AQA

GCSE

COMPUTER SCIENCE

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

Paper 1 Computational thinking and programming skills

Mark scheme

Specimen Assessment Materials

Programming skills– VB.Net

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.

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.

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

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 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>
<tr><td>02</td><td>1</td><td>Mark is for AO2 (apply)

D 4;

R. If more than one lozenge shaded</td><td>1</td></tr>
<tr><td>02</td><td>2</td><td>Mark is for AO2 (apply)

D<code>'computer sciencegcse'</code>;

R. If more than one lozenge shaded</td><td>1</td></tr>
<tr><td>03</td><td>1</td><td>Mark is for AO2 (apply)

A Line number 2;

R. If more than one lozenge shaded</td><td>1</td></tr>
<tr><td>03</td><td>2</td><td>Mark is for AO2 (apply)

C Line number 11;

R. If more than one lozenge shaded</td><td>1</td></tr>
</table>

Question	Part	Marking guidance	Total marks
03	3	**Mark is for AO2 (apply)** **A** 1 subroutine call; **R.** If more than one lozenge shaded	1
03	4	**Mark is for AO2 (apply)** **B** String; **R.** If more than one lozenge shaded	1
03	5	**Mark is for AO2 (apply)** 2//twice//two;	1

04		**5 marks for AO3 (program)**
		
		1 mark for each correct item in the correct location.
		
		Python
		
		`num1 = int(input("Enter a number: "))`
		`num2 = int (input("Enter a second number: "))`
		`if num1 > num2:`
		`  print(" num1 is bigger.")`
		`elif num1 < num2:`
		`  print(" num2 is bigger.")`
		`else:`
		`  print("The numbers are equal.")`
		
		I. Case of response
		R. if any spelling mistakes
		
		C#
		
		`int num1;`
		`int num2;`
		
		`Console.WriteLine("Enter a number: ");`
		
		`num1 = int.Parse(Console.ReadLine());`
		
		`Console.WriteLine("Enter another number: ");`
		
		`num2 = int.Parse(Console.ReadLine());`
			5

```
if (num1 > num2)
{
    Console.WriteLine("    num1    is bigger.");
}
else
if (num1 <          num2)
{
    Console.WriteLine("    num2    is bigger.");
}
else
{
    Console.WriteLine("The numbers are equal.");
}
```

I. Case of response
R. if any spelling mistakes

VB.Net

```
Dim num1 As Integer
Dim num2 As  Integer

Console.Write("Enter a number: ")

num1 = Console.ReadLine()

Console.Write("Enter another number: ")

num2 = Console.ReadLine()
If num1 > num2 Then
    Console.WriteLine(" num1     is bigger.")
ElseIf num1  <          num2 Then
    Console.WriteLine(" num2     is bigger.")
Else
    Console.WriteLine("The numbers are equal.")
End If
```

I. Case of response
R. if any spelling mistakes

Question	Part	Marking guidance	Total marks
05		**2 marks for AO3 (design) and 5 marks for AO3 (program)** **Program Design** **Mark A** for using meaningful variable names throughout (even if logic is incorrect); **Mark B** for using suitable data types throughout (distance can be real or integer, passengers must be integer); **Program Logic** **Mark C** for getting user input for the distance in an appropriate place; **Mark D** for getting user input for the number of passengers in an appropriate place; **Mark E** for a fare that correctly charges £2 per passenger; **Mark F** for a fare that correctly charges £1.50 for every kilometre; **Mark G** for outputting the correct final fare; **I.** Case of program code **Maximum 6 marks** if any errors in code. **Python Example 1 (fully correct)** **Mark A** awarded. `distance = float(input())` (**Part of B, C**) `passengers = int(input())` (**Part of B, D**) `fare = 2 * passengers` (**E**) `fare = fare + (1.5 * distance)` (**F**) `print(fare)` (**G**) **C# Example (fully correct)** **Mark A** awarded. `int passengers;` (**Part of B**) `double distance, fare;` (**Part of B**) `distance = double.Parse(Console.ReadLine());` (**C**) `passengers = int.Parse(Console.ReadLine());` (**D**) `fare = 2 * passengers;` (**E**) `fare = fare + (1.5 * distance);` (**F**) `Console.WriteLine(fare);` (**G**) **I.** indentation in C# **VB Example (fully correct)** **Marks A**, **B** awarded. `Dim distance, fare As Double` (**Part of B**) `Dim passengers As Integer` (**Part of B**) `distance = Console.ReadLine()` (**C**) `passengers = Console.ReadLine()` (**D**)	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		**2 marks for AO3 (design), 3 marks for AO3 (program)**
		
		Program Design
		Mark A for the use of a selection construct (even if the logic is incorrect);
		Mark B for the correct, consistent use of meaningful variable names throughout (even if the code would not work);
		
		Program Logic
		Mark C for using user input and storing the result in a variable correctly;
		Mark D for a correct expression that checks if the entered password is `'secret'` (even if the syntax is incorrect);
		Mark E for outputting `Welcome` and `Not welcome` correctly in logically separate places such as the `IF` and `ELSE` part of selection;
		
		I. Case of output strings for **Mark E**, but spelling must be correct.
		I. Case of program code
		
		Maximum 4 marks if any errors in code.
		
		Python Example 1 (fully correct)
		All design marks are achieved (**Marks A and B**)
		
		`password = input()` (**C**)
		`if password == 'secret':` (**D**)
		`    print('Welcome')` (**Part of E**)
		`else:`
		`    print('Not welcome')` (**Part of E**)
		
		C# Example (fully correct)
		All design marks are achieved (**Marks A and B**)
		
		`string password;`
		`password = Console.ReadLine();` (**C**)
		`if (password == "secret")` (**D**)
		`{`
		`   Console.WriteLine("Welcome");` (**Part of E**)
		`}`
		`else`
		`{`
		`   Console.WriteLine("Not welcome");` (**Part of E**)
		`}`
		
		I. indentation in C#
		
		VB Example (fully correct)
		All design marks are achieved (**Marks A and B**)
		
		`Dim password As String`
		`password = Console.ReadLine()` (**C**)
			5

```
If (password = "secret") Then                    (D)
   Console.WriteLine("Welcome")                  (Part of E)
Else
   Console.WriteLine("Not welcome")              (Part of E)
End If
```

I. indentation in VB.NET

Python Example 2 (partially correct – 4 marks)
Mark A is awarded. **Mark B** is not awarded.

```
p = input()                                      (C)
if p == 'secret'                                 (D)
    print('Welcome')                             (Part of E)
    else:
    print('Not welcome')                         (Part of E)
```

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>07</td><td>1</td><td>Mark is for AO2 (apply)

Boolean//bool;

I. Case</td><td>1</td></tr>
<tr><td>07</td><td>2</td><td>2 marks for AO2 (apply)

(The identifier) <code>swapsMade</code> describes the purpose//role//meaning of the variable;
this makes the algorithm easier to understand//maintain//follow;

or

(The identifier) <code>s</code> does not describe the purpose//role//meaning of the variable;
this makes the algorithm harder to understand//maintain//follow;</td><td>2</td></tr>
<tr><td>07</td><td>3</td><td>Mark is for AO2 (apply)

A The algorithm uses a named constant;

R. If more than one lozenge shaded</td><td>1</td></tr>
<tr><td>07</td><td>4</td><td>6 marks for AO2 (apply)

1 mark for column <code>arr[0]</code> correct;
1 mark for column <code>arr[1]</code> correct;
1 mark for column <code>arr[2]</code> correct only if <code>arr[0]</code> and <code>arr[1]</code> are correct;
1 mark for <code>swapsMade</code> column correct;
1 mark for <code>i</code> column correct;
1 mark for <code>t</code> column correct;

<table>
<tr><th colspan="3">Arr</th><th rowspan="2">swapsMade</th><th>i</th><th>t</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th></th><th></th></tr>
<tr><td>4</td><td>1</td><td>6</td><td>false</td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td>true</td><td>0</td><td></td></tr>
<tr><td>1</td><td>4</td><td></td><td>false</td><td></td><td>4</td></tr>
<tr><td></td><td></td><td></td><td></td><td>1</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>2</td><td></td></tr>
<tr><td></td><td></td><td></td><td>true</td><td>0</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>1</td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td>2</td><td></td></tr>
</table>

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

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

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

```
char character = (char)Console.Read();          (D,E)
if (Char.IsLower(character))                    (F)
{
 Console.WriteLine("LOWER");                    (Part of G)
}
else
{
 Console.WriteLine("NOT LOWER");                (Part of G)
}
```

I. indentation in C#

VB.Net Example (fully correct)
All design marks are achieved (**Marks A, B and C**)

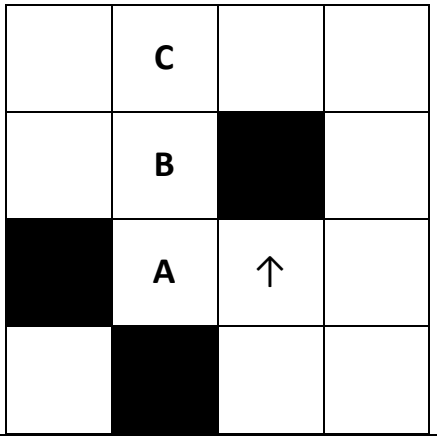
```
Dim character As Char
character = Console.ReadLine()                  (D,E)
If (Char.IsLower(character)) Then               (F)
  Console.WriteLine("LOWER")                    (Part of G)
Else
  Console.WriteLine("NOT LOWER")                (Part of G)
End If
```

I. indentation in VB.NET

Python Example 3 (partially correct – 5 marks)
All design marks are achieved (**Marks A, B and C**)

```
character = input()                             (D,E)
if (character > 'a') or (character < 'z'):      (NOT F)
   print('NOT LOWER')                           (NOT G)
else:
   print('LOWER')                               (NOT G)
```

Question	Part	Marking guidance	Total marks
09	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**; **C** **B** **A** **A** ↑	3

Question	Part	Marking guidance	Total marks
09	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
10		**2 Marks for AO1 (understanding)** Max 2 marks from: 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; **A.** Other valid reasons	2
11		**5 marks for AO2 (apply)** **1 mark** for each correct change (allow follow on); The correct sequence is: 3 **1** **5** 4 2 3 1 **4** **5** 2 3 1 4 **2** **5** **1** **3** 4 2 5 1 3 **2** **4** 5	5
12	1	**1 mark for AO1 (recall)** **A** Abstraction; **R.** if more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
12	2	**2 marks for AO2 (apply)** All friends have different first names; The time is rounded up to the nearest half-hour;	2
13	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
13	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
13	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)
```

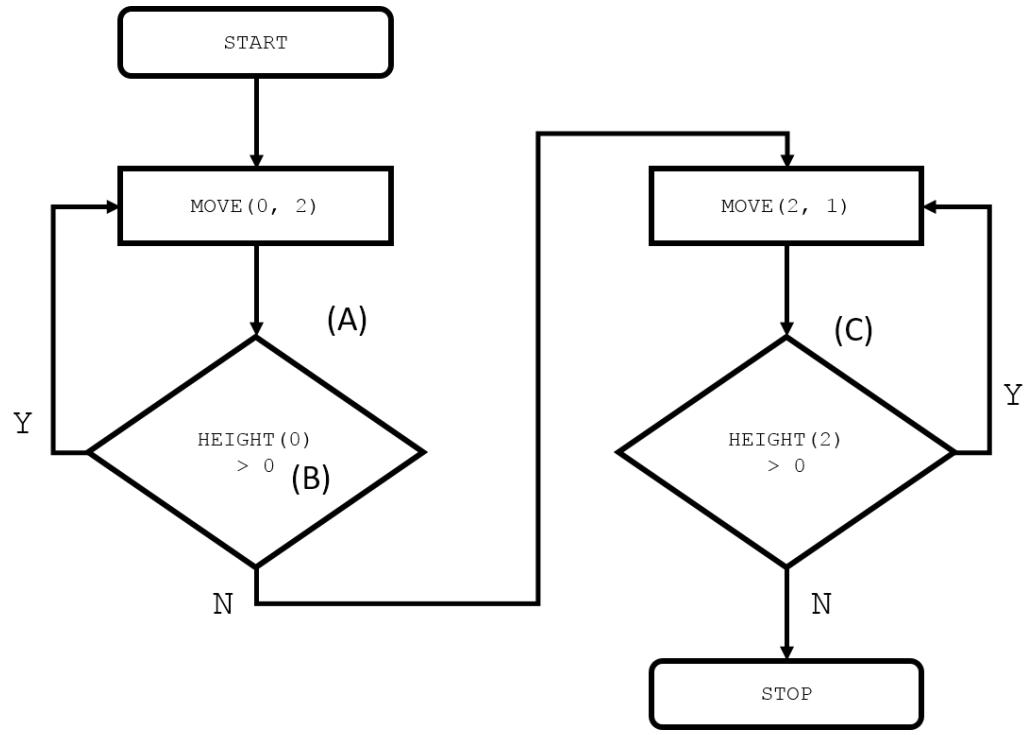
(MOVE and HEIGHT are used correctly throughout so **D**.)

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
14		**1 mark for AO3 (refine)** B; **R.** if more than 1 lozenge shaded	1

15		**4 marks for AO3 (refine)**
		
		Program Logic
		Mark A: for using a selection structure with else part **or** two selection structures (even if the syntax is incorrect)
		Mark B: for correct condition(s) in selection statement(s) (even if the syntax is incorrect)
		Mark C: for statement that subtracts two from `odd` under the correct conditions (even if the syntax is incorrect)
		Mark D: 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)
		
		I. while loop from question if included in answer
		I. case of program code
		
		Maximum 3 marks if any errors in code.
		
		Python Example 1 (fully correct)
		
		`print(odd)` (**Part of D**)
		`if number < 0` (**A, B**)
		`  odd = odd - 2` (**C, Part of D**)
		`else:`
		`  odd = odd + 2` (**Part of D**)
		
		C# Example (fully correct)
		
		`Console.WriteLine(odd);` (**Part of D**)
		`if (number < 0)` (**A, B**)
		`{`
		`  odd = odd - 2;` (**C, Part of D**)
		`}`
		`else`
		`{`
		`  odd = odd + 2;` (**Part of D**)
		`}`
		
		I. indentation in C#
			4

		VB.Net Example (fully correct)
		
		`Console.WriteLine(odd)` (**Part of D**)
		`If number < 0 Then` (**A, B**)
		`  odd = odd - 2` (**C, Part of D**)
		`Else`
		`  odd = odd + 2` (**Part of D**)
		`End If`
		
		I. indentation in VB.Net	

		Python Example 2 (partially correct – 3 marks) `print(odd)` (**Part of D**) `if number != 0` (**A, NOT B**) `  odd = odd - 2` (**C, Part of D**) `else:` `  odd = odd + 2` (**Part of D**)	

16		**2 marks for AO3 (test)** **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) 1 mark for each completely correct row to a maximum of 2 marks.	2

17	1	**1 mark for AO3 (test)** 2;	1

17	2	**1 mark for AO3 (test)** 5;	1

17	3	**1 mark for AO3 (refine)** Change the < sign to <= // change `num1` to `num1 + 1`; **A.** answers where line of code has been rewritten	1

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

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

```
int intSpeed;
double braking_distance;
string IsWet;
intSpeed = int.Parse(Console.ReadLine());          (Part of C)
while (intSpeed < 10 || intSpeed > 50)             (D, E)
{
   intSpeed = int.Parse(Console.ReadLine());       (Part of D)
}
braking_distance = (double)intSpeed / 5;           (F)
IsWet = Console.ReadLine();                        (Part of C)
if (IsWet == "yes")                                (Part of G)
{
   braking_distance = braking_distance * 1.5;      (Part of G)
}
Console.WriteLine(braking_distance);               (H)
```

I. indentation in C#

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

```
Dim speed As Integer
Dim braking_distance As Decimal
Dim IsWet As String
speed = Console.ReadLine()                         (Part of C)
while speed < 10 Or speed > 50                     (D, E)
   speed = Console.ReadLine()                      (Part of D)
End While
braking_distance = speed / 5                       (F)
IsWet = Console.ReadLine()                         (Part of C)
if IsWet = "yes" Then                              (Part of G)
   braking_distance = braking_distance * 1.5       (Part of G)
End If
Console.WriteLine(braking_distance)                (H)
```

I. indentation in VB.Net

Python Example (partially correct – 7 marks)

All design marks are achieved (**Marks A and B**)

```
speed = float(input())                                  (Part of C)
while speed <= 10 and speed > 50                        (D, NOT E)
   speed = float(input())                               (Part of D)
   braking_distance = speed / 5                         (F)

IsWet = input()                                         (Part of C)
if IsWet = 'yes'                                        (Part of G)
   braking_distance = braking_distance * 1.5            (Part of G)
print(braking_distance)                                 (H)
```