<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>
</table>

Question	Part	Marking guidance	Total marks

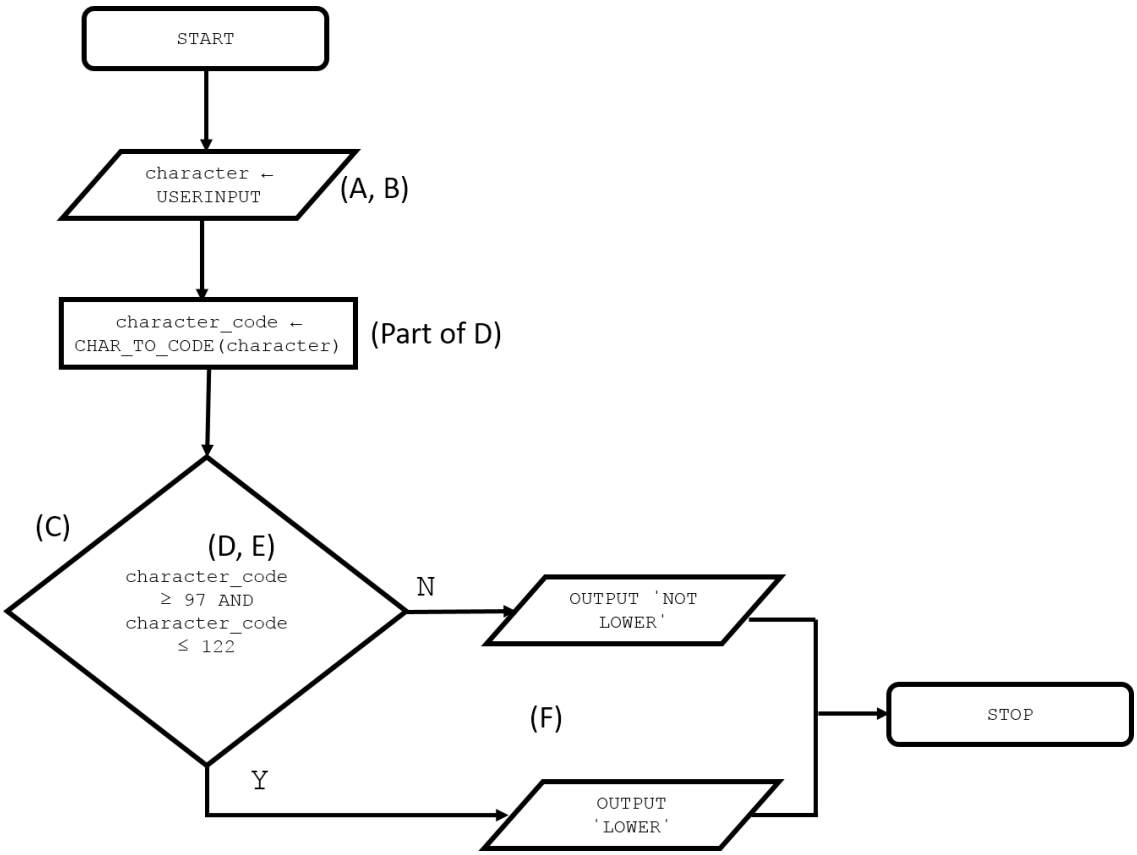
02		**7 marks for AO3 (program)**
		
		If `CHAR_TO_CODE` is not used then a maximum of 6 marks.
		
		Mark A for using user input;
		Mark B for storing the result of user input in a variable or using the user input directly as a parameter to `CHAR_TO_CODE`;
		Mark C for using selection to determine if 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)

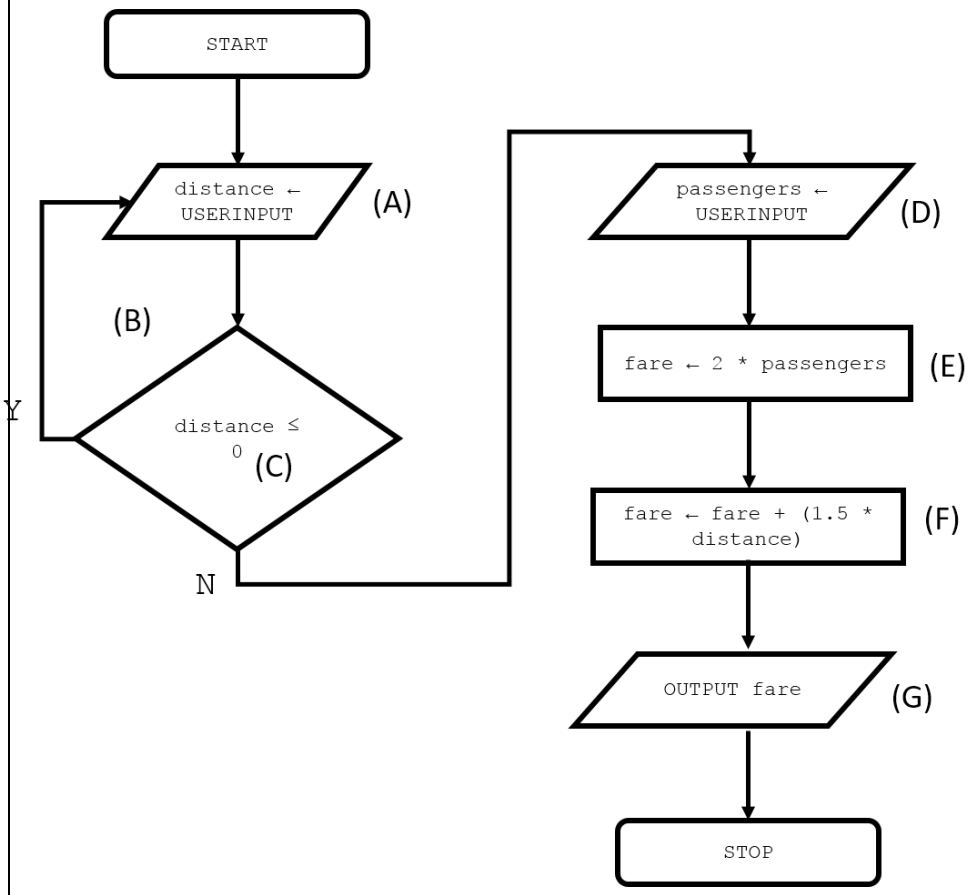
Example 6 (6 marks)

```
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
03		**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)

Question	Part	Marking guidance	Total marks
04	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 C A B	3
04	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
04	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 A B C	3

Question	Part	Marking guidance	Total marks
04	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)
```

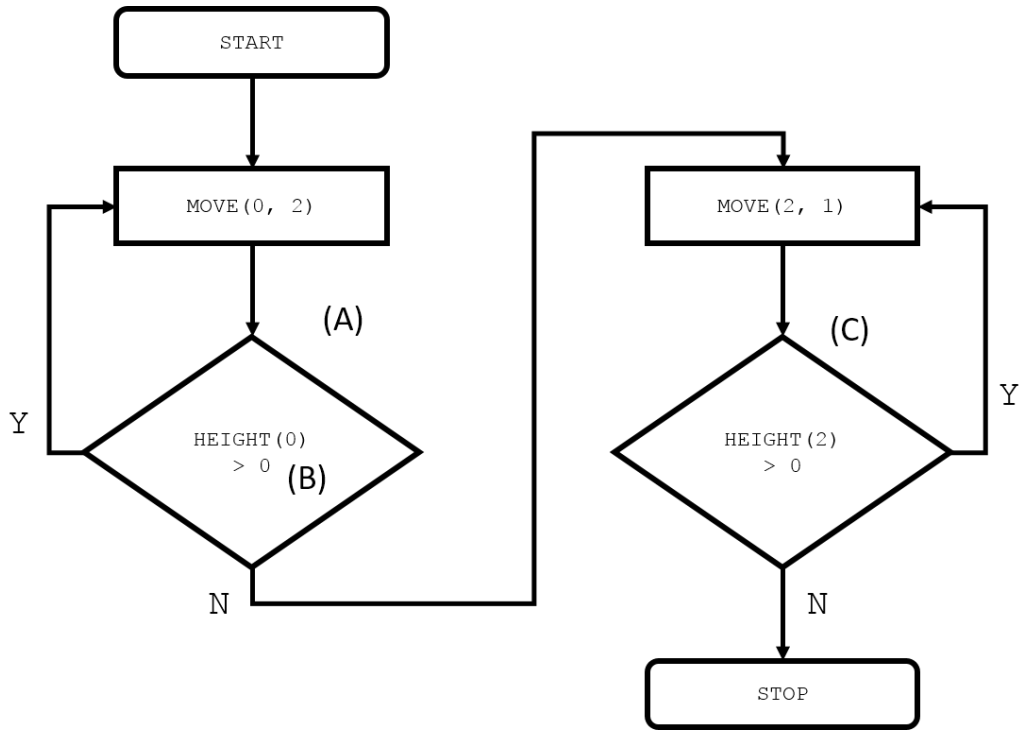
(`MOVE` and `HEIGHT` are used correctly throughout so D and completely correct so also E.)

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.)	

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>05</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>totalSize</th><th>dataToBeSent</th><th>numberOfPackets</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>05</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>05</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>05</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
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

Question	Part	Marking guidance	Total marks

07		**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

Qu	Part	Marking guidance	Total marks
08	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
08	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
08	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
08	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

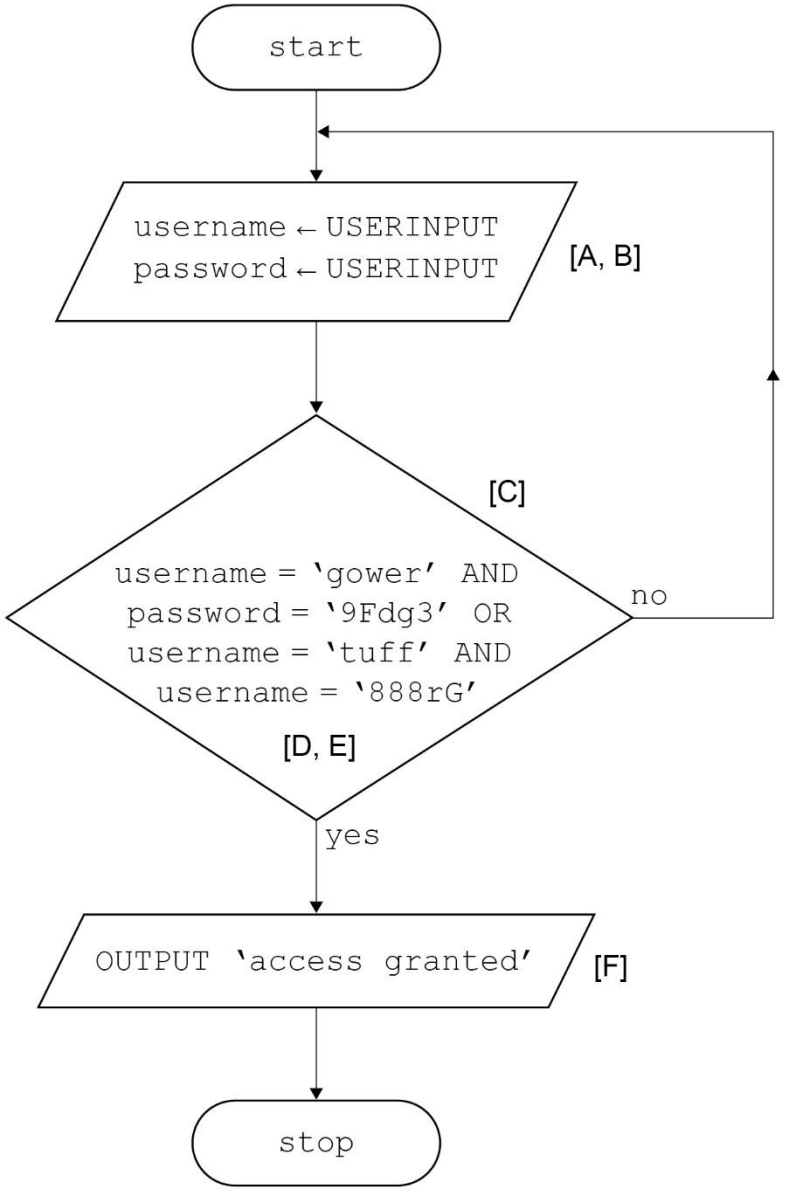
Qu	Part	Marking guidance	Total marks
09	1	**4 marks for AO2 (apply)** first (calculated) value of 10; next calculated value of 5; next calculated value of 16; all values of 8, 4, 2 and 1 in that order; Stop marking at the first incorrect value. Max of 3 marks if additional outputs are given. **Output** 3 10 5 16 8 4 2 1	4
09	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

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:



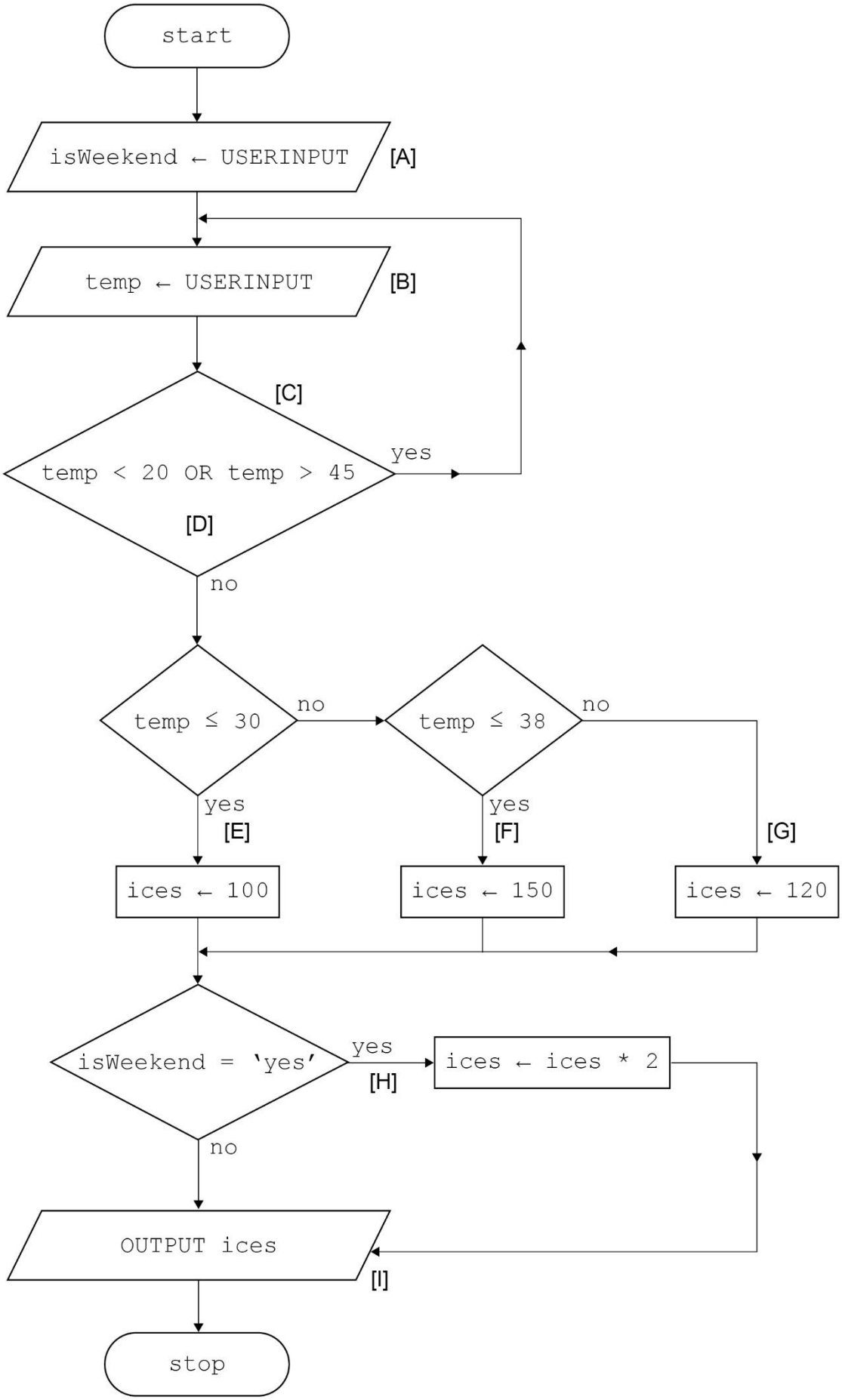
Qu	Part	Marking guidance	Total marks

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

Question	Part	Marking guidance	Total marks
12	1	**Mark is for AO1 (recall)** Removing unnecessary detail (from the problem); **A.** data / information in place of detail for this year only	1

Question	Part	Marking guidance	Total marks
12	2	**2 marks for AO2 (apply)** **A** Confirm / enter email address; **B** Log out; **A.** any wording with the same meaning	2

Question	Part	Marking guidance	Total marks

13		**2 marks for AO3 (design), 3 marks for AO3 (program)**
		
		Program Design
		Note that AO3 (design) marks are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.
		
		Mark A for using meaningful variable names throughout and for using two variables to store the two email address inputs;
		Mark B for the use of a selection construct // use of multiple selection constructs;
		
		Program Logic
		Mark C for using user input and storing the results in two variables correctly for the first email address and the second email address;
		Mark D for a correct expression that checks if the first entered email address is equal to the second entered email address (or not equal to);
		Mark E for outputting `Do not match` and `Match` in logically separate places such as the IF and ELSE part of selection, and for outputting the email address if both email addresses match;
		
		A. Any suitable alternative messages.
		
		I. Case
		I. Messages or no messages with input statements
		
		Maximum 4 marks if any errors in code.
		
		C# Example 1 (fully correct)
		All design marks are achieved (**Marks A and B**)
		
		`string email1 = Console.ReadLine();` (**Part of C**)
		`string email2 = Console.ReadLine();` (**Part of C**)
		
		`if (email1 != email2) {` (**D**)
		`  Console.WriteLine("Do not match");` (**Part of E**)
		`}`
		`else {`
		`  Console.WriteLine("Match");` (**Part of E**)
		`  Console.WriteLine(email1);` (**Part of E**)
		`}`
			5

C# Example 2 (fully correct)
All design marks are achieved (**Marks A and B**)

```
string em1 = Console.ReadLine();                    (Part of C)
string em2 = Console.ReadLine();                    (Part of C)

if (em1 == em2)      {                              (D)
   Console.WriteLine("Match");                      (Part of E)
   Console.WriteLine(em2);                          (Part of E)
}
else {
   Console.WriteLine("Do not match");               (Part of E)
}
```

Python Example 1 (fully correct)
All design marks are achieved (**Marks A and B**)

```
email1 = input()                                    (Part of C)
email2 = input()                                    (Part of C)

if email1 != email2:                                (D)
   print("Do not match")                            (Part of E)
else:
   print("Match")                                   (Part of E)
   print(email1)                                    (Part of E)
```

Python Example 2 (fully correct)
All design marks are achieved (**Marks A and B**)

```
em1 = input()                                       (Part of C)
em2 = input()                                       (Part of C)

if em1 == em2:                                      (D)
   print("Match")                                   (Part of E)
   print(em2)                                       (Part of E)
else:
   print("Do not match")                            (Part of E)
```

Python Example 3 (partially correct – 4 marks)
All design marks are achieved (**Marks A and B**)

```
email1 = input()                                    (Part of C)
email2 = input()                                    (Part of C)

if email1 == email2:                                (D)
   print("Match")
```

VB.NET Example 1 (fully correct)
All design marks are achieved (**Marks A and B**)

```
Dim email1 As String = Console.ReadLine()      (Part of C)
Dim email2 As String = Console.ReadLine()      (Part of C)

If email1 <> email2 Then                       (D)
   Console.WriteLine("Do not match")           (Part of E)
Else
   Console.WriteLine("Match")                  (Part of E)
   Console.WriteLine(email1)                   (Part of E)
End If
```

VB.NET Example 2 (fully correct)
All design marks are achieved (**Marks A and B**)

```
Dim em1 As String = Console.ReadLine()         (Part of C)
Dim em2 As String = Console.ReadLine()         (Part of C)

If em1 = em2 Then                              (D)
   Console.WriteLine("Match")                  (Part of E)
   Console.WriteLine(em2)                      (Part of E)
Else
   Console.WriteLine("Do not match")           (Part of E)
End If
```

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>14</td><td>1</td><td>3 marks for AO2 (apply)

Maximum 2 marks if Output shows numbers or text only with no other errors OR fully correct but contains additional characters.

Maximum 1 mark if Output shows numbers or text only or is inconsistent AND there is at least one error, even if additional characters present.

<table>
<tr><th>First user input</th><th>Second user input</th><th>Third user input</th><th>Output</th></tr>
<tr><td>5</td><td>6</td><td>–1</td><td>Area 30</td></tr>
<tr><td>10</td><td>4</td><td>0</td><td>Volume 0</td></tr>
<tr><td>3</td><td>5</td><td>10</td><td>Volume 150</td></tr>
</table>
I. quotation marks in the Output column</td><td>3</td></tr>
</table>

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>14</td><td>2</td><td>Mark is for AO2 (apply)

Maximum of 1 mark from:

● Add validation; A. by example eg check width/length are positive numbers // check height is –1 or a positive number;

● Change data types used in the question to float / single / double / decimal / real for inputs;</td><td>1</td></tr>
</table>

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>15</td><td></td><td>3 marks for AO2 (apply)

<table>
<tr><th>a</th><th>b</th><th>c</th></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>2</td></tr>
<tr><td>1</td><td>2</td><td>3</td></tr>
<tr><td>2</td><td>3</td><td>5</td></tr>
</table>

1 mark for correct first row;
1 mark for correct second row;
1 mark for correct third and fourth rows;

Maximum 2 marks if any errors

I. different rows used as long as the order within columns is clear
I. duplicate values on consecutive rows within a column

Note to examiners: Check vertically as well as horizontally for the effect of duplicate values.</td><td>3</td></tr>
</table>

Question	Part	Marking guidance	Total marks
16	1	**5 marks for AO2 (apply)** **1 mark** for `count` column correct; **1 mark** for column `i` correct; **1 mark** for the first Natalie row, including `j` and `result` correct – not including `i` and `count`; **1 mark** for the second Natalie row, including `j` and `result` correct – not including `i` and `count`; **1 mark** for all of Alex and Roshana rows correct as for Natalie above; **I.** different rows used as long as the order within columns is clear **I.** duplicate values on consecutive rows within a column **I.** quotes used around letters (person column) **I.** minor spelling mistakes in the person column	5

count	i	person	j	result
0	0	Natalie	0	78
1			1	81
2	1	Alex	0	27
3			1	51
4	2	Roshana	0	52
5			1	55
6				

Question	Part	Marking guidance	Total marks
16	2	**Mark is for AO2 (apply)** **C** Change line number 7 to: `FOR j ← 0 TO 2` **R.** if more than one lozenge shaded	1

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>17</td><td>1</td><td>2 marks for AO1 (recall)

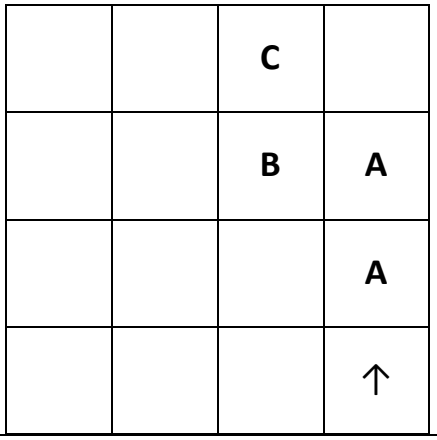
A sequence of steps/instructions;
that can be followed to complete a task;

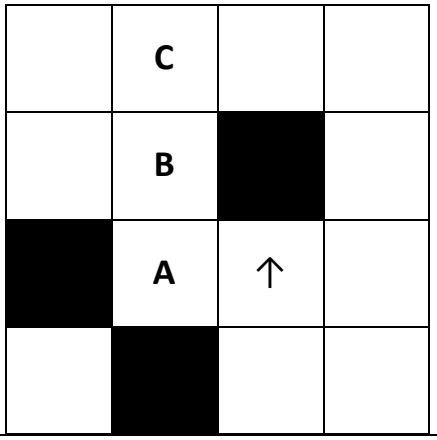
A. Different wording with similar meaning</td><td>2</td></tr>
<tr><td>17</td><td>2</td><td>3 marks for AO1 (recall)

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 setting the value stored in a variable</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>
</table>

Question	Part	Marking guidance	Total marks
18	1	**3 marks for AO2 (apply)** **Mark as follows:** **1 mark** for the robot moving to **both** squares marked **A**; **1 mark** for the robot moving to the square marked **B**; **1 mark** for the robot moving to the square marked **C**; 	3

Question	Part	Marking guidance	Total marks
18	2	**3 marks for AO2 (apply)** **Mark as follows:** **1 mark** for the robot moving to the square marked **A**; **1 mark** for the robot moving to the square marked **B**; **1 mark** for the robot moving to the square marked **C**; 	3
19	1	**1 mark for AO1 (recall)** **A** Abstraction; **R.** if more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
19	2	**2 marks for AO2 (apply)** All friends have different first names; The time is rounded up to the nearest half-hour;	2
20	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 (in any order); 1 mark for A and B written once and in correct positions in column 2; Column 0 Column 1 Column 2 ____ C A B	3
20	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

Question	Part	Marking guidance	Total marks
20	3	**4 marks for AO3 (design)** **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 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; **A.** Minor spelling errors such as `HIEGHT` for `HEIGHT` **I.** Case **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**.) **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**.)	4

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)
```

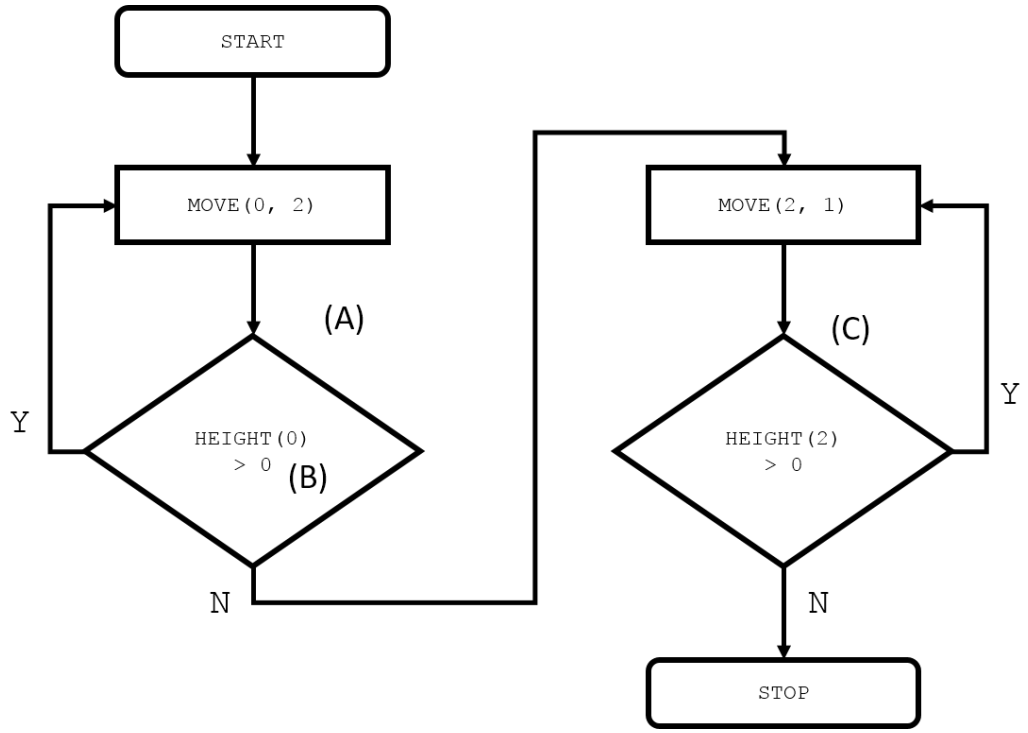
(MOVE and HEIGHT are used correctly throughout so **D**.)

Example 4

```
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**.)

Example 5



(MOVE and HEIGHT are used correctly throughout so **D**.)

Question	Part	Marking guidance	Total marks
21	1	**Mark is for AO1 (recall)** Removing unnecessary detail/information/data from the problem/task; **R.** references to code/programs	1

Question	Part	Marking guidance	Total marks
21	2	**Mark is for AO1 (recall)** Decomposition; **I.** minor spelling errors	1

Question	Part	Marking guidance	Total marks
22		**3 marks for AO2 (apply)** **Input value for** `numberOfGuests` / **Input value for** `numberOfRooms` / **Output** `50` / `30` / 3200; `20` / `10` / 1125; `500` / `5` / 1500; **DPT.** Quotes around output values **I.** commas in output values	3

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>23</td><td></td><td>
6 marks for AO2 (apply)

1 mark for the <code>i</code> column correct;

1 mark for the first value in the <code>daysTotal</code> column correct;
I. preceding zeroes

1 mark for the rest of <code>daysTotal</code> column correct;

1 mark for the second value of <code>weeks[0]</code> column correct;

1 mark for the rest of <code>weeks</code> columns correct;

1 mark for the correct total of <code>weeks[0]</code>, <code>weeks[1]</code> and <code>weeks[2]</code> in the final column and no other value;
I. preceding zeroes
A. follow through value as long as the total is correct for the three final values the student has written in the <code>weeks</code> columns.

Maximum of 5 marks if any errors.

<table>
<tr><th rowspan="2">i</th><th rowspan="2">daysTotal</th><th colspan="3">weeks</th><th>weeksTotal</th></tr>
<tr><th>[0]</th><th>[1]</th><th>[2]</th><th></th></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>0</td><td></td></tr>
<tr><td>0</td><td>30</td><td>4</td><td>0</td><td>0</td><td></td></tr>
<tr><td>1</td><td>48</td><td>4</td><td>6</td><td>0</td><td></td></tr>
<tr><td>2</td><td>16</td><td>4</td><td>6</td><td>2</td><td></td></tr>
<tr><td></td><td></td><td colspan="3"></td><td>12</td></tr>
</table>

I. Different rows used so long as the order within columns is clear
I. Duplicate values on consecutive rows within a column
</td><td>6</td></tr>
</table>