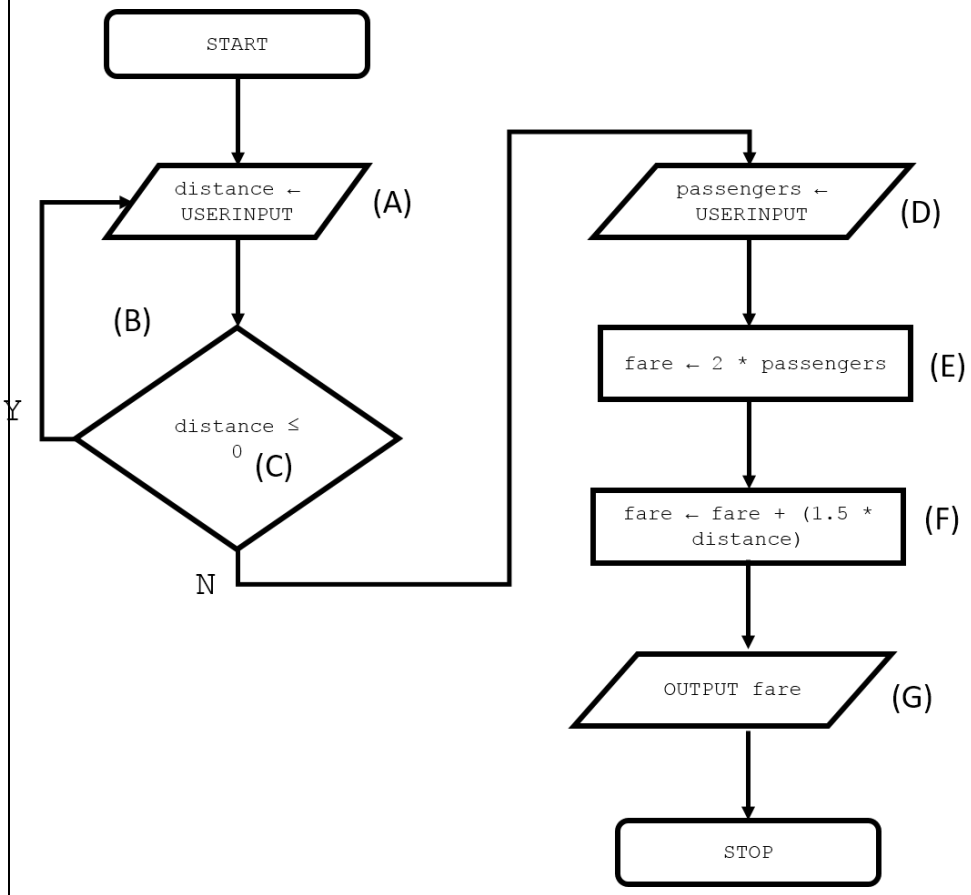
Question	Part	Marking guidance	Total marks
01		**8 marks for AO3 (program)** **DPT.** For repeated errors in user input and variable assignment. Mark A for getting user input for the distance and storing in a variable; Mark B for using a WHILE loop or similar to re-prompt for and re-assign the user input; 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)** **D** `USERINPUT`; **If more than one lozenge shaded then mark is not awarded**	1
02	2	**Mark is for AO2 (apply)** **B** `0`; **If more than one lozenge shaded then mark is not awarded**	1
02	3	**Mark is for AO2 (apply)** **A** `=` ; **If more than one lozenge shaded then mark is not awarded**	1
02	4	**Mark is for AO2 (apply)** **D** `OUTPUT count`; **If more than one lozenge shaded then mark is not awarded**	1
02	5	**Mark is for AO2 (apply)** **B** `i ← i + 1`; **If more than one lozenge shaded then mark is not awarded**	1
02	6	**2 marks for AO2 (apply)** **Maximum of 1 mark if Upper Case Characters given** • 1 mark for a series of more than one correct frequency/value or value/frequency pairs (ignore order of pairs); • 1 mark for all correct pairs in the correct order; Correct answer is: 2 t 2 j 3 e 2 s Other, clear ways to show frequency/value or value/frequency pairs such as '(2, t), (2, j),…' or 't2 j2…'.	2

Question	Part	Marking guidance	Total marks
02	7	**3 marks for AO2 (apply)** Maximum three marks from: • It could be tested with only 1s; • It could be tested with different lengths of input; • It could be tested with an input where the 1s and 0s vary; • It could be tested with an input where the last two numbers are different; • It could be tested with the empty string; • It could be tested with a string of length one; • It could be tested with two runs of 0s separated by a run of 1s / two runs of 1s separated by a run of 0s; • It could be tested with invalid data (such as 1010abc); Any other correct reasoning as long as clearly distinct from other mark points. **R.** not enough tests are carried out.	3
03	1	**Mark is for AO2 (apply)** **C** Selection; **If more than one lozenge shaded then mark is not awarded**	1
03	2	**Mark is for AO2 (apply)** **D** String; **If more than one lozenge shaded then mark is not awarded**	1
03	3	**Mark is for AO2 (apply)** 3//three;	1
03	4	**2 marks for AO2 (apply)** `'no'` followed by `'yes'`; any value that isn't 'no' followed by 'yes' (allow by examples such as 'yes' followed by 'yes'); **R.** if a sequence does not contain two user inputs.	2

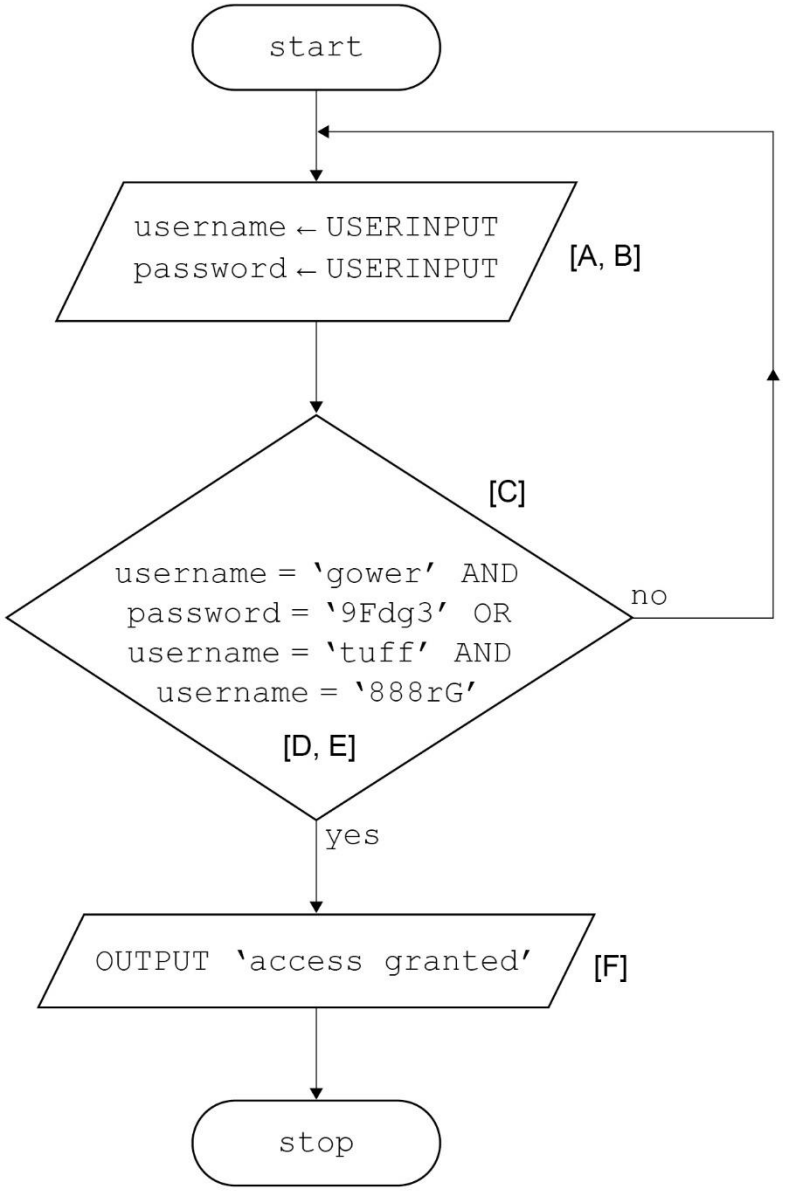
Question	Part	Marking guidance	Total marks
03	5	**3 marks for AO2 (apply)** Maximum three marks overall. **Maximum two marks from each section.** **<u>Reason</u>** • The output message is not descriptive enough/the user is not told what word/words they should use to answer (before user input); • The Boolean expression (at lines 3, 6 and 14) only matches exact values//the program is only written for the exact words yes and no // a **clear** indication that y is not recognised as yes or n is not recognised as no; • A clear explanation of how to fix the problem; **<u>What would happen</u>** Any clear descriptions of what would happen. Line numbers may or may not be included. If the logic and explanation is clear credit the answer. This can include but is not limited to: • Line 3 will only be true if they enter 'no' // Line 3 will not be true if they enter anything other than 'no'; • Line 6/14 will only be true if they enter 'yes' // Line 6/14 will not be true if they enter anything other than 'yes'; • if they enter 'n' at line 2 the algorithm will execute an incorrect code block; • if they enter 'y' at line 5 or line 13 an incorrect message will be output;	3

Qu	Part	Marking guidance	Total marks
04		**6 marks for AO3 (program)** **Mark A** for assigning user input to a variable (username); **Mark B** for assigning user input to a variable (password, the identifier must be different to that used in mark A); **Mark C** for using indefinite iteration and including user input within the iteration structure; **Mark D** for using a Boolean condition that checks the username is `gower` and the password is `9Fdg3` / the username is `tuff` and the password is `888rG`; **Mark E** for using the Boolean `OR` operator for both combinations of username and password, alternatively having sequential `IF` or `ELSE-IF` structures; **Mark F** for outputting the string after the iteration structure; **Max 5 marks** if the algorithm contains any errors. **I.** use of quote marks for usernames or passwords. **I.** minor spelling errors for username or passwords. Example of fully correct answer: `REPEAT` *[part C]* `  username ← USERINPUT` *[A, part C]* `  password ← USERINPUT` *[B, part C]* `UNTIL (username = 'gower' AND password = '9Fdg3') OR (username = 'tuff' AND password = '888rG')` *[D, E]* `OUTPUT 'access granted'` *[F]* Another example of a fully correct answer: `username ← USERINPUT` *[A]* `password ← USERINPUT` *[B]* `WHILE NOT ((username = 'gower' AND password = '9Fdg3') OR (username = 'tuff' AND password = '888rG'))` *[D, E, part C]* `  username ← USERINPUT` *[part C]* `  password ← USERINPUT` *[part C]* `ENDWHILE` `OUTPUT 'access granted'` *[F]*	6

Another example of a fully correct answer:

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

An example of a fully correct flowchart solution:



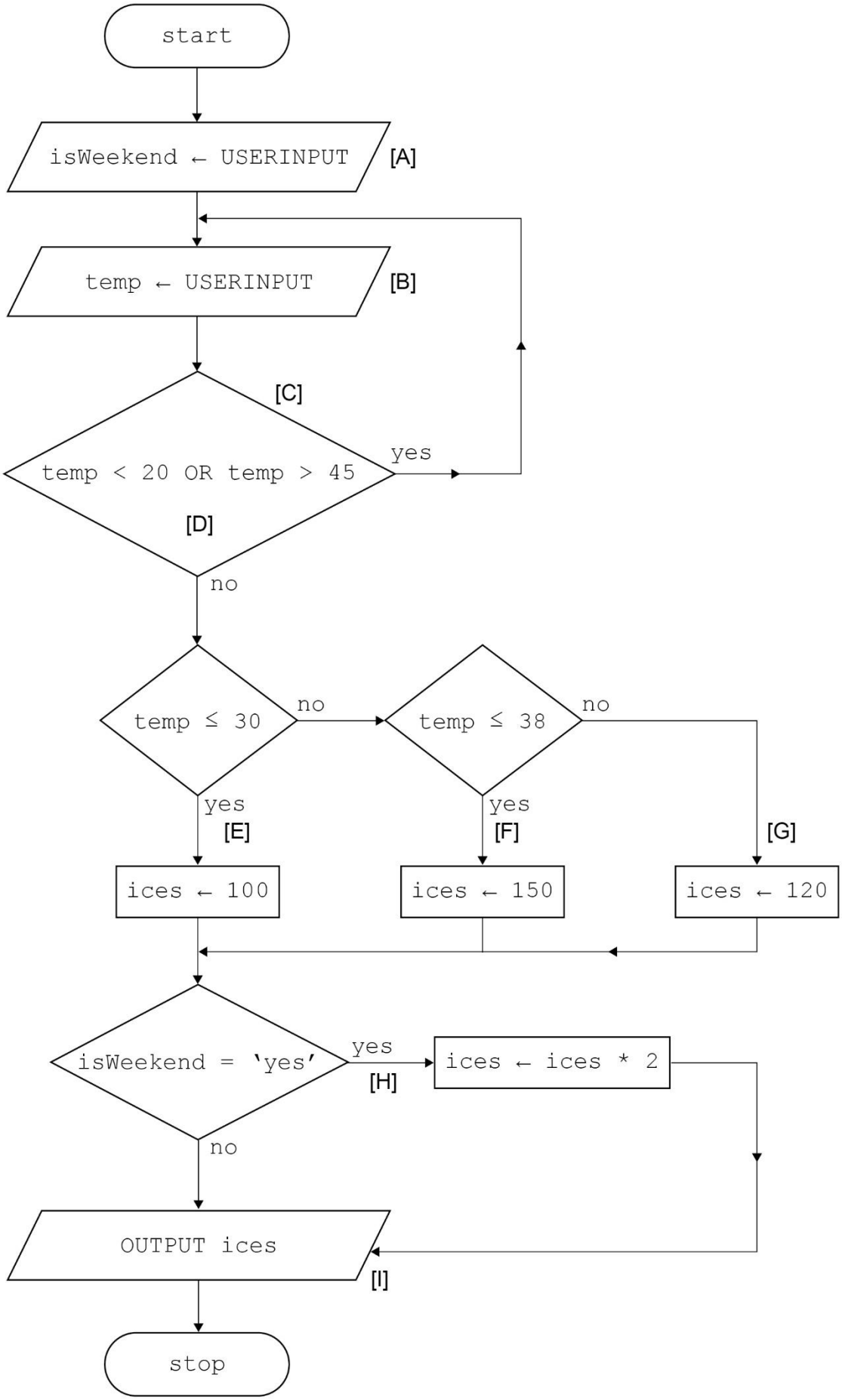
Qu	Part	Marking guidance	Total marks

05		**9 marks for AO3 (program)**
		
		Mark A for assigning user input to a variable (weekend or weekday);
		Mark B for assigning user input to a variable (temperature);
		Mark C for using indefinite iteration to repeatedly input the temperature;
		Mark D for a Boolean condition used to check the temperature between 20 and 45 inclusive;
		Mark E for using selection to set ice creams to be 100 if the temp is between 20 and 30 inclusive;
		Mark F for using selection to set ice creams to be 150 if the temp is between 31 and 38 inclusive;
		Mark G for using selection to set ice creams to be 120 if the temp is higher than 38;
		Mark H for doubling the quantity if it is a weekend (mark A is not required);
		Mark I for **always** outputting the estimated number of ice creams;
		
		Max 8 marks if solution contains any errors.
		
		An example of a fully correct solution:
		
		`isWeekend ← USERINPUT` *[A]*
		`temp ← USERINPUT` *[B]*
		`WHILE temp < 20 OR temp > 45` *[part C, D]*
		`  temp ← USERINPUT` *[part C]*
		`ENDWHILE`
		`IF temp ≤ 30 THEN` *[part E]*
		`  ices ← 100` *[part E]*
		`ELSE IF temp ≤ 38 THEN` *[part F]*
		`  ices ← 150` *[part F]*
		`ELSE` *[part G]*
		`  ices ← 120` *[part G]*
		`ENDIF`
		`IF isWeekend = 'yes' THEN` *[part H]*
		`  ices ← ices * 2` *[part H]*
		`ENDIF`
		`OUTPUT ices` *[part I]*
			9

Another example of a fully correct solution:

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

An example of a fully correct flowchart solution:


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

Question	Part	Marking guidance	Total marks

06		**2 marks for AO3 (design), 3 marks for AO3 (program)**
		
		Program Design
		Note that AO3 (design) marks are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.
		
		Mark A for using meaningful variable names throughout and for using two variables to store the two email address inputs;
		Mark B for the use of a selection construct // use of multiple selection constructs;
		
		Program Logic
		Mark C for using user input and storing the results in two variables correctly for the first email address and the second email address;
		Mark D for a correct expression that checks if the first entered email address is equal to the second entered email address (or not equal to);
		Mark E for outputting `Do not match` and `Match` in logically separate places such as the IF and ELSE part of selection, and for outputting the email address if both email addresses match;
		
		A. Any suitable alternative messages.
		
		I. Case
		I. Messages or no messages with input statements
		
		Maximum 4 marks if any errors in code.
		
		C# Example 1 (fully correct)
		All design marks are achieved (**Marks A and B**)
		
		`string email1 = Console.ReadLine();` (**Part of C**)
		`string email2 = Console.ReadLine();` (**Part of C**)
		
		`if (email1 != email2) {` (**D**)
		`  Console.WriteLine("Do not match");` (**Part of E**)
		`}`
		`else {`
		`  Console.WriteLine("Match");` (**Part of E**)
		`  Console.WriteLine(email1);` (**Part of E**)
		`}`
			5

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Question	Part	Marking guidance	Total marks

07		**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
```

Question	Part	Marking guidance	Total marks

08	1	**2 marks for AO3 (design), 2 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 idea of inputting a number within the iteration/validation structure;
		Mark B for the use of indefinite iteration;
		
		Program Logic
		Mark C for using a Boolean condition that checks the lower or upper bound of `position`;
		Mark D for using a Boolean condition that checks **BOTH** the lower and upper bounds of `position` correctly;
		Marks C and **D** could be one expression eg `0 < position <= 100;`
		
		I. Case
		I. Missing prompts
		
		Maximum 3 marks if any errors in code.
		
		C# Example 1 (fully correct)
		All design marks are achieved (**Marks A and B**)
		`while (position < 1 || position > 100) {` (**C,D**)
		`  Console.Write("Enter card position: ");`
		`  position = Convert.ToInt32(Console.ReadLine());`
		`}`
		
		C# Example 2 (fully correct)
		All design marks are achieved (**Marks A and B**)
		`while (position <= 0 || position >= 101) {` (**C,D**)
		`  Console.Write("Enter card position: ");`
		`  position = Convert.ToInt32(Console.ReadLine());`
		`}`
		
		C# Example 3 (partially correct – 3 marks)
		1 design mark achieved (**Mark A**)
		`if (position < 1 || position > 100) {` (**C,D**)
		`  Console.Write("Enter card position: ");`
		`  position = Convert.ToInt32(Console.ReadLine());`
		`}`
			4

C# Example 4 (partially correct – 3 marks)

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

```
while (position < 1 || position >= 100) {          (Mark C)
   Console.Write("Enter card position: ");
   position = Convert.ToInt32(Console.ReadLine());
}
```

I. Indentation in C#

I. `WriteLine` instead of `Write`

Python Example 1 (fully correct)

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

```
while position < 1 or position > 100:                (C,D)
   position = int(input("Enter card position: "))
```

Python Example 2 (fully correct)

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

```
while position <= 0 or position >= 101:              (C,D)
   position = int(input("Enter card position: "))
```

Python Example 3 (partially correct – 3 marks)

1 design mark achieved (**Mark A**)

```
if position < 1 or position > 100:                   (C,D)
   position = int(input("Enter card position: "))
```

Python Example 4 (partially correct – 3 marks)

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

```
while position < 1 or position >= 100:      (C)
   position = int(input("Enter card position: "))
```

VB.NET Example 1 (fully correct)

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

```
While position < 1 Or position > 100                 (C,D)
   Console.Write("Enter card position: ")
   position = Console.ReadLine()
End While
```

VB.NET Example 2 (fully correct)

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

```
While position <= 0 Or position >= 101               (C,D)
   Console.Write("Enter card position: ")
   position = Console.ReadLine()
End While
```

VB.NET Example 3 (partially correct – 3 marks)

1 design mark achieved (**Mark A**)

```
If position < 1 Or position > 100 Then               (C,D)
   Console.Write("Enter card position: ")
   position = Console.ReadLine()
End If
```

		VB.NET Example 4 (partially correct – 3 marks)
		All design marks are achieved (**Marks A and B**)
		`Do While position < 1 Or position >= 100` (**Mark C**)
		`  Console.Write("Enter card position: ")`
		`  position = Convert.ToInt32(Console.ReadLine())`
		`Loop`
		
		I. Indentation in VB.NET
		I. `WriteLine` instead of `Write`	

Question	Part	Marking guidance	Total marks

08	2	**2 marks for AO3 (design), 4 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 the idea of using an iteration structure which attempts to access each element in the `cards` array; // attempts to repeat 100 times;
		Mark B for the idea of using a selection structure which attempts to compare two cards;
		
		Program Logic
		Mark C for using a loop or similar to correctly iterate through the `cards` array using valid indices that do not go out of range;
		Mark D for using correct Boolean conditions that compare values in the `cards` array;
		Mark E for correctly checking if there are five values in the `cards` array that are in sequence;
		Mark F for setting `gameWon` to `True` in the correct place;
		
		I. Case
		
		Maximum 5 marks if any errors in code.
		
		C# Example 1 (fully correct)
		All design marks are achieved (**Marks A and B**)
		`int count = 1;` **(Part of E)**
		`for (int i = 0; i < 99; i++) {` **(C)**
		`if (cards[i] + 1 == cards[i+1]) {` **(D, Part of E)**
		`count = count + 1;` **(Part of E)**
		`if (count == 5) {` **(Part F)**
		`gameWon = true;` **(Part F)**
		`}`
		`}`
		`else {`
		`count = 1;` **(Part of E)**
		`}`
		`}`
			6

C# Example 2 (fully correct)

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

```
int count = 1;                                  (Part of E)
int i = 0;                                      (Part of C)
while (i < 99) {                                (Part of C)
     if (cards[i] + 1 == cards[i+1]) {          (D, Part of E)
          count = count + 1;                    (Part of E)
          if (count == 5) {                     (Part F)
               gameWon = true;                  (Part F)
          }
     }
     else {
          count = 1;                            (Part of E)
     }
     i = i + 1;                                 (Part of C)
}
```

I. Indentation in C#

Python Example 1 (fully correct)

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

```
count = 1                                       (Part of E)
for i in range(99):                             (C)
  if cards[i] + 1 == cards[i + 1]:              (D, Part of E)
    count = count + 1                           (Part of E)
    if count == 5:                              (Part F)
      gameWon = True                            (Part F)
  else:
    count = 1                                   (Part of E)
```

Python Example 2 (fully correct)

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

```
count = 0                                       (Part of E)
i = 0                                           (Part of C)
while i < len(cards) - 1:                       (Part of C)
  if cards[i] + 1 == cards[i + 1]:              (D, Part of E)
    count = count + 1                           (Part of E)
    if count == 4:                              (Part F)
      gameWon = True                            (Part F)
  else:
    count = 0                                   (Part of E)
  i = i + 1                                     (Part of C)
```

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

```
gameWon = False                                          (Part F)
for i in range(96):                                      (C)
  count = 1                                              (Part of E)
  for j in range(1, 5):                                  (Part of D)
    if cards[i + j] - 1 == cards[i + j - 1]:             (Part of D)
                                                         (Part of E)
      count += 1                                         (Part of E)
  if count == 5:                                         (Part F)
    gameWon = True                                       (Part F)
```

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

```
Dim count As Integer = 1                                 (Part of E)
For i = 0 To 98                                          (C)
     If cards(i) + 1 = cards(i+1) Then                   (D, Part of E)
          count = count + 1                              (Part of E)
          If count = 5 Then                              (Part F)
               gameWon = True                            (Part F)
          End If
     Else
          count = 1                                      (Part of E)
     End If
Next
```

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

```
Dim count As Integer = 0                                 (Part of E)
Dim i As Integer = 0                                     (Part of C)
While i < 99                                             (Part of C)
     If cards(i) + 1 = cards(i+1) Then                   (D, Part of E)
          count = count + 1                              (Part of E)
          If count = 4 Then                              (Part F)
               gameWon = True                            (Part F)
          End If
     Else
          count = 0                                      (Part of E)
     End If
     i = i + 1                                           (Part of C)
End While
```

I. Indentation in VB.NET

Question	Part	Marking guidance	Total marks
09	1	**Mark is for AO2 (apply)** **A** Line number 2; **R.** if more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
09	2	**Mark is for AO2 (apply)** **A** `0`; **R.** if more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
09	3	**Mark is for AO2 (apply)** **C** 4; **R.** if more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
10		**4 marks for AO3 (design)** **1 mark** for each correct answer L1 `USERINPUT` L2 `username` L3 `''` **R.** `""` L4 `User not found` **I.** case / spelling mistakes so long as it is clear which option from **Figure 6** has been selected. **Note to Examiners:** If the student has re-written the entire line and added in the correct missing item, award the mark.	4

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>11</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
12		**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 using the variable `check` within their own code; **Mark B** for using selection or equivalent to check the grid references; **Program Logic** **Mark C** for correctly using an appropriate technique (slicing/indexing/`substring` function) with correct syntax to extract the left **and** right characters of input // for correctly comparing all nine possible valid grid references; **Mark D** for using **one** appropriate correct Boolean condition, eg =”A” // =”2” or equivalent; **Mark E** for having **all** the appropriate correct Boolean conditions to check the letters and numbers **AND** for `check` being set appropriately in all cases; **Mark F** for outputting an appropriate message in a logically appropriate location if their checks have failed; **Maximum 5 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.	6

C# Example 1 (fully correct)

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

```
bool check = false;
while (check == false) {
   string square = "";
   while (square.Length != 2) {
      Console.Write("Enter grid reference: ");
      square = Console.ReadLine();
      square = square.ToUpper();
   }
   char letter = square[0];
   char number = square[1];
   if ((letter == 'A' || letter == 'B' || letter
== 'C') && (number == '1' || number == '2' ||
number == '3'))
   {
      check = true;
   }
   else
   {
      Console.WriteLine("Not valid, try again.");
   }
}
```

(Part of C, Part of C) **(D)** **(E)**

(F)

I. Indentation in C#

I. Duplicate } at the end of the program (as if student has missed the bracket in the writing lines)

A. use of double quotes for **Mark E**

A. `Write` in place of `WriteLine`

Python Example 1 (fully correct)

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

```
check = False
while check == False:
   square = ""
   while len(square) != 2:
      square = input("Enter grid reference: ")
      square = square.upper()

   letter = square[0]
   number = square[1]

   if letter in "ABC" and number in "123":
      check = True
   else:
      print("Not valid, try again. ")
```

(Part of C, Part of C)

(D)(E)

(F)

A. use of single quotes for **Mark E**

VB.NET Example 1 (fully correct)

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

```
Dim check As Boolean = False
While check = False
   Dim square As String = ""
   While square.Length <> 2
      Console.Write("Enter grid reference: ")
      square = Console.ReadLine()
      square = square.ToUpper()
   End While
   Dim letter As String = square(0)
   Dim number As String = square(1)

   If (letter = "A" Or letter = "B" Or letter =
"C") And (number = "1" Or number = "2" Or number
= "3") Then
      check = True
   Else
      Console.WriteLine("Not valid, try again. ")
   End If
End While
```

(Part of C, Part of C)

(D)
(E)

(F)

I. Indentation in VB.NET

I. Duplicate `End While` at the end of the program (as if student has missed the bracket in the writing lines)

A. `Write` in place of `WriteLine`

A. use of single quotes for **Mark E**

VB.NET Example 2 (fully correct)

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

```
Dim check As Boolean = False
While check = False
   Dim square As String = ""
   While square.Length <> 2
      Console.Write("Enter grid reference: ")
      square = Console.ReadLine()
      square = square.ToUpper()
   End While
   Dim letter As String = square.substring(0,1)
   Dim number As String = square.substring(1,1)

   If (letter = "A" Or letter = "B" Or letter =
"C") And (number = "1" Or number = "2" Or number
= "3") Then
      check = True
   Else
      Console.WriteLine("Not valid, try again. ")
   End If
End While
```

(Part of C, Part of C)

(D)
(E)

(F)

I. Indentation in VB.NET

I. Duplicate `End While` at the end of the program (as if student has missed the bracket in the writing lines)

A. `Write` in place of `WriteLine`

A. use of single quotes for **Mark E**

Question	Part	Marking guidance	Total marks
13		**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
14		**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
15		**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

16		**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)
```

Question	Part	Marking guidance	Total marks
17		**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
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**)	

Question	Part	Marking guidance	Total marks
19		**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`

Question	Part	Marking guidance	Total marks
20		**1 mark for AO3 (design), 3 marks for AO3 (program)** **Program Design** **Note** that AO3 (design) marks are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works. **Mark A** for the idea of using concatenation to create the stock code; **Program Logic** **Mark B** for using user input correctly for the `sweetID`, `sweetName` and `brand`; **A.** similar distinct/meaningful variable names. **Mark C** for correctly creating each part of the stock code; **A.** if stock code is output instead of assigned to variable. **Mark D** for assigning the stock code / three string variables representing `sweetID`, `sweetName` and `brand` correctly to the variable `code` (even if the generated stock code is not correct); **R.** any other variable name for `code` **Maximum 3 marks** if any errors. **I.** `print` / `Console.WriteLine` statements **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 **R.** commas used to show concatenation	4

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.

C# Example 1 (fully correct)
Design mark is achieved (**Mark A**)

```
sweetID = Console.ReadLine();                    (Part B)
sweetName = Console.ReadLine();                  (Part B)
brand = Console.ReadLine();                      (Part B)
code = sweetID + sweetName[0] + sweetName[1]     (C, D)
+ brand[0];
```

A. `sweetID.Substring(0, 2)`
I. Indentation

C# Example 2 (fully correct)
Design mark is achieved (**Mark A**)

```
code = Console.ReadLine() +                      (B,C,D)
Console.ReadLine().Substring(0, 2) +
Console.ReadLine()[0];
```

I. Indentation

Python Example 1 (fully correct)
Design mark is achieved (**Mark A**)

```
sweetID = input()                                (Part B)
sweetName = input()                              (Part B)
brand = input()                                  (Part B)
code = sweetID + sweetName[0] + sweetName[1]     (C, D)
+ brand[0]
```

A. `sweetID[0:2]`

Python Example 2 (fully correct)
Design mark is achieved (**Mark A**)

```
code = input() + input()[0:2] + input()[0]       (B, C, D)
```

Python Example 3 (partially correct – 3 marks)
Design mark is achieved (**Mark A**)

```
code = input() + input() + input()               (B, D)
```

VB.NET Example 1 (fully correct)
Design mark is achieved (**Mark A**)

```
sweetID = Console.ReadLine()                    (Part B)
sweetName = Console.ReadLine()                  (Part B)
brand = Console.ReadLine()                      (Part B)
code = sweetID + sweetName(0) + sweetName(1)    (C, D)
+ brand(0)
```

A. `sweetID.Substring(0, 2)`
I. Indentation

VB.NET Example 2 (fully correct)
Design mark is achieved (**Mark A**)

```
code = Console.ReadLine() &                     (B, C, D)
Console.ReadLine().Substring(0, 2) &
Console.ReadLine()(0)
```

I. Indentation

Question	Part	Marking guidance	Total marks
21	1	**Mark is for AO1 (understanding)** **D** An organised collection of values; **R.** If more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
21	2	**3 marks for AO2 (apply)** **3 marks** if all **four** are correct: • **`Book`** on line 1 • **`author`** on line 3 • **`Real`** on line 4 • **`Book`** on line 7 **2 marks** if any **three** are correct **1 mark** if any **two** are correct 1 `RECORD Book` 2 `  bookName : String` 3 `  author : String` 4 `  price : Real` 5 `ENDRECORD` 6 `B1 ← Book("The Book Thief", "M Zusak", 9.99)` 7 `B2 ← Book("Divergent", "V Roth", 6.55)` **I.** Case	3

Question	Part	Marking guidance	Total marks

21	3	**3 marks for AO2 (apply)**
		
		`IF B1.price > B2.price THEN`
		`  OUTPUT B1.bookName`
		`ELSEIF B1.price < B2.price THEN`
		`  OUTPUT B2.bookName`
		`ELSE`
		`  OUTPUT "Neither"`
		`ENDIF`
		
		1 mark for correctly using a selection structure with multiple conditions // use of multiple selection structures to compare B1 and B2 in some way (even if Boolean conditions incorrect);
		
		1 mark for correct Boolean conditions throughout to compare the prices;
		
		1 mark for displaying the correct output in each case;
		
		Max 2 marks if any errors
		
		I. Case
		A. Pseudo-code statements written using different syntax as long as the logic is still correct.
			3

Question	Part	Marking guidance	Total marks
22		**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 an indefinite iteration structure that exists within their language, for validation of the inputs; **Program Logic** **Mark C** for using user input and storing the result in two variables correctly for the username and password; **Mark D** for using correct Boolean expressions to check if the username and password entered matches at least **one** of the valid pairs; **A.** if the **only** error is missing quotes around string values **Mark E** for using correct Boolean expressions to check if the username and password entered matches **both of** the valid pairs; **R.** if **any** quotes missing around string values **Mark F** for allowing the user to enter the username and password again in an appropriate place (even if the Boolean expression is not correct); **DPT.** If mark C not awarded due to incorrect syntax. **Mark G** for displaying `Access granted` or `Access denied` in the appropriate places; **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 **Maximum 6 marks** if any errors in code.	7

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.

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

```
username = Console.ReadLine();                    (Part C)
password = Console.ReadLine();                    (Part C)
while ((username != "Yusuf5" || password          (D, E)
!= "33kk") && (username != "Mary80" ||
password != "af5r"))
{
    Console.WriteLine("Access denied");           (Part G)
    username = Console.ReadLine();                (Part F)
    password = Console.ReadLine();                (Part F)
}
Console.WriteLine ("Access granted");             (Part G)
```

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

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

```
valid = false;
do {
   username = Console.ReadLine();          (Part C, Part F)
   password = Console.ReadLine();          (Part C, Part F)
   if (username == "Yusuf5" &&             (Part D, Part E)
password == "33kk") {
      valid = true;                        (Part D)
   }
   else if (username == "Mary80" &&        (Part D, Part E)
password == "af5r") {
       valid = true;                       (Part D)
   }
   if (!valid) {                           (Part G)
      Console.WriteLine("Access            (Part G)
denied");
   }
} while (!valid);
Console.WriteLine ("Access granted");      (Part G)
```

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

C# Example 3 (fully correct)
All design marks are achieved (**Marks A and B**)

```
do {
   username = Console.ReadLine();          (Part C, Part F)
   password = Console.ReadLine();          (Part C, Part F)
   access = (username == "Yusuf5" &&       (D, E)
password == "33kk") || (username ==
"Mary80" && password == "af5r");
   if (access == false) {
      Console.WriteLine("Access            (Part G)
   }
} while (!access);
Console.WriteLine ("Access                 (Part G)
```

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

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

```
username = input()                                  (Part C)
password = input()                                  (Part C)
while (username != "Yusuf5" or password !=          (D, E)
"33kk") and (username != "Mary80" or
password != "af5r"):
    print("Access denied")                          (Part G)
    username = input()                              (Part F)
    password = input()                              (Part F)
print("Access granted")                             (Part G)
```

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

```
access = False                                      (Part F)
while access == False:                              (Part F)
   username = input()                               (Part C)
   password = input()                               (Part C)
   if (username == "Yusuf5" and password            (D, E)
== "33kk") or (username == "Mary80" and
password == "af5r"):
      print("Access granted")                       (Part G)
      access = True
   else:
      print("Access denied")                        (Part G)
```

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

```
username = Console.ReadLine()                          (Part C)
password = Console.ReadLine()                          (Part C)
While (username <> "Yusuf5" Or password <>             (D, E)
"33kk") And (username <> "Mary80" Or
password <> "af5r")
    Console.WriteLine("Access denied")                 (Part G)
    username = Console.ReadLine()                      (Part F)
    password = Console.ReadLine()                      (Part F)
End While
Console.WriteLine ("Access granted")                   (Part G)
```

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

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

```
valid = False
Do
   username = Console.ReadLine()                       (Part C, Part F)
   password = Console.ReadLine()                       (Part C, Part F)
   If username = "Yusuf5" And password =               (Part D, Part E)
"33kk" Then
      valid = True                                     (Part D)
   ElseIf username = "Mary80" And password             (Part D, Part E)
= "af5r" Then
       valid = True                                    (Part D)
   End If
   If Not valid Then                                   (Part G)
      Console.WriteLine("Access denied")               (Part G)
   End If
Loop Until valid
Console.WriteLine ("Access granted")                   (Part G)
```

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

Question	Part	Marking guidance	Total marks
23		**2 marks for AO3 (design), 6 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 selection structure which outputs `Bad move`; **Mark B** for the use of a nested selection structure // a selection structure with multiple conditions // use of multiple selection structures **Program Logic** **Mark C** for correctly inputting a move in an appropriate place within the `while` loop; **Mark D** for correctly checking the input for a move is either `1` or `2`; **I.** data validation attempts **Mark E** for adding the input value for a move to `pos` once per move; **Mark F** for resetting `pos` to `0` if the move takes a player beyond the end of the row; **A.** if the index used could go out of range. **Mark G** for a condition equivalent to `row() == "X"` that checks for the character `X` in `row` and resets `pos` to `0` if appropriate; **I.** missing or incorrect index number on `row`. **A.** if the index used could go out of range. **Mark H** for the correct use of indices to access the elements in the array `row` and the index does not go out of range; **Maximum 7 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.	8

C# Example 1 (fully correct)

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

```
    move =
Convert.ToInt32(Console.ReadLine());          (C)
    if (move == 1 || move == 2) {             (D)
        pos += move;                          (E)
    }
    if (pos > lastPos) {                      (Part F)
        pos = 0;                              (Part F)
        Console.WriteLine("Bad move");
    }
    else if (row[pos] == "X") {               (Part G, H)
        pos = 0;                              (Part G)
        Console.WriteLine("Bad move");
    }
```

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

C# Example 2 (7 marks)

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

No **Mark D** as program also adds numbers other than `1` or `2` to `pos`.

```
    move =
Convert.ToInt32(Console.ReadLine());          (C)
    if (pos + move > lastPos || row[pos +
move] == "X") {                               (Part F, Part G, H)
        Console.WriteLine("Bad move");
        pos = 0;                              (Part F, Part G)
    }
    else {
        pos = pos + move;                     (E)
    }
}
```

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

C# Example 3 (fully correct)

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

```
    move =
Convert.ToInt32(Console.ReadLine());                    (C)
    if (move == 1) {                                    (Part D)
        if (row[pos + 1] == "X") {                      (Part G)
            pos = 0;                                    (Part G)
            Console.WriteLine("Bad move");
        }
        else {
            pos = pos + 1;                              (Part E)
        }
    }
    if (move == 2) {                                    (Part D)
        if (pos + move > lastPos || row[pos +           (Part F, Part G, H)
2] == "X") {
            pos = 0;                                    (Part F)
            Console.WriteLine("Bad move");
        }
        else {
            pos = pos + 2;                              (Part E)
        }
    }
```

I. Indentation

A. `Write` in place of `WriteLine`

Python Example 1 (fully correct)

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

```
move = int(input())                          (C)
if move == 1 or move == 2:                   (D)
   pos += move                               (E)
if pos > lastPos:                            (Part F)
   pos = 0                                   (Part F)
   print("Bad move")
elif row[pos] == "X":                        (Part G, H)
   pos = 0                                   (Part G)
   print("Bad move")
```

Python Example 2 (fully correct)

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

```
move = int(input())                          (C)
if move == 1:                                (Part D)
   if row[pos + 1] == 'X':                   (Part G)
       print("Bad move")
       pos = 0                               (Part G)
   else:
       pos = pos + 1                         (Part E)
if move == 2:                                (Part D)
   if pos + 2 > lastPos or row[pos
+ 2] == 'X':                                 (Part F, Part G, Part H)
       print("Bad move")
       pos = 0                               (Part F, Part G)
   else:
       pos = pos + 2                         (Part E)
```

Python Example 3 (7 marks)
All design marks are achieved (**Marks A and B**)

No **Mark D** as program also adds numbers other than `1` or `2` to `pos`.

```
    move = int(input())                          (C)

    if pos + move > lastPos or row[pos +         (Part F,
 move] == 'X':                                   Part G, H)

        print("Bad move")
        pos = 0                                  (Part F,
                                                 Part G)
    else:
        pos = pos + move                         (E)
```

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

```
    move =
Convert.ToInt32(Console.ReadLine())              (C)

    If move = 1 Or move = 2 Then                 (D)
        pos += move                              (E)
    End If
    If pos > lastPos Then                        (Part F)
        pos = 0                                  (Part F)
        Console.WriteLine("Bad move")
    ElseIf row(pos) = "X" Then                   (Part G, H)
        pos = 0                                  (Part G)
        Console.WriteLine("Bad move")
    End If
```

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

VB.NET Example 2 (7 marks)

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

```
   move =
Convert.ToInt32(Console.ReadLine())
```
(C)

```
   If move = 1 Then
```
(Part D)

```
      If row(pos + 1) = "X" Then
```
(Part G)

```
          Console.WriteLine("Bad move")
          pos = 0
```
(Part G)

```
      Else
          pos = pos + 1
```
(Part E)

```
      End If
   End If
   If move = 2 Then
```
(Part D)

```
      If pos + move > lastPos Or row(pos +
2) = "X" Then
```
(Part F, Part G)

```
          Console.WriteLine("Bad move")
          pos = 0
```
(Part F, Part G)

```
      Else
          pos = pos + 2
```
(Part E)

```
      End If
   End If
```

I. Indentation

A. `Write` in place of `WriteLine`

VB.NET Example 3 (6 marks)

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

No **Mark D** as program also adds numbers other than `1` or `2` to `pos`.

```
   move =
Convert.ToInt32(Console.ReadLine())
```
(C)

```
   If pos + move > lastPos Or row(pos +
move) = "X" Then
```
(Part F, Part G)

```
      Console.WriteLine("Bad move")

      pos = 0
```
(Part F, Part G)

```
   Else
      pos = pos + move
   End If
```
(E)

I. Indentation

A. `Write` in place of `WriteLine`