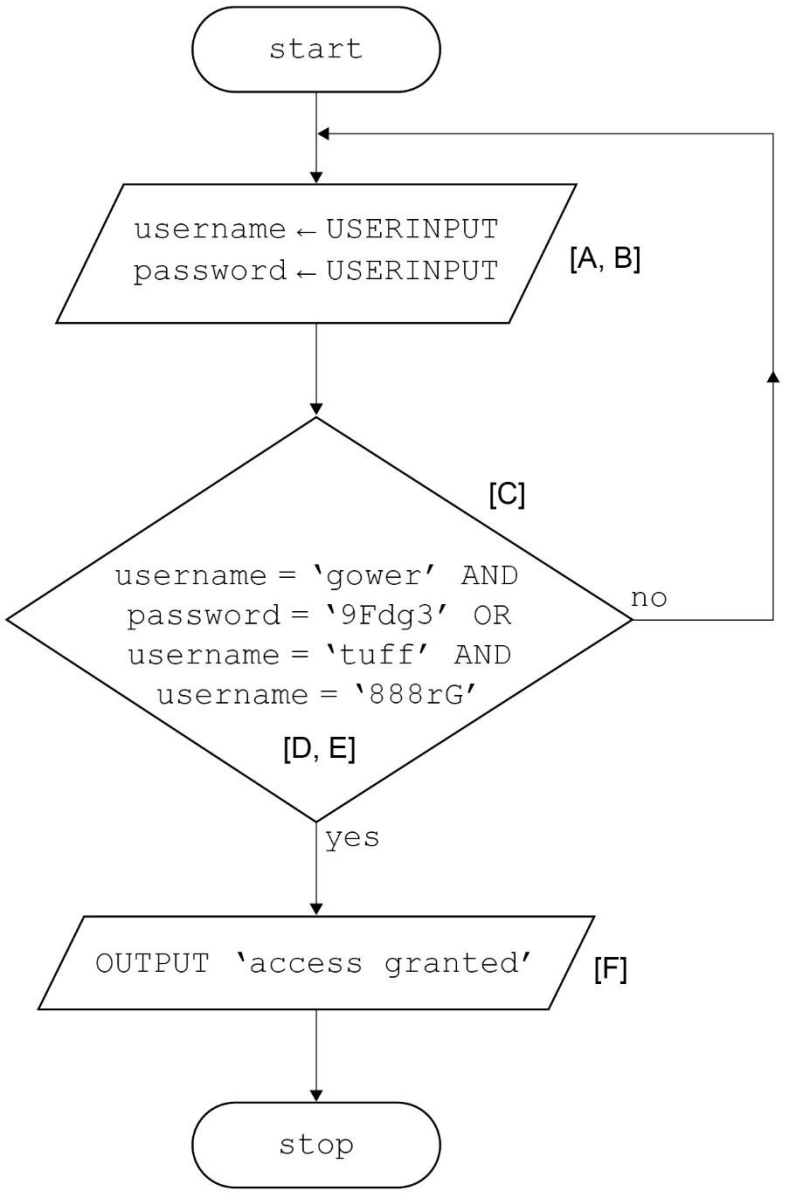
09		**4 marks for AO2 (apply)** A record could be used to store the data of one song; An array could store all of the songs/records; One mark for one of the following, two marks for all three: • The song title could be a string • The singer could be a string • The year of release could be an integer/date.	4

Qu	Part	Marking guidance	Total marks
10		**6 marks for AO3 (program)** **Mark A** for assigning user input to a variable (username); **Mark B** for assigning user input to a variable (password, the identifier must be different to that used in mark A); **Mark C** for using indefinite iteration and including user input within the iteration structure; **Mark D** for using a Boolean condition that checks the username is `gower` and the password is `9Fdg3` / the username is `tuff` and the password is `888rG`; **Mark E** for using the Boolean `OR` operator for both combinations of username and password, alternatively having sequential `IF` or `ELSE-IF` structures; **Mark F** for outputting the string after the iteration structure; **Max 5 marks** if the algorithm contains any errors. **I.** use of quote marks for usernames or passwords. **I.** minor spelling errors for username or passwords. Example of fully correct answer: `REPEAT` *[part C]* `  username ← USERINPUT` *[A, part C]* `  password ← USERINPUT` *[B, part C]* `UNTIL (username = 'gower' AND password = '9Fdg3') OR (username = 'tuff' AND password = '888rG')` *[D, E]* `OUTPUT 'access granted'` *[F]* Another example of a fully correct answer: `username ← USERINPUT` *[A]* `password ← USERINPUT` *[B]* `WHILE NOT ((username = 'gower' AND password = '9Fdg3') OR (username = 'tuff' AND password = '888rG'))` *[D, E, part C]* `  username ← USERINPUT` *[part C]* `  password ← USERINPUT` *[part C]* `ENDWHILE` `OUTPUT 'access granted'` *[F]*	6

Another example of a fully correct answer:

```
username ← USERINPUT                    [A]
password ← USERINPUT                    [B]
valid ← false                           [part D]
WHILE NOT valid                         [part C, part D]
   IF (username = 'gower' AND           [part D, E]
       password = '9Fdg3') OR
      (username = 'tuff' AND
       password = '888rG')) THEN
      valid ← true
   ELSE
      username ← USERINPUT              [part C]
      password ← USERINPUT              [part C]
ENDWHILE
OUTPUT 'access granted'                 [F]
```

An example of a fully correct flowchart solution:



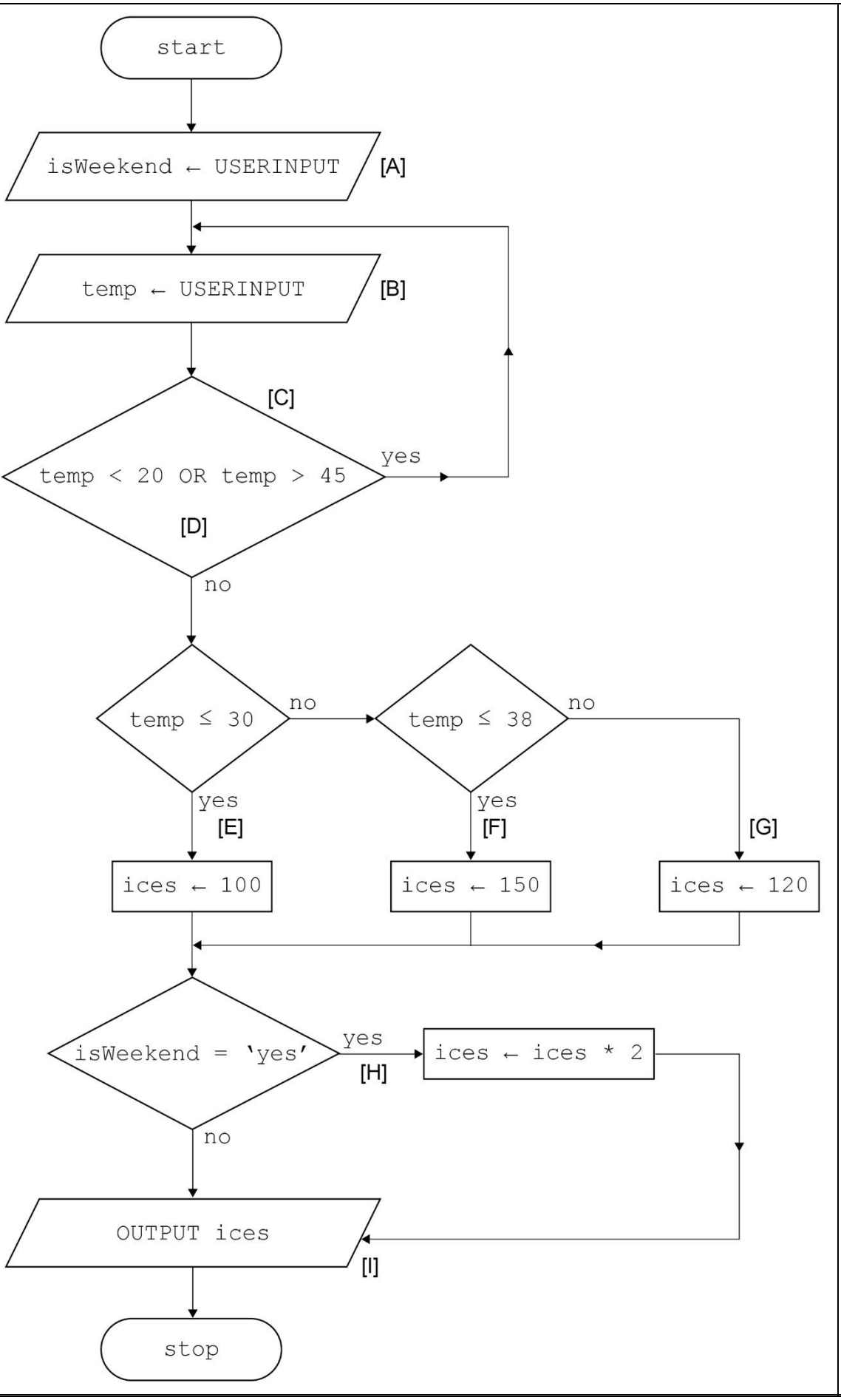
Qu	Part	Marking guidance	Total marks

11		**9 marks for AO3 (program)**
		
		Mark A for assigning user input to a variable (weekend or weekday);
		Mark B for assigning user input to a variable (temperature);
		Mark C for using indefinite iteration to repeatedly input the temperature;
		Mark D for a Boolean condition used to check the temperature between 20 and 45 inclusive;
		Mark E for using selection to set ice creams to be 100 if the temp is between 20 and 30 inclusive;
		Mark F for using selection to set ice creams to be 150 if the temp is between 31 and 38 inclusive;
		Mark G for using selection to set ice creams to be 120 if the temp is higher than 38;
		Mark H for doubling the quantity if it is a weekend (mark A is not required);
		Mark I for **always** outputting the estimated number of ice creams;
		
		Max 8 marks if solution contains any errors.
		
		An example of a fully correct solution:
		
		`isWeekend ← USERINPUT` *[A]*
		`temp ← USERINPUT` *[B]*
		`WHILE temp < 20 OR temp > 45` *[part C, D]*
		`  temp ← USERINPUT` *[part C]*
		`ENDWHILE`
		`IF temp ≤ 30 THEN` *[part E]*
		`  ices ← 100` *[part E]*
		`ELSE IF temp ≤ 38 THEN` *[part F]*
		`  ices ← 150` *[part F]*
		`ELSE` *[part G]*
		`  ices ← 120` *[part G]*
		`ENDIF`
		`IF isWeekend = 'yes' THEN` *[part H]*
		`  ices ← ices * 2` *[part H]*
		`ENDIF`
		`OUTPUT ices` *[part I]*
			9

Another example of a fully correct solution:

```
isWeekend ← USERINPUT                [A]
DO                                   [part C]
   temp ← USERINPUT                  [B]
WHILE temp < 20 OR temp > 45         [part C, D]
IF temp ≤ 30 THEN                    [part E]
   ices ← 100                        [part E]
ELSE IF temp ≤ 38 THEN               [part F]
   ices ← 150                        [part F]
ELSE                                 [part G]
   ices ← 120                        [part G]
ENDIF
IF isWeekend = 'yes' THEN            [part H]
   ices ← ices * 2                   [part H]
ENDIF
OUTPUT ices                          [part I]
```

An example of a fully correct flowchart solution:


start
isWeekend ← USERINPUT
[A]
temp ← USERINPUT
[B]
[C]
temp < 20 OR temp > 45
yes
[D]
no
temp ≤ 30
no
temp ≤ 38
no
yes
[E]
yes
[F]
[G]
ices ← 100
ices ← 150
ices ← 120
isWeekend = 'yes'
yes
[H]
ices ← ices * 2
no
OUTPUT ices
[I]
stop

Qu	Part	Marking guidance	Total marks
12	1	**3 marks for AO2 (apply)** 1 mark for index 0 set to off; 1 mark for index 2 set to on; 1 mark for index 3 set to off; **Max 2 marks** if one error anywhere in the array. **Max 1 mark** if two errors anywhere in the array. **0 marks** if more than two errors anywhere in the array. 0 1 2 3 4 5 6 off off on off off off on	3
12	2	**3 marks for AO2 (apply)** 1 mark for indices 0, 1 and 2 set to on, on and off respectively; 1 mark for index 4 set to off; 1 mark for index 5 set to off; **Max 2 marks** if one error anywhere in the array. **Max 1 mark** if two errors anywhere in the array. **0 marks** if more than two errors anywhere in the array. 0 1 2 3 4 5 6 on on off off off off on	3
12	3	**3 marks for AO2 (apply)** 1 mark for index 0 set to on and index 1 set to off; 1 mark for index 2 set to on; 1 mark for indices 5 and 6 set to off and on respectively; **Max 2 marks** if one error anywhere in the array. **Max 1 mark** if two errors anywhere in the array. **0 marks** if more than two errors anywhere in the array. 0 1 2 3 4 5 6 on off on on off off on	3

Qu	Part	Marking guidance	Total marks

12	4	**3 marks for AO3 (program)**
		
		3 marks if each of the subroutines is used correctly exactly once to produce the correct final array;;;
		
		2 marks if the subroutines are used correctly to produce the correct final array but three subroutines are not used or a subroutine is used more than once;;
		
		1 mark if at least two subroutines (possibly the same) are used correctly but the final array is incorrect;
		
		A. 1 mark for `RANGEOFF(-1, 7);`
		
		First full mark example answer:
		
		`RANGEOFF(0, 6)`
		`NEIGHBOUR(0)`
		`SWITCH(6)`
		
		Second full mark example answer:
		
		`RANGEOFF(0, 6)`
		`SWITCH(6)`
		`NEIGHBOUR(0)`
		
		An example 2 mark answer (not all subroutines are used):
		
		`RANGEOFF(0, 6)`
		`SWITCH(6)`
		`SWITCH(0)`
			3