



---

GCSE

**COMPUTER SCIENCE**

**8525A/1, 8525B/1, 8525C/1**

Paper 1 Computational thinking and programming skills

---

**Mark scheme**

Specimen Assessment Materials

---

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](http://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.

**Note to Examiners**

In the real world minor syntax errors are often identified and flagged by the development environment. To reflect this, all responses in a high-level programming language will assess a candidate's ability to create an answer using precise programming commands/instructions but will avoid penalising them for minor errors in syntax.

When marking program code, examiners must take account of the different rules between the languages and only consider how the syntax affects the logic flow of the program. If the syntax is not perfect but the logic flow is unaffected then the response should not be penalised.

The case of all program code written by students is to be ignored for the purposes of marking. This is because it is not always clear which case has been used depending on the style and quality of handwriting used.

Examiners must ensure they follow the mark scheme instructions exactly. If an examiner is unsure as to whether a given response is worthy of the marks they must escalate the question to their team leader.

Question	Part	Marking guidance	Total marks								
01	1	<p><b>2 marks for AO1 (recall)</b></p> <p>A sequence of steps/instructions; that can be followed to complete a task;</p> <p><b>A.</b> Different wording with similar meaning</p>	2								
01	2	<p><b>3 marks for AO1 (recall)</b></p> <p>One mark for each correct distinct label.</p> <p>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:</p> <table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th></th> <th style="text-align: center;">Label</th> </tr> </thead> <tbody> <tr> <td>Breaking a problem down into a number of sub-problems</td> <td style="text-align: center;">C</td> </tr> <tr> <td>The process of setting the value stored in a variable</td> <td style="text-align: center;">A</td> </tr> <tr> <td>Defines the sort of values a variable may take</td> <td style="text-align: center;">B</td> </tr> </tbody> </table> <p><b>A.</b> If actual terms are written out instead of labels  <b>R.</b> All instances of duplicate labels</p>		Label	Breaking a problem down into a number of sub-problems	C	The process of setting the value stored in a variable	A	Defines the sort of values a variable may take	B	3
	Label										
Breaking a problem down into a number of sub-problems	C										
The process of setting the value stored in a variable	A										
Defines the sort of values a variable may take	B										
02	1	<p><b>Mark is for AO2 (apply)</b></p> <p><b>D</b> 4;</p> <p><b>R.</b> If more than one lozenge shaded</p>	1								
02	2	<p><b>Mark is for AO2 (apply)</b></p> <p><b>D</b> 'computer sciencegcse';</p> <p><b>R.</b> If more than one lozenge shaded</p>	1								
03	1	<p><b>Mark is for AO2 (apply)</b></p> <p><b>A</b> Line number 2;</p> <p><b>R.</b> If more than one lozenge shaded</p>	1								
03	2	<p><b>Mark is for AO2 (apply)</b></p> <p><b>C</b> Line number 11;</p> <p><b>R.</b> If more than one lozenge shaded</p>	1								

Question	Part	Marking guidance	Total marks
03	3	<p><b>Mark is for AO2 (apply)</b></p> <p><b>A</b> 1 subroutine call;</p> <p><b>R.</b> If more than one lozenge shaded</p>	1
03	4	<p><b>Mark is for AO2 (apply)</b></p> <p><b>B</b> String;</p> <p><b>R.</b> If more than one lozenge shaded</p>	1
03	5	<p><b>Mark is for AO2 (apply)</b></p> <p>2//twice//two;</p>	1
04		<p><b>5 marks for AO3 (program)</b></p> <p>1 mark for each correct item in the correct location.</p> <p><b>Python</b></p> <pre> num1 = int(input("Enter a number: ")) num2 = <b>int</b> _____ (input("Enter a second number: ")) if num1 &gt; num2:     print(" <b>num1</b> _____ is bigger.") elif num1 &lt; _____ num2:     print(" <b>num2</b> _____ is bigger.") <b>else:</b>     _____     print("The numbers are equal.")                     </pre> <p><b>I.</b> Case of response  <b>R.</b> if any spelling mistakes</p> <p><b>C#</b></p> <pre> int num1; <b>int</b> _____ num2;  Console.WriteLine("Enter a number: ");  num1 = int.Parse(Console.ReadLine());  Console.WriteLine("Enter another number: ");  num2 = int.Parse(Console.ReadLine());                     </pre>	5

	<pre> if (num1 &gt; num2) {     Console.WriteLine("  <u>num1</u> is bigger."); } else if (num1 &lt; <u>num2</u>) {     Console.WriteLine("  <u>num2</u> is bigger."); } else <u>num2</u> {     Console.WriteLine("The numbers are equal."); } </pre> <p><b>I. Case of response</b> <b>R. if any spelling mistakes</b></p> <p><b>VB.Net</b></p> <pre> Dim num1 As Integer Dim num2 As <u>Integer</u>  Console.Write("Enter a number: ")  num1 = Console.ReadLine()  Console.Write("Enter another number: ")  num2 = Console.ReadLine()  If num1 &gt; num2 Then     Console.WriteLine("  <u>num1</u> is bigger.") ElseIf num1 &lt; <u>num2</u> Then     Console.WriteLine("  <u>num2</u> is bigger.") Else <u>num2</u>     Console.WriteLine("The numbers are equal.") End If </pre> <p><b>I. Case of response</b> <b>R. if any spelling mistakes</b></p>	
--	---	--

Question	Part	Marking guidance	Total marks
05		<p><b>2 marks for AO3 (design) and 5 marks for AO3 (program)</b></p> <p><b><u>Program Design</u></b>  <b>Mark A</b> for using meaningful variable names throughout (even if logic is incorrect);  <b>Mark B</b> for using suitable data types throughout (distance can be real or integer, passengers must be integer);</p> <p><b><u>Program Logic</u></b>  <b>Mark C</b> for getting user input for the distance in an appropriate place;  <b>Mark D</b> for getting user input for the number of passengers in an appropriate place;  <b>Mark E</b> for a fare that correctly charges £2 per passenger;  <b>Mark F</b> for a fare that correctly charges £1.50 for every kilometre;  <b>Mark G</b> for outputting the correct final fare;</p> <p>I. Case of program code</p> <p><b>Maximum 6 marks</b> if any errors in code.</p> <p><b><u>Python Example 1 (fully correct)</u></b>  <b>Mark A</b> awarded.</p> <pre>distance = float(input()) passengers = int(input()) fare = 2 * passengers fare = fare + (1.5 * distance) print(fare)</pre> <p style="text-align: right;">(Part of B, C) (Part of B, D) (E) (F) (G)</p> <p><b><u>C# Example (fully correct)</u></b>  <b>Mark A</b> awarded.</p> <pre>int passengers; double distance, fare; distance = double.Parse(Console.ReadLine()); passengers = int.Parse(Console.ReadLine()); fare = 2 * passengers; fare = fare + (1.5 * distance); Console.WriteLine(fare);</pre> <p style="text-align: right;">(Part of B) (Part of B) (C) (D) (E) (F) (G)</p> <p>I. indentation in C#</p> <p><b><u>VB Example (fully correct)</u></b>  <b>Marks A, B</b> awarded.</p> <pre>Dim distance, fare As Double Dim passengers As Integer distance = Console.ReadLine() passengers = Console.ReadLine()</pre> <p style="text-align: right;">(Part of B) (Part of B) (C) (D)</p>	7

```
fare = 2 * passengers (E)
fare = fare + (1.5 * distance) (F)
Console.WriteLine(fare) (G)
```

#### I. indentation in VB.NET

##### **Python Example 2 (partially correct – 6 marks)**

**Mark A** awarded. **Mark B** not awarded because float conversion missing.

```
dist = input() (C but NOT B)
pass = int(input()) (Part of B, D)
fare = 2 * pass (E)
fare = 1.5 * dist (F)
print fare (G – still awarded even though
parentheses missing in print command
as logic still clear)
```



Question	Part	Marking guidance	Total marks
06		<p><b>2 marks for AO3 (design), 3 marks for AO3 (program)</b></p> <p><b><u>Program Design</u></b>  <b>Mark A</b> for the use of a selection construct (even if the logic is incorrect);  <b>Mark B</b> for the correct, consistent use of meaningful variable names throughout (even if the code would not work);</p> <p><b><u>Program Logic</u></b>  <b>Mark C</b> for using user input and storing the result in a variable correctly;  <b>Mark D</b> for a correct expression that checks if the entered password is 'secret' (even if the syntax is incorrect);  <b>Mark E</b> for outputting Welcome and Not welcome correctly in logically separate places such as the IF and ELSE part of selection;</p> <p>I. Case of output strings for <b>Mark E</b>, but spelling must be correct.  I. Case of program code</p> <p><b>Maximum 4 marks</b> if any errors in code.</p> <p><b><u>Python Example 1 (fully correct)</u></b>  All design marks are achieved (<b>Marks A and B</b>)</p> <pre>password = input() if password == 'secret':     print('Welcome') else:     print('Not welcome')</pre> <p style="text-align: right;">(C) (D) <b>(Part of E)</b> <b>(Part of E)</b></p> <p><b><u>C# Example (fully correct)</u></b>  All design marks are achieved (<b>Marks A and B</b>)</p> <pre>string password; password = Console.ReadLine(); if (password == "secret") {     Console.WriteLine("Welcome"); } else {     Console.WriteLine("Not welcome"); }</pre> <p style="text-align: right;">(C) (D) <b>(Part of E)</b> <b>(Part of E)</b></p> <p>I. indentation in C#</p> <p><b><u>VB Example (fully correct)</u></b>  All design marks are achieved (<b>Marks A and B</b>)</p> <pre>Dim password As String password = Console.ReadLine()</pre> <p style="text-align: right;">(C)</p>	5

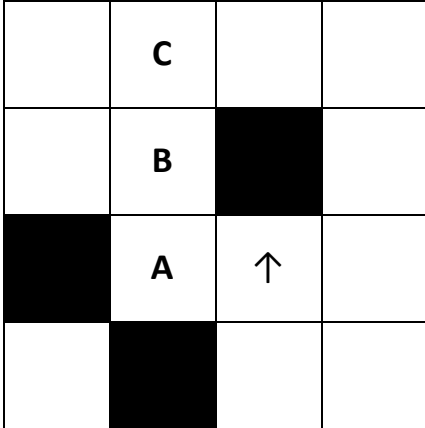
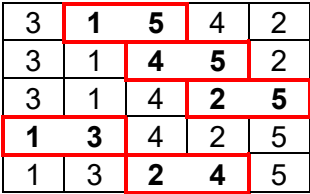
		<pre>If (password = "secret") Then     Console.WriteLine("Welcome") Else     Console.WriteLine("Not welcome") End If</pre> <p><b>I. indentation in VB.NET</b></p> <p><b><u>Python Example 2 (partially correct – 4 marks)</u></b>  <b>Mark A</b> is awarded. <b>Mark B</b> is not awarded.</p> <pre>p = input() if p == 'secret'     print('Welcome') else:     print('Not welcome')</pre>	<p>(D)  <b>(Part of E)</b></p> <p><b>(Part of E)</b></p> <p></p> <p>(C)  (D)  <b>(Part of E)</b></p> <p><b>(Part of E)</b></p>	
--	--	--	--	--

Question	Part	Marking guidance	Total marks																																												
07	1	<p><b>Mark is for AO2 (apply)</b></p> <p>Boolean//bool;</p> <p>I. Case</p>	1																																												
07	2	<p><b>2 marks for AO2 (apply)</b></p> <p>(The identifier) <code>swapsMade</code> describes the purpose//role//meaning of the variable; this makes the algorithm easier to understand//maintain//follow;</p> <p>or</p> <p>(The identifier) <code>s</code> does not describe the purpose//role//meaning of the variable; this makes the algorithm harder to understand//maintain//follow;</p>	2																																												
07	3	<p><b>Mark is for AO2 (apply)</b></p> <p><b>A</b> The algorithm uses a named constant;</p> <p><b>R.</b> If more than one lozenge shaded</p>	1																																												
07	4	<p><b>6 marks for AO2 (apply)</b></p> <p>1 mark for column <code>arr[0]</code> correct;</p> <p>1 mark for column <code>arr[1]</code> correct;</p> <p>1 mark for column <code>arr[2]</code> correct <b>only if</b> <code>arr[0]</code> and <code>arr[1]</code> are correct;</p> <p>1 mark for <code>swapsMade</code> column correct;</p> <p>1 mark for <code>i</code> column correct;</p> <p>1 mark for <code>t</code> column correct;</p> <table border="1" data-bbox="456 1503 1337 1890" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="3">Arr</th> <th rowspan="2">swapsMade</th> <th rowspan="2">i</th> <th rowspan="2">t</th> </tr> <tr> <th>0</th> <th>1</th> <th>2</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>1</td> <td>6</td> <td>false</td> <td rowspan="2">0</td> <td rowspan="4">4</td> </tr> <tr> <td>1</td> <td>4</td> <td></td> <td>true</td> </tr> <tr> <td></td> <td></td> <td></td> <td>false</td> <td>1</td> </tr> <tr> <td></td> <td></td> <td></td> <td>true</td> <td>2</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td>0</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td>1</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> <td>2</td> </tr> </tbody> </table> <p>I. different rows used as long as the order within columns is clear</p> <p>I. duplicate values on consecutive rows within a column</p>	Arr			swapsMade	i	t	0	1	2	4	1	6	false	0	4	1	4		true				false	1				true	2					0					1					2	6
Arr			swapsMade	i	t																																										
0	1	2																																													
4	1	6	false	0	4																																										
1	4		true																																												
			false	1																																											
			true	2																																											
				0																																											
				1																																											
				2																																											

Question	Part	Marking guidance	Total marks
08		<p><b>3 marks for AO3 (design), 4 marks for AO3 (program)</b></p> <p><b><u>Program Design</u></b>  <b>Mark A</b> for the idea of inputting a character and checking if it is lower case (even if the code would not work);  <b>Mark B</b> for the use of a selection construct (even if the logic is incorrect);  <b>Mark C</b> for the correct, consistent use of meaningful variable names throughout (even if the code would not work);</p> <p><b><u>Program Logic</u></b>  <b>Mark D</b> for using user input correctly;  <b>Mark E</b> for storing the result of user input in a variable correctly;  <b>Mark F</b> for a correct expression/method that checks if the character is lowercase;  <b>Mark G</b> for outputting LOWER and NOT LOWER correctly in logically separate places such as the IF and ELSE part of selection;</p> <p>I. Case of output strings for <b>Mark G</b>, but spelling must be correct.  I. Case of program code</p> <p><b>Maximum 6 marks</b> if any errors in code.</p> <p><b><u>Python Example 1 (fully correct)</u></b>  All design marks are achieved (<b>Marks A, B and C</b>)</p> <pre> character = input() if (character &gt;= 'a') and (character &lt;= 'z'):     print('LOWER') else:     print('NOT LOWER') </pre> <p style="text-align: right;">(D,E) (F) (Part of G) (Part of G)</p> <p><b><u>Python Example 2 (fully correct)</u></b>  All design marks are achieved (<b>Marks A, B and C</b>)</p> <pre> character = input() if character.islower():     print('LOWER') else:     print('NOT LOWER') </pre> <p style="text-align: right;">(D,E) (F) (Part of G) (Part of G)</p>	7

	<p><b><u>C# Example (fully correct)</u></b>                  All design marks are achieved (Marks A, B and C)</p> <pre> char character = (char)Console.Read(); if (Char.IsLower(character)) {     Console.WriteLine("LOWER"); } else {     Console.WriteLine("NOT LOWER"); }                 </pre> <p>(D,E) (F) (Part of G) (Part of G)</p> <p>I. indentation in C#</p> <p><b><u>VB.Net Example (fully correct)</u></b>                  All design marks are achieved (Marks A, B and C)</p> <pre> Dim character As Char character = Console.ReadLine() If (Char.IsLower(character)) Then     Console.WriteLine("LOWER") Else     Console.WriteLine("NOT LOWER") End If                 </pre> <p>(D,E) (F) (Part of G) (Part of G)</p> <p>I. indentation in VB.NET</p>	
	<p><b><u>Python Example 3 (partially correct – 5 marks)</u></b>                  All design marks are achieved (Marks A, B and C)</p> <pre> character = input() if (character &gt; 'a') or (character &lt; 'z'):     print('NOT LOWER') else:     print('LOWER')                 </pre> <p>(D,E) (NOT F) (NOT G) (NOT G)</p>	

Question	Part	Marking guidance	Total marks																
09	1	<p><b>3 marks for AO2 (apply)</b></p> <p><b>Mark as follows:</b></p> <p><b>1 mark</b> for the robot moving to <b>both</b> squares marked <b>A</b>;  <b>1 mark</b> for the robot moving to the square marked <b>B</b>;  <b>1 mark</b> for the robot moving to the square marked <b>C</b>;</p> <table border="1" data-bbox="694 595 1118 1021"> <tbody> <tr> <td></td> <td></td> <td><b>C</b></td> <td></td> </tr> <tr> <td></td> <td></td> <td><b>B</b></td> <td><b>A</b></td> </tr> <tr> <td></td> <td></td> <td></td> <td><b>A</b></td> </tr> <tr> <td></td> <td></td> <td></td> <td>↑</td> </tr> </tbody> </table>			<b>C</b>				<b>B</b>	<b>A</b>				<b>A</b>				↑	3
		<b>C</b>																	
		<b>B</b>	<b>A</b>																
			<b>A</b>																
			↑																

Question	Part	Marking guidance	Total marks
09	2	<p><b>3 marks for AO2 (apply)</b></p> <p><b>Mark as follows:</b></p> <p><b>1 mark</b> for the robot moving to the square marked <b>A</b>;  <b>1 mark</b> for the robot moving to the square marked <b>B</b>;  <b>1 mark</b> for the robot moving to the square marked <b>C</b>;</p> 	3
10		<p><b>2 Marks for AO1 (understanding)</b></p> <p>Max 2 marks from:</p> <p>Subroutines can be developed in isolation/independently/separately;  Easier to discover errors/testing is more effective (than without a structure);  Subroutines can be updated without affecting the overall program;</p> <p><b>A.</b> Other valid reasons</p>	2
11		<p><b>5 marks for AO2 (apply)</b></p> <p><b>1 mark</b> for each correct change (allow follow on);</p> <p>The correct sequence is:</p> 	5
12	1	<p><b>1 mark for AO1 (recall)</b></p> <p><b>A</b> Abstraction;</p> <p><b>R.</b> if more than one lozenge shaded</p>	1





Question	Part	Marking guidance	Total marks
13	3	<p><b>4 marks for AO3 (design)</b></p> <p><b>Mark A</b> for using a <code>WHILE</code> loop or similar to move from column 0 to column 2;</p> <p><b>Mark B</b> for a Boolean condition that detects when column 0 is empty;</p> <p><b>Mark C</b> for using a second <code>WHILE</code> 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);</p> <p><b>or</b></p> <p><b>Mark A</b> for using a <code>FOR</code> loop or similar to move from column 0 to column 2;</p> <p><b>Mark B</b> for ascertaining the terminating value for the <code>FOR</code> loop;</p> <p><b>Mark C</b> for using a second <code>FOR</code> 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);</p> <p><b>and</b></p> <p><b>Mark D</b> for using the subroutines correctly throughout, i.e. called with appropriate parameters and return values handled correctly;</p> <p><b>A.</b> Minor spelling errors such as <code>HIEGHT</code> for <code>HEIGHT</code></p> <p><b>I.</b> Case</p> <p><b>Example 1</b></p> <pre> WHILE HEIGHT(0) &gt; 0           (Part of A, B)     MOVE(0, 2)                 (Part of A) ENDWHILE WHILE HEIGHT(2) &gt; 0           (Part of C)     MOVE(2, 1)                 (Part of C) ENDWHILE                 </pre> <p>(<code>MOVE</code> and <code>HEIGHT</code> are used correctly throughout so <b>D.</b>)</p> <p><b>Example 2</b></p> <pre> DO                               (Part of A)     MOVE(0, 2)                   (Part of A) WHILE HEIGHT(0) &gt; 0           (Part of A, B) DO                               (Part of C)     MOVE(2, 1)                   (Part of C) WHILE HEIGHT(2) &gt; 0           (Part of C)                 </pre> <p>(<code>MOVE</code> and <code>HEIGHT</code> are used correctly throughout so <b>D.</b>)</p>	4

	<p><b>Example 3</b></p> <pre> REPEAT   MOVE (0, 2) UNTIL HEIGHT(0) = 0 REPEAT   MOVE (2, 1) WHILE HEIGHT(2) = 0                     </pre> <p>(MOVE and HEIGHT are used correctly throughout so D.)</p>	<p>(Part of A)  (Part of A)  (Part of A, B)  (Part of C)  (Part of C)  (Part of C)</p>
	<p><b>Example 4</b></p> <pre> number_of_blocks ← HEIGHT(0) FOR x ← 0 TO number_of_blocks   MOVE (0, 2) ENDFOR FOR x ← 0 TO number_of_blocks   MOVE (2, 1) ENDFOR                     </pre> <p>(MOVE and HEIGHT are used correctly throughout so D.)</p>	<p>(Part of B)  (Part of A, Part of B)  (Part of A)    (Part of C)  (Part of C)  (Part of C)</p>
	<p><b>Example 5</b></p> <pre> graph TD     START([START]) --&gt; MOVE0[MOVE (0, 2)]     MOVE0 --&gt; DEC0{HEIGHT(0) &gt; 0}     DEC0 -- Y --&gt; MOVE0     DEC0 -- N --&gt; MOVE2[MOVE (2, 1)]     MOVE2 --&gt; DEC2{HEIGHT(2) &gt; 0}     DEC2 -- Y --&gt; MOVE2     DEC2 -- N --&gt; STOP([STOP])                     </pre> <p>(MOVE and HEIGHT are used correctly throughout so D.)</p>	

Question	Part	Marking guidance	Total marks
14		<p><b>1 mark for AO3 (refine)</b></p> <p>B;</p> <p>R. if more than 1 lozenge shaded</p>	1
15		<p><b>4 marks for AO3 (refine)</b></p> <p><b>Program Logic</b></p> <p><b>Mark A:</b> for using a selection structure with else part <b>or</b> two selection structures (even if the syntax is incorrect)</p> <p><b>Mark B:</b> for correct condition(s) in selection statement(s) (even if the syntax is incorrect)</p> <p><b>Mark C:</b> for statement that subtracts two from odd under the correct conditions (even if the syntax is incorrect)</p> <p><b>Mark D:</b> for odd being output and doing one of adding or subtracting two but not both each time loop repeats (even if the syntax is incorrect)</p> <p>I. while loop from question if included in answer I. case of program code</p> <p><b>Maximum 3 marks</b> if any errors in code.</p> <p><b><u>Python Example 1 (fully correct)</u></b></p> <pre>print(odd) if number &lt; 0     odd = odd - 2 else:     odd = odd + 2</pre> <p>(Part of D) (A, B) (C, Part of D) (Part of D)</p> <p><b><u>C# Example (fully correct)</u></b></p> <pre>Console.WriteLine(odd); if (number &lt; 0) {     odd = odd - 2; } else {     odd = odd + 2; }</pre> <p>(Part of D) (A, B) (C, Part of D) (Part of D)</p> <p>I. indentation in C#</p>	4

		<p><b><u>VB.Net Example (fully correct)</u></b></p> <pre> Console.WriteLine(odd) If number &lt; 0 Then     odd = odd - 2 Else     odd = odd + 2 End If                 </pre> <p><b>I. indentation in VB.Net</b></p>	<p><b>(Part of D)</b>  <b>(A, B)</b>  <b>(C, Part of D)</b>    <b>(Part of D)</b></p>
		<p><b><u>Python Example 2 (partially correct – 3 marks)</u></b></p> <pre> print(odd) if number != 0     odd = odd - 2 else:     odd = odd + 2                 </pre>	<p><b>(Part of D)</b>  <b>(A, NOT B)</b>  <b>(C, Part of D)</b>    <b>(Part of D)</b></p>

16		<p><b>2 marks for AO3 (test)</b></p> <table border="1"> <thead> <tr> <th>Test type</th> <th>Test data</th> <th>Expected result</th> </tr> </thead> <tbody> <tr> <td>Normal data</td> <td>5</td> <td>Valid choice message displayed</td> </tr> <tr> <td>Invalid data</td> <td>Any value other than the numbers 1 to 10 inclusive</td> <td>Invalid choice (message displayed)</td> </tr> <tr> <td>Boundary data</td> <td>Any one of 0, 1, 10 or 11</td> <td>if 1 or 10 given as test data Valid choice (message displayed)  if 0 or 11 given as test data Invalid choice (message displayed)</td> </tr> </tbody> </table> <p>1 mark for each completely correct row to a maximum of 2 marks.</p>	Test type	Test data	Expected result	Normal data	5	Valid choice message displayed	Invalid data	Any value other than the numbers 1 to 10 inclusive	Invalid choice (message displayed)	Boundary data	Any one of 0, 1, 10 or 11	if 1 or 10 given as test data Valid choice (message displayed)  if 0 or 11 given as test data Invalid choice (message displayed)	2
Test type	Test data	Expected result													
Normal data	5	Valid choice message displayed													
Invalid data	Any value other than the numbers 1 to 10 inclusive	Invalid choice (message displayed)													
Boundary data	Any one of 0, 1, 10 or 11	if 1 or 10 given as test data Valid choice (message displayed)  if 0 or 11 given as test data Invalid choice (message displayed)													

17	1	<p><b>1 mark for AO3 (test)</b></p> <p>2;</p>	1
----	---	---	---

17	2	<p><b>1 mark for AO3 (test)</b></p> <p>5;</p>	1
----	---	---	---

17	3	<p><b>1 mark for AO3 (refine)</b></p> <p>Change the &lt; sign to &lt;= // change num1 to num1 + 1;</p> <p><b>A. answers where line of code has been rewritten</b></p>	1
----	---	---	---

Question	Part	Marking guidance	Total marks
18		<p><b>2 marks for AO3 (design) and 6 marks for AO3 (program)</b></p> <p><b><u>Program Design</u></b>  <b>Mark A</b> for using an iterative structure to validate the user input of speed (even if logic is incorrect);  <b>Mark B</b> for using meaningful variable names <b>and</b> suitable data types throughout (speed can be real or integer, braking distance must be real, the IsWet input must be string);</p> <p><b><u>Program Logic</u></b>  <b>Mark C</b> for getting user input for <b>both</b> the speed and IsWet in appropriate places;  <b>Mark D</b> for using a WHILE loop or similar to re-prompt for the user input (even if it would not work);  <b>Mark E</b> for using a correct Boolean condition with the validation structure;  <b>Mark F</b> for calculating the braking distance correctly (i.e. divided by 5);  <b>Mark G</b> for using a selection structure to adjust the braking distance calculation if the user input required it (even if it would not work);  <b>Mark H</b> for outputting the braking distance in a logically correct place;</p> <p>I. Case of program code</p> <p><b>Maximum 7 marks</b> if any errors in code.</p> <p><b><u>Python Example (fully correct)</u></b>  All design marks are achieved (<b>Marks A and B</b>)</p> <pre> speed = float(input()) while speed &lt; 10 or speed &gt; 50:     speed = float(input()) braking_distance = speed / 5  IsWet = input() if IsWet == 'yes':     braking_distance = braking_distance * 1.5 print(braking_distance) </pre> <p style="text-align: right;"> <b>(Part of C)</b>  <b>(D, E)</b>  <b>(Part of D)</b>  <b>(F)</b>    <b>(Part of C)</b>  <b>(Part of G)</b>  <b>(Part of G)</b>  <b>(H)</b> </p>	8

	<p><b><u>C# Example (fully correct)</u></b>  <b>All design marks are achieved (Marks A and B)</b></p> <pre> int intSpeed; double braking_distance; string IsWet; intSpeed = int.Parse(Console.ReadLine()); while (intSpeed &lt; 10    intSpeed &gt; 50) {     intSpeed = int.Parse(Console.ReadLine()); } braking_distance = (double)intSpeed / 5; IsWet = Console.ReadLine(); if (IsWet == "yes") {     braking_distance = braking_distance * 1.5; } Console.WriteLine(braking_distance); </pre> <p><b>I. indentation in C#</b></p> <p><b><u>VB Example (fully correct)</u></b>  <b>All design marks are achieved (Marks A and B)</b></p> <pre> Dim speed As Integer Dim braking_distance As Decimal Dim IsWet As String speed = Console.ReadLine() while speed &lt; 10 Or speed &gt; 50     speed = Console.ReadLine() End While braking_distance = speed / 5 IsWet = Console.ReadLine() if IsWet = "yes" Then     braking_distance = braking_distance * 1.5 End If Console.WriteLine(braking_distance) </pre> <p><b>I. indentation in VB.Net</b></p>	<p><b>(Part of C)</b>  <b>(D, E)</b></p> <p><b>(Part of D)</b></p> <p><b>(F)</b>  <b>(Part of C)</b>  <b>(Part of G)</b></p> <p><b>(Part of G)</b></p> <p><b>(H)</b></p> <p><b>(Part of C)</b>  <b>(D, E)</b>  <b>(Part of D)</b></p> <p><b>(F)</b>  <b>(Part of C)</b>  <b>(Part of G)</b>  <b>(Part of G)</b></p> <p><b>(H)</b></p>
--	---	---

		<p><b>Python Example (partially correct – 7 marks)</b>  <b>All design marks are achieved (Marks A and B)</b></p> <pre> speed = float(input()) while speed &lt;= 10 and speed &gt; 50     speed = float(input())     braking_distance = speed / 5  IsWet = input() if IsWet = 'yes'     braking_distance = braking_distance * 1.5 print(braking_distance) </pre>	<p><b>(Part of C)</b>  <b>(D, NOT E)</b>  <b>(Part of D)</b>  <b>(F)</b></p> <p><b>(Part of C)</b>  <b>(Part of G)</b>  <b>(Part of G)</b>  <b>(H)</b></p>
--	--	---	--

**Copyright information**

AQA retains the copyright on all its publications. However, registered schools/colleges for AQA are permitted to copy material from this booklet for their own internal use, with the following important exception: AQA cannot give permission to schools/colleges to photocopy any material that is acknowledged to a third party even for internal use within the centre.