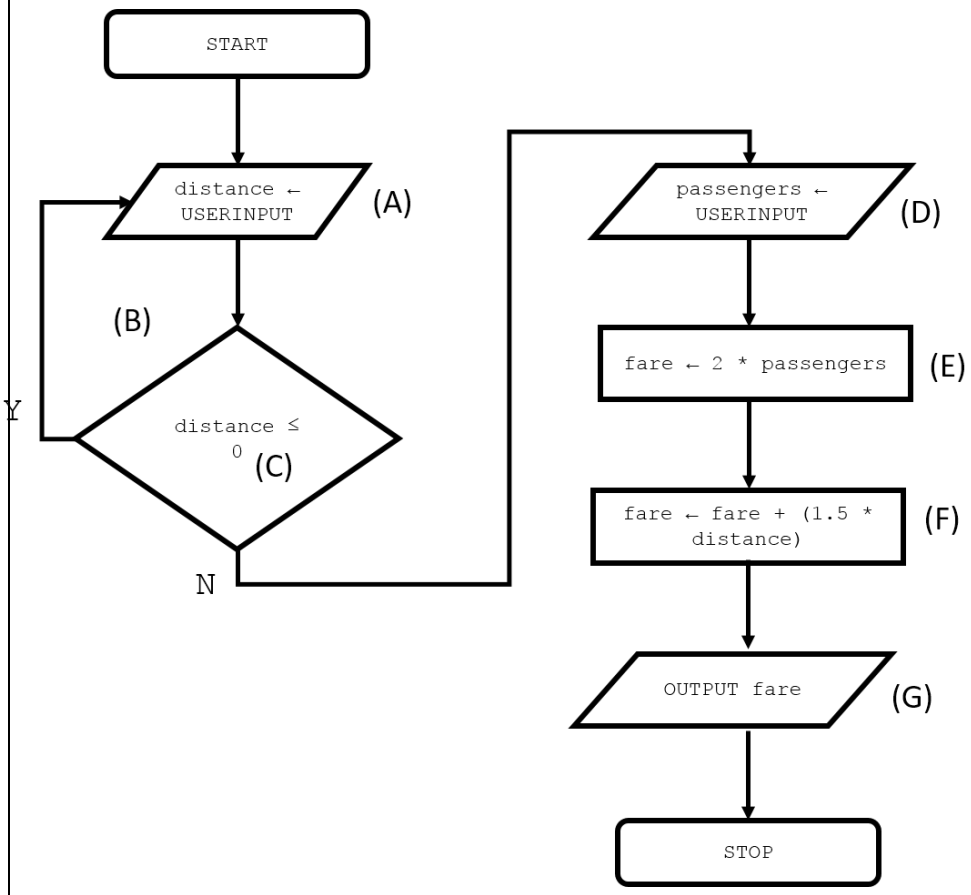
Question	Part	Marking guidance	Total marks
01		**8 marks for AO3 (program)** **DPT.** For repeated errors in user input and variable assignment. Mark A for getting user input for the distance and storing in a variable; Mark B for using a WHILE loop or similar to re-prompt for and re-assign the user input; 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
02	1	**Mark is for AO2 (apply)** **C** `flourNeeded ← eggsUsed * 100;` **If more than one lozenge shaded then mark is not awarded**	1

Question	Part	Marking guidance	Total marks

03	1	**Mark is for AO3 (refine)**
		
		C#
		`string displayMessage = carReg + " is not valid";`
		
		Python
		`displayMessage = carReg + " is not valid"`
		
		VB.NET
		`Dim displayMessage As String = carReg + " is not valid"` //
		`Dim displayMessage As String = carReg & " is not valid"`
		
		I. Case
		I. Space between variable outputs
		I. Order of strings
			1

Question	Part	Marking guidance	Total marks

03	2	**Mark is for AO3 (refine)**
		
		C#
		`charge = hours * 2 + 2;` //
		`charge = 2 + hours * 2;`
		
		Python
		`charge = hours * 2 + 2` //
		`charge = 2 + hours * 2`
		
		VB.NET
		`charge = hours * 2 + 2` //
		`charge = 2 + hours * 2`
		
		I. Case
		I. Parentheses, unless altering result eg, `hours * (2 + 2)`
			1

Question	Part	Marking guidance	Total marks

04		**3 marks for AO3 (design) and 4 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;
		Mark B for the use of a selection construct;
		Mark C for the use of a nested selection construct or multiple conditions;
		
		Program Logic
		Mark D for using user input and storing the result in two variables correctly for the items sold **and** years of employment;
		Mark E for correct expression that checks the years entered against the criteria for years employed;
		Mark F for correct Boolean expressions throughout;
		Mark G for outputting correct bonus depending on inputs entered in logically separate places such as IF, ELSE part of selection;
		
		I. Case
		I. Prompts
		
		Maximum 6 marks if any errors in code
		
		C# Example 1 (fully correct)
		
		`Console.Write("How many items?: ");`
		`int items = Convert.ToInt32(Console.ReadLine());` **(Part of A, D)**
		`Console.Write("How many years employed?: ");`
		`int years = Convert.ToInt32(Console.ReadLine());` **(Part of A, D)**
		`if (years <= 2) {` **(Part of B, E)**
		`  if (items > 100) {` **(Part of C, F)**
		`  Console.WriteLine(items * 2);` **(Part of G)**
		`  }`
		`  else {` **(Part of B, E)**
		`  Console.WriteLine(0);` **(Part of G)**
		`  }`
		`}`
		`else {` **(Part of B, E)**
		`  Console.WriteLine(items * 10);` **(Part of G)**
		`}`
			7

Python Example 1 (fully correct)

```
items = int(input("How many items?: "))                    (Part of A, D)
years = int(input("How many years employed?: "))           (Part of A, D)
if years <= 2:                                             (Part of B, E)
    if items > 100:                                        (Part of C, F)
        print(items * 2)                                   (Part of G)
    else:                                                  (Part of C, F)
        print(0)                                           (Part of G)
else:                                                      (Part of B, E)
    print(items * 10)                                      (Part of G)
```

Python Example 2 (fully correct)

```
items = int(input("Enter items: "))                        (Part of A, D)
years = int(input("Enter years employed: "))               (Part of A, D)
if years <= 2 and items > 100:                             (Part of B, C, E, F)
   print(items * 2)                                        (Part of G)
elif years > 2:                                            (Part of B, C, E, F)
   print(items * 10)                                       (Part of G)
else:                                                      (Part of B, E)
   print(0)                                                (Part of G)
```

VB.NET Example 1 (fully correct)

```
Console.Write("Enter items: ")
Dim items As Integer = Console.ReadLine()                  (Part of A, D)
Console.Write("Enter years: ")
Dim years As Integer = Console.ReadLine()                  (Part of A, D)
If years <= 2 And items > 100 Then                         (Part of B, C, E, F)
   Console.WriteLine(items * 2)                            (Part of G)
ElseIf years > 2 Then                                      (Part of B, C, E, F)
   Console.WriteLine(items * 10)                           (Part of G)
Else                                                       (Part of B, E)
   Console.WriteLine(0)                                    (Part of G)
End If
```

<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>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>05</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>06</td><td></td><td>2 marks for AO3 (design), 4 marks for AO3 (program)

<u>Program Design</u>
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 inputting the number in the group and storing in a variable;
Mark B for using selection;

<u>Program Logic</u>

Mark C for correctly multiplying the number in the group by 15;
Mark D for using an appropriate correct Boolean condition(s) that covers all paths through the problem, eg >=6 // >5 or equivalent;
Mark E for using an appropriate method to reduce the total charge by £5;
Mark F for outputting the final total in a logical place;

Maximum 5 marks if any errors in code.

I. Case
I. Messages or no messages with input statements
I. Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect.</td><td>6</td></tr>
<tr><td></td><td></td><td><u>C# Example 1 (fully correct)</u>
All design marks are achieved (Marks A and B)

<pre>int group = Convert.ToInt32(Console.ReadLine());
int total = group * 15;   (C)
if (group >= 6) {   (D)
   total = total - 5;   (E)
}
Console.WriteLine(total);   (F)</pre>
I. Indentation in C#
A. Write in place of WriteLine

<u>Python Example 1 (fully correct)</u>
All design marks are achieved (Marks A and B)

<pre>group = int(input())
total = group * 15   (C)
if group >= 6:   (D)
  total = total - 5   (E)
print(total)   (F)</pre></td><td></td></tr>
</table>

VB.NET Example 1 (fully correct)

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

```
Dim group As Integer = Console.ReadLine()
Dim total As Integer = group * 15                    (C)
If (group >= 6) Then                                 (D)
   total = total - 5                                 (E)
End If
Console.WriteLine(total)                             (F)
```

I. Indentation in VB.NET

A. `Write` in place of `WriteLine`

Question	Part	Marking guidance	Total marks
07		**3 marks for AO3 (design), 5 marks for AO3 (program)** Any solution that does not map to the mark scheme refer to lead examiner **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 storing a user input in a variable with a meaningful name; **Mark B** for using an iteration structure which attempts to pay the bill; **Mark C** for using a selection structure with `ELSE` / `ELSEIF` // use of multiple selection constructs; **Program Logic** **Mark D** for getting the user input for the total amount of the bill (outside the loop) **AND** deducting a payment towards the bill (within the loop); **A.** if there is no loop and both elements are present in the right order. **Mark E** for a mechanism which will correctly terminate the iteration structure, **in all situations**, when the bill is fully paid; **Mark F** for two conditions. One which checks / handles if the amount left to pay is `0` (or less, ie bill is paid), **AND** one which checks if the amount left to pay is less than `0` (for tip); **Mark G** for outputting in an appropriate place `Tip is` and the tip as a number; **R.** if tip is outputted when the amount left to pay is not less than zero **Mark H** for outputting `Bill paid` **and** the amount left to pay in logically appropriate places; **Maximum 7 marks** if any errors in code. **I.** Case **I.** Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect. **I.** Messages or no messages with input statements	8

C# Example 1 (fully correct)

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

```
bool billPaid = false;                                          (Part of E)
decimal total = Convert.ToDecimal                               (Part of D)
(Console.ReadLine());
while (billPaid == false)                                       (Part of E)
{
   decimal partPayment = Convert.ToDecimal                      (Part of D)
(Console.ReadLine());
   total = total - partPayment;                                 (Part of D)
   Console.WriteLine(total);                                    (Part of H)
   if (total == 0)                                              (Part of F)
   {
      Console.WriteLine("Bill paid");                           (Part of H)
      billPaid = true;                                          (Part of E)
   } else if (total < 0)                                        (Part of F, G)
   {
      Console.WriteLine("Tip is " + -total);                    (Part of G)
      billPaid = true;                                          (Part of E)
   }
}
```

I. Indentation in C#

A. `Write` in place of `WriteLine`

Python Example 1 (fully correct)

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

```
total = float(input())                                          (Part of D)
billPaid = False                                                (Part of E)
while billPaid == False:                                        (Part of E)
   partPayment = float(input())                                 (Part of D)
   total = round(total – partPayment, 2)                        (Part of D)
   print(total)                                                 (Part of H)
   if total == 0:                                               (Part of F)
      print("Bill paid")                                        (Part of H)
      billPaid = True                                           (Part of E)
   elif total < 0:                                              (Part of F, G)
      print(f"Tip is: {-total}")                                (Part of G)
      billPaid = True                                           (Part of E)
```

A. without rounding / `round()` statements

VB.NET Example 1 (fully correct)

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

```
Dim billPaid As Boolean = False                                  (Part of E)
Dim total As Decimal = Console.ReadLine()                        (Part of D)
While billPaid = False

   Dim partPayment As Decimal = Console.ReadLine()               (Part of D)
   total = total - partPayment
   Console.WriteLine(total)                                      (Part of D)
   If total = 0 Then                                             (Part of H)
      Console.WriteLine("Bill paid")                             (Part of F)
      billPaid = True                                            (Part of H)
   ElseIf total < 0                                              (Part of E)
      Console.WriteLine("Tip is " & -total)                      (Part of F, G)
      billPaid = True                                            (Part of G)
   End If                                                        (Part of E)
End While
```

I. Indentation in VB.NET

A. `Write` in place of `WriteLine`

Question	Part	Marking guidance	Total marks
08		**4 marks for AO3 (design), 7 marks for AO3 (program)** Any solution that does not map to the mark scheme refer to lead examiner **Note to Examiners:** For marks **E** and **J** be careful not to penalise the same error twice. For example, if they have used 6 instead of 7 in mark E and then 21 instead of 22 in mark J apply a **DPT** **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 attempting to randomly generate **two** numbers; **Mark B** for use of selection to check the current score against 21; **Mark C** for using iteration to keep rolling the dice; **Mark D** for outputting the dice rolls in appropriate places; **Program Logic** **Mark E** for generating **two** random numbers between 1 and 6 inclusive; **Mark F** for correctly adding the **two** dice values cumulatively to the previous score; **Mark G** for a loop that terminates if the current score is less than 21 **and** player chooses not to roll again; **Mark H** for a correct mechanism to end the game if the player has a score greater than or equal to 21; **Mark I** for a selection statement which correctly checks if the player has lost (final score is greater than 21) **OR** won (final score is 21); **Mark J** for generating a random number between 15 and 21 inclusive in a logically correct place **AND** checking if the result is greater than the final score; **Mark K** for **at least one** correct set of messages output in appropriate places to show whether the user has won or lost; **A.** yes/y, no/n or any other appropriate equivalents **Maximum 10 marks** if any errors in code. **I.** Case **I.** Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect. **I.** Messages or no messages with input statements	11

C# Example 1 (fully correct)

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

```
Random r = new Random();
int score = 0;
string rollAgain = "yes";

while (rollAgain == "yes")                                  (C, Part of G, Part of H)
{
   int dice1 = r.Next(1, 7);                                (Part of A,E)
   int dice2 = r.Next(1, 7);                                (Part of A,E)
   score = score + dice1 + dice2;                           (F)
   Console.WriteLine("Roll 1: " + dice1);                   (Part of D)
   Console.WriteLine("Roll 2: " + dice2);                   (Part of D)
   Console.WriteLine("Current score: " + score);            (Part of D)
   if (score < 21)                                          (Part of G)
   {
      rollAgain = Console.ReadLine();                       (Part of G)
   } else
   {
      rollAgain = "no";                                     (Part of H)
   }
}
if (score > 21)                                             (Part of I)
{
   Console.WriteLine("You lost! ");                         (Part of K)
} else if (score == 21)                                     (Part of I)
{
   Console.WriteLine("You won! ");                          (Part of K)
} else                                                      (Part of I)
{
   if (r.Next(15, 22) > score)                              (J)
   {
      Console.WriteLine("You lost! ");                      (Part of K)
   } else
   {
      Console.WriteLine("You won! ");                       (Part of K)
   }
}
```

I. Indentation in C#

A. `Write` in place of `WriteLine`

Python Example 1 (fully correct)

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

```
import random
score = 0
rollAgain = "yes"

while rollAgain == "yes":                                   (C, Part of G, Part of H)
   dice1 = random.randrange(1, 7)                           (Part of A,E)
   dice2 = random.randrange(1, 7)                           (Part of A,E)
   score = score + dice1 + dice2                            (F)
   print(f"Roll 1: {dice1}")                                (D)
   print(f"Roll 2: {dice2}")
   print(f"Current score: {score}")
   if score < 21:                                           (Part of G)
      rollAgain = input()                                   (Part of G)
   else:
      rollAgain = "no"                                      (Part of H)
if score > 21:                                              (Part of I)
   print("You lost! ")                                      (Part of K)
elif score == 21:                                           (Part of I)
   print("You won! ")                                       (Part of K)
else:                                                       (Part of I)
   if random.randrange(15,22) > score:                      (J)
      print("You lost!")                                    (Part of K)
   else:
      print("You won! ")                                    (Part of K)
```

A. `random.randint(1, 6)`

A. `random.randint(15, 21)`

VB.NET Example 1 (fully correct)

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

```
Dim r As Random = New Random()
Dim score As Integer
Dim rollAgain As String = "yes"
Dim dice1, dice2 As Integer

While rollAgain = "yes"                                          (C, Part of G, Part of H)

   dice1 = r.Next(1, 7)                                          (Part of A,E)
   dice2 = r.Next(1, 7)                                          (Part of A,E)
   score = score + dice1 + dice2                                 (F)
   Console.WriteLine("Roll 1: " & dice1)                         (Part of D)
   Console.WriteLine("Roll 2: " & dice2)                         (Part of D)
   Console.WriteLine("Current score: " & score)                  (Part of D)
   If score < 21 Then                                            (Part of G)
      rollAgain = Console.ReadLine()                             (Part of G)
   Else
      rollAgain = "no"                                           (Part of H)
   End If
End While

If score > 21 Then                                               (Part of I)
   Console.WriteLine("You lost! ")                               (Part of K)
ElseIf score = 21 Then                                           (Part of I)
   Console.WriteLine("You won! ")                                (Part of K)
Else                                                             (Part of I)
   If r.Next(15, 22) > score Then                                (J)
      Console.WriteLine("You lost! ")                            (Part of K)
   Else
      Console.WriteLine("You won! ")                             (Part of K)
   End If
End If
```

I. Indentation in VB.NET

A. `Write` in place of `WriteLine`

Question	Part	Marking guidance	Total marks

09		**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
10		**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
11		**1 mark for AO3 (refine)** B; **R.** if more than 1 lozenge shaded	1

Question	Part	Marking guidance	Total marks
12		**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**)	

Question	Part	Marking guidance	Total marks
13		**2 marks for AO3 (design), 5 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; **Mark B** for the use of a selection structure to check the total mark is less than zero or equivalent; **Program Logic** **Mark C** for using user input and storing the result in a numeric variable for the number of late essays; **Mark D** for correctly summing the total marks using the contents of variables `e1`, `e2` and `e3` in all circumstances **and** either reducing the total by `10` **or** halving the total mark **Mark E** for **two** expressions / a **combined** expression that checks the number of late essays correctly; **Mark F** for a correct expression(s) that prevents the total mark being less than 0 (eg by resetting the total mark to 0 or preventing it going below 0); **Mark G** for outputting total mark in the correct place; **R.** if any required calculations are performed on total mark after the last time the variable is output. **Maximum 6 marks** if any errors in code. **I.** Case **I.** Messages or no messages with input statements **I.** Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect **Note to examiners** In C#/VB.NET examples, explicit variable declarations are not shown. Refer to the specific language type issues section of the appropriate Marking guidance document. Any correct variable declarations in student answers should be accepted.	7

C# Example 1 (fully correct)

```
lateCount = Convert.ToInt32(Console.ReadLine());   (C)
total = e1 + e2 + e3;                              (Part D)
if (lateCount == 1)                                (Part E)
{
   total = total - 10;                             (Part D)
}
if (lateCount > 1)                                 (Part E)
{
   total = total / 2;                              (Part D)
}
if (total < 0)                                     (Part F)
{
   total = 0;                                      (Part F)
}
Console.WriteLine(total);                          (G)
```

I. Indentation
A. `Write` in place of `WriteLine`

Python Example 1 (fully correct)

```
lateCount = int(input())                           (C)
total = e1 + e2 + e3                               (Part D)
if lateCount == 1:                                 (Part E)
   total = total - 10                              (Part D)
if lateCount > 1:                                  (Part E)
   total = total / 2                               (Part D)
if total < 0:                                      (Part F)
   total = 0                                       (Part F)
print(total)                                       (G)
```

Python Example 2 (fully correct)

```
lateCount = int(input())                           (C)
total = e1 + e2 + e3                               (Part D)
if lateCount == 1 and total >= 10:                 (Part E, Part F)
   total = total - 10                              (Part D)
elif lateCount == 1 and total < 10:                (Part E, Part F)
   total = 0                                       (Part F)
elif lateCount > 1:                                (Part E)
   total = total * 0.5                             (Part D)
print(total)                                       (G)
```

VB.NET Example 1 (fully correct)

```
lateCount = Console.ReadLine()                    (C)
total = e1 + e2 + e3                              (Part D)
If lateCount = 1 Then                             (Part E)
   total = total - 10                             (Part D)
End If
If lateCount > 1 Then                             (Part E)
   total = total / 2                              (Part D)
End If
If total < 0 Then                                 (Part F)
   total = 0                                      (Part F)
End If
Console.WriteLine(total)                          (G)
```

I. Indentation
A. `Write` in place of `WriteLine`

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>14</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>

Question	Part	Marking guidance	Total marks
15	1	**Mark is for AO2 (apply)** 11;	1

Question	Part	Marking guidance	Total marks
15	2	**Mark is for AO2 (apply)** 17;	1

Question	Part	Marking guidance	Total marks
16	2	**2 marks for AO3 (design), 4 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 the use of a definite iteration structure, or similar, that exists within their language to carry out the requirements of the task; **Mark B** for the use of a selection structure to check visitor numbers; **Program Logic** **Mark C** for correctly defining the subroutine and parameter; **Mark D** for accepting user input multiple times as per the parameter or equivalent, representing the number of days; **R.** if iteration syntax or boundaries are not fully correct **Mark E** for adding one to a counter variable inside a selection structure under the correct conditions, which has been appropriately initialised (to `0`); **Mark F** for returning the counter value calculated within the subroutine; **Maximum 5 marks** if any errors in code. **I.** Case **I.** Messages or no messages with input statements **I.** Gaps/spaces throughout the code, except where to do so would explicitly alter the logic of the code in a way that makes it incorrect **Note to examiners** In C#/VB.NET examples, explicit variable declarations are not shown. Refer to the specific language type issues section of the appropriate Marking guidance document. Any correct variable declarations in student answers should be accepted.	6

C# Example 1 (fully correct)

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

```
static int countDays(days) {                                    (C)
   count = 0;                                                   (Part E)
   visitors = 0;
   for (i = 0; i < days; i++) {                                 (Part D)
      visitors =
Convert.ToInt32(Console.ReadLine());                            (Part D)
      if (visitors > 200) {                                     (Part E)
         count = count + 1;                                     (Part E)
      }
   }
   return count;                                                (F)
}
```

A. with or without `static`

A. Alternative numerical data type for return value

I. Indentation in C#

Python Example 1 (fully correct)

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

```
def countDays(days):                                            (C)
   count = 0                                                    (Part E)
   for i in range(days):                                        (Part D)
      visitors = int(input())                                   (Part D)
      if visitors > 200:                                        (Part E)
         count = count + 1                                      (Part E)
   return count                                                 (F)
```

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

```
def countDays(days):                          (C)
    count = 0                                 (Part E)
    i = 0                                     (Part D)
    while (i < days):                         (Part D)
        if int(input()) > 200:                (Part D) (Part E)
            count += 1                        (Part E)
        i += 1                                (Part D)
    return count                              (F)
```

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

```
Function countDays(days) As Integer           (C)
    count = 0                                 (Part E)
    For i = 1 To days                         (Part D)
        visitors = Console.ReadLine()         (Part D)
        If visitors > 200 Then                (Part E)
            count = count + 1                 (Part E)
        End If
    Next
    Return count                              (F)
End Function
```

A. Alternative numerical data type for return value
I. Indentation in VB.NET

VB.NET Example 2 (fully correct)

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

```
Function countDays(days) As Integer          (C)
   Dim count As Integer                      (Part E)
   For i = 1 To days                         (Part D)
      If Console.ReadLine() > 200 Then       (Part E)
         count += 1                          (Part E)
      End If
   Next
   Return count                              (F)
End Function
```

A. Alternative numerical data type for return value
I. Indentation in VB.NET