Question	Part	Marking guidance	Total marks

01		**6 marks for AO3 (program)**
		
		Any fully correct answer should get 6 marks even if it does not map exactly to the following mark points.
		
		Maximum 5 marks if the answer contains any errors.
		
		Mark A: using a selection statement in the nested `WHILE` loop;
		Mark B: using a Boolean condition that tests for equality//inequality of the `image1` and `image2` variables;
		Mark C: indexing either `image1` or `image2` using the variables `i` and `j`;
		Mark D: assigning false to inverse within the selection if logically correct throughout the code (if assigned true then check for correctness);
		Mark E: incrementing `j` in the relevant place;
		Mark F: incrementing `i` in the relevant place;
		
		Example 6 mark answer:
		
		`image1 ← [ [0, 0, 0], [0, 1, 1], [1, 1, 0] ]`
		`image2 ← [ [1, 1, 1], [1, 1, 0], [0, 0, 1] ]`
		`inverse ← true`
		`i ← 0`
		`WHILE i ≤ 2`
		`  j ← 0`
		`  WHILE j ≤ 2`
		`  IF image1[i][j] = image2[i][j] THEN (A,B,C)`
		`  inverse ← false (D)`
		`  ENDIF`
		`  j ← j + 1 (E)`
		`  ENDWHILE`
		`  i ← i + 1 (F)`
		`ENDWHILE`
			6

02		**4 marks for AO2 (apply)** A record could be used to store the data of one song; An array could store all of the songs/records; One mark for one of the following, two marks for all three: • The song title could be a string • The singer could be a string • The year of release could be an integer/date.	4

Question	Part	Marking guidance	Total marks
03	1	**2 marks for AO1 (recall)** **B** A syntax error is a mistake in the grammar of the code; **D** A syntax error will stop a program from running; **R.** If more than two lozenges shaded	2

Question	Part	Marking guidance	Total marks
03	2	**Mark is for AO2 (apply)** **Mark is for AO3 (refine)** **C#** Line number: 7; Corrected line of code: `Console.WriteLine(numbers[number]);` **Python** Line number: 7; Corrected line of code: `print(numbers[number])` **VB.NET** Line number: 7; Corrected line of code: `Console.WriteLine(numbers(number))` **A**. `WriteLine` changed to `Write` as long as all other required changes have been made	2

Question	Part	Marking guidance	Total marks
03	3	**Mark is for AO2 (apply)** Array // List (of integers);	1

Question	Part	Marking guidance	Total marks
04	1	**Mark is for AO2 (apply)** **D** S; **R.** If more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
04	2	**Mark is for AO2 (apply)** **B** 2; **R.** If more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
04	3	**Mark is for AO2 (apply)** Sara; **I.** Case	1

Question	Part	Marking guidance	Total marks

04	4	**2 marks for AO3 (program)**
		
		Mark A for correct identification of **2, 4**;
		Mark B for correct identification of **1**;
		
		Model Answer
		`var ← SUBSTRING(2, 4, name1)`
		`OUTPUT (names[1] + var)`
			2

Question	Part	Marking guidance	Total marks

05	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**)
		(**C,D**)
		`while (position < 1 || position > 100) {`
		`  Console.Write("Enter card position: ");`
		`  position = Convert.ToInt32(Console.ReadLine());`
		`}`
		
		C# Example 2 (fully correct)
		All design marks are achieved (**Marks A and B**)
		(**C,D**)
		`while (position <= 0 || position >= 101) {`
		`  Console.Write("Enter card position: ");`
		`  position = Convert.ToInt32(Console.ReadLine());`
		`}`
		
		C# Example 3 (partially correct – 3 marks)
		1 design mark achieved (**Mark A**)
		(**C,D**)
		`if (position < 1 || position > 100) {`
		`  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

05	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

06	1	**4 marks for AO3 (refine)**
		1 mark for initialising `j` to 0 in correct place;
		1 mark for using `i` and `j` as indices in `ticket`;
		1 mark for incrementing `j` by 1 in correct place;
		1 mark for incrementing `i` by 1 in correct place;
		
		A. `i` and `j` in opposite indices in `ticket`
		I. Case
		
		C# Example 1 (fully correct)
		
		`int i = 0;`
		`while (i < 3) {`
		`  int j = 0;`
		`  while (j < 3) {`
		`  ticket[i, j] = generateKeyTerm();`
		`  j = j + 1;`
		`  }`
		`  i = i + 1;`
		`}`
		
		C# Example 2 (fully correct)
		
		`int i = 0;`
		`while (i < 3) {`
		`  int j = 0;`
		`  while (j < 3) {`
		`  ticket[i, j] = generateKeyTerm();`
		`  j++;`
		`  }`
		`  i++;`
		`}`
		
		Python Example 1 (fully correct)
		
		`i = 0`
		`while i < 3:`
		`  j = 0`
		`  while j < 3:`
		`  ticket[i][j] = generateKeyTerm()`
		`  j = j + 1`
		`  i = i + 1`
			4

Python Example 2 (fully correct)

```
i = 0
while i < 3:
    j = 0
    while j < 3:
        ticket[i][j] = generateKeyTerm()
        j += 1
    i += 1
```

VB.NET Example 1 (fully correct)

```
Dim i As Integer = 0
While (i < 3)
    Dim j As Integer = 0
    While (j < 3)
        ticket(i, j) = generateKeyTerm()
        j = j + 1
    End While
    i = i + 1
End While
```

VB.NET Example 2 (fully correct)

```
Dim i As Integer = 0
While (i < 3)
    Dim j As Integer = 0
    While (j < 3)
        ticket(i, j) = generateKeyTerm()
        j += 1
    End While
    i += 1
End While
```

Question	Part	Marking guidance	Total marks
06	2	**4 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 defining a subroutine called `checkWinner`; **A.** if syntax is incorrect **Mark B** for passing the entire array `ticket` as a parameter to the subroutine; **Mark C** for the use of iteration / selection to attempt to access each element in the `ticket` array; **Mark D** for the use of a selection construct for displaying the output(s); **Program Logic** **Mark E** for initialising a counter to 0 and incrementing the counter in the relevant place; **Mark F** for the correct use of indices which accesses each element in the array; **Mark G** for using a Boolean condition that tests for equality of the array elements with the correct value `"*"`; **Mark H** for outputting the word `Bingo` **and** the count of asterisks in the relevant place; **I.** Case **Maximum 7 marks** if any errors in code.	8

C# Example 1 (fully correct)

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

```
static void checkWinner(string[,] ticket)
{
   int count = 0;                                   (Part of E)
   for (int i = 0; i < 3; i++) {                    (Part of F)
      for (int j = 0; j < 3; j++) {                 (Part of F)
         if (ticket[i, j] == "*") {                 (G)
            count = count + 1;                      (Part of E)
        }
      }
   }
   if (count == 9) {                                (Part of H)
      Console.WriteLine("Bingo");                   (Part of H)
   }
   else {
      Console.WriteLine(count);
   }
}
```

C# Example 2 (fully correct)

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

```
static void checkWinner(string[,] ticket)
{
   int count = 0;                                   (Part of E)
   if (ticket[0, 0] == "*") {                       (F, G)
       count += 1; }                                (Part of E)
   if (ticket[0, 1] == "*") {
       count += 1; }
   if (ticket[0, 2] == "*") {
       count += 1; }
   if (ticket[1, 0] == "*") {
       count += 1; }
   if (ticket[1, 1] == "*") {
       count += 1; }
   if (ticket[1, 2] == "*") {
       count += 1; }
   if (ticket[2, 0] == "*") {
       count += 1; }
   if (ticket[2, 1] == "*") {
       count += 1; }
   if (ticket[2, 2] == "*") {
       count += 1; }
   if (count < 9) {
      Console.WriteLine(count);                     (Part of H)
   }
   else {
      Console.WriteLine("Bingo");                   (Part of H)
   }
```

```
}
```

C# Example 3 (fully correct)

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

```
static void checkWinner(string[,] ticket){
   int count = 0;                                   (Part of E)

   int i = 0;                                       (Part of F)
   while (i < 3) {                                  (Part of F)

      if (ticket[0, i] == "*") {                    (Part of F, G)
         count += 1; }                              (Part of E)
      i++;                                          (Part of F)
   }
   i = 0;
   while (i < 3) {
      if (ticket[1, i] == "*") {
         count += 1; }
      i++;
   }

   i = 0;
   while (i < 3) {
      if (ticket[2, i] == "*") {
         count += 1; }
      i++;
   }
   if (count < 9) {                                 (Part of H)
     Console.WriteLine(count);
   }
   else {
      Console.WriteLine("Bingo");                   (Part of H)
   }
}
```

I. Indentation in C#

I. Missing `static` in C#

Python Example 1 (fully correct)

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

```
def checkWinner(ticket):
   count = 0                                        (Part of E)
   for i in range(3):                               (Part of F)
      for j in range(3):                            (Part of F)
         if ticket[i][j] == "*":                    (Part of F, G)
            count = count + 1                       (Part of E)
   if count == 9:
      print("Bingo")                                (Part of H)
   else:
      print(count)                                  (Part of H)
```

Python Example 2 (fully correct)

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

```
def checkWinner(ticket):
    count = 0                                   (Part of E)
   if ticket[0][0] == "*":                      (F, G)
      count += 1                                (Part of E)
   if ticket[0][1] == "*":
      count += 1
   if ticket[0][2] == "*":
      count += 1
   if ticket[1][0] == "*":
      count += 1
   if ticket[1][1] == "*":
      count += 1
   if ticket[1][2] == "*":
      count += 1
   if ticket[2][0] == "*":
      count += 1
   if ticket[2][1] == "*":
      count += 1
   if ticket[2][2] == "*":
      count += 1
   if count < 9:
      print(count)                              (Part of H)
   else:
      print("Bingo")                            (Part of H)
```

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

```
def checkWinner(ticket):
    count = 0                                   (Part of E)
    i = 0
    while i < 3:                                (Part of F)
      if ticket[0][i] == "*":                   (Part of F, G)
         count = count + 1                      (Part of E)
      i = i + 1

    i = 0
    while i < 3:
       if ticket[1][i] == "*":
          count = count + 1
       i = i + 1

    i = 0
    while i < 3:
       if ticket[2][i] == "*":
          count = count + 1
       i = i + 1

    if count == 9:
       print("Bingo")                           (Part of H)
    else:
       print(count)                             (Part of H)
```

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

```
Sub checkWinner(ticket)

    Dim count As Integer = 0                    (Part of E)
    For i = 0 To 2                              (Part of F)
       For j = 0 To 2                           (Part of F)
          If ticket(i, j) = "*" Then            (G)
             count = count + 1                  (Part of E)
          End If
       Next
    Next

    If count = 9 Then
       Console.WriteLine("Bingo")               (Part of H)
    Else
       Console.WriteLine(count)                 (Part of H)
    End If
End Sub
```

VB.NET Example 2 (fully correct)

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

```
Sub checkWinner(ticket)

   Dim count As Integer = 0                    (Part of E)
   If ticket(0, 0) = "*" Then                  (F, G)
      count = count + 1                        (Part of E)
   End If
   If ticket(0, 1) = "*" Then
      count = count + 1
   End If
   If ticket(0, 2) = "*" Then
      count = count + 1
   End If
   If ticket(1, 0) = "*" Then
      count = count + 1
   End If
   If ticket(1, 1) = "*" Then
      count = count + 1
   End If
   If ticket(1, 2) = "*" Then
      count = count + 1
   End If
   If ticket(2, 0) = "*" Then
      count = count + 1
   End If
   If ticket(2, 1) = "*" Then
      count = count + 1
   End If
   If ticket(2, 2) = "*" Then
      count = count + 1
   End If
   If count < 9 Then
      Console.WriteLine(count)                 (Part of H)
   Else
      Console.WriteLine("Bingo")               (Part of H)
   End If
End Sub
```

VB.NET Example 3 (fully correct)

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

```
Sub checkWinner(ticket)

   Dim count As Integer = 0                (Part of E)
   Dim i As Integer = 0                    (Part of F)
   While i < 3                             (Part of F)
      If ticket(0,i) = "*" Then            (Part of F, G)
         count = count + 1                 (Part of E)
      End If
      i = i + 1                            (Part of F)
   End While

   i = 0
   While i < 3
      If ticket(1,i) = "*" Then
         count = count + 1
      End If
      i = i + 1
   End While

   i = 0
   While i < 3
      If ticket(2,i) = "*" Then
         count = count + 1
      End If
      i = i + 1
   End While

   If count = 9 Then
      Console.WriteLine("Bingo")           (Part of H)
   Else
      Console.WriteLine(count)             (Part of H)
   End If
End Sub
```

I. Indentation in VB.NET

Question	Part	Marking guidance	Total marks
07	1	**Mark is for AO2 (apply)** **A** 2; **R.** if more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
07	2	**Mark is for AO2 (apply)** **A** `hulk.year ← 2003`; **R.** if more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
07	3	**Mark is for AO2 (apply)** **C** `LEN(filmCollection) – 1`; **R.** if more than one lozenge shaded	1

Question	Part	Marking guidance	Total marks
07	4	**Mark is for AO2 (apply)** `antMan.beingShown ← True` // `filmCollection[0].beingShown ← True`; **A.** `antMan ← Film('Ant-Man', '12A', 2015, True)` **R.** quotation marks around 2015 **I.** Case **A.** = instead of ← **R.** `Ant-Man` **R.** Quotation marks around `True`	1

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>08</td><td>1</td><td>5 marks for AO2 (apply)

1 mark for <code>count</code> column correct;
1 mark for column <code>i</code> correct;
1 mark for the first Natalie row, including <code>j</code> and <code>result</code> correct – not including <code>i</code> and <code>count</code>;
1 mark for the second Natalie row, including <code>j</code> and <code>result</code> correct – not including <code>i</code> and <code>count</code>;
1 mark for all of Alex and Roshana rows correct as for Natalie above;

<table>
<tr><th>count</th><th>i</th><th>person</th><th>j</th><th>result</th></tr>
<tr><td>0</td><td>0</td><td>Natalie</td><td>0</td><td>78</td></tr>
<tr><td>1</td><td></td><td></td><td>1</td><td>81</td></tr>
<tr><td>2</td><td>1</td><td>Alex</td><td>0</td><td>27</td></tr>
<tr><td>3</td><td></td><td></td><td>1</td><td>51</td></tr>
<tr><td>4</td><td>2</td><td>Roshana</td><td>0</td><td>52</td></tr>
<tr><td>5</td><td></td><td></td><td>1</td><td>55</td></tr>
<tr><td>6</td><td></td><td></td><td></td><td></td></tr>
</table>

I. different rows used as long as the order within columns is clear
I. duplicate values on consecutive rows within a column
I. quotes used around letters (person column)
I. minor spelling mistakes in the person column</td><td>5</td></tr>
</table>

<table>
<tr><th>Question</th><th>Part</th><th>Marking guidance</th><th>Total marks</th></tr>
<tr><td>08</td><td>2</td><td>Mark is for AO2 (apply)

C Change line number 7 to: <code>FOR j ← 0 TO 2</code>

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

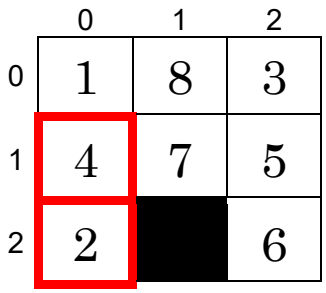
Question	Part	Marking guidance	Total marks
09		**3 marks for AO2 (apply)** L1 `1;` L2 `i;` L3 `method;` **Note to Examiners:** If the student has re-written the entire line and added in the correct missing item, award the mark.	3

Question	Part	Marking guidance	Total marks
10	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
10	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

10	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
11	1	**2 marks for AO2 (apply)**  **1 mark** for 4 in the correct position; **1 mark** for 2 in the correct position; **Maximum 1 mark** if any errors. **A**. `0` instead of blank space or any other sensible indicator for the blank space. **A**. unaffected cell contents not shown as long as it is clear which is the blank space. **A**. answers written on **Figure 15** if board is left blank.	2

Question	Part	Marking guidance	Total marks
11	2	**2 marks for AO2 (apply)** **A** Nested iteration is used; **C** The number of comparisons made between `getTile(i, j)` and `0` will be nine; **R.** if more than two lozenges shaded	2

Question	Part	Marking guidance	Total marks
11	3	**Mark is for AO2 (apply)** (The first iteration structure) is used to iterate through the rows; **Note to examiners:** award both marks (Q12.3 and Q12.4) if the student answers are correct but the opposite way around, ie 'columns' for Q12.3 and 'rows' for Q12.4	1

11	4	**Mark is for AO2 (apply)** (The second iteration structure) is used to iterate through the columns; **Note to examiners:** award both marks (Q12.3 and Q12.4) if the student answers are correct but the opposite way around, ie 'columns' for Q12.3 and 'rows' for Q12.4	1

Question	Part	Marking guidance	Total marks
11	5	**Mark is for AO2 (apply)** To find/store the position/coordinates of the blank space // to find the tile/value of `getTile` that is blank/`0`;	1

Question	Part	Marking guidance	Total marks
11	6	**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 use of a selection structure with multiple conditions // use of multiple selection structures // an iteration structure containing one selection structure; **Program Logic** **Mark B** for correctly checking three consecutive values in `getTile` (even if the wrong row/column); **Mark C** for fully correct indices used in `getTile` for the first row; **Mark D** for a structure that would output either `Yes` or `No` correctly in all circumstances, but never both; **A.** if conditions are not fully correct **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 3 marks** if any errors in code. **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.	4

C# Example 1 (fully correct)

Design mark is achieved (**Mark A**)

```
if (getTile(0, 0) + 1 == getTile(0, 1)) {
    if (getTile(0, 1) + 1 == getTile(0, 2)) {
        Console.WriteLine("Yes");
    }
    else {
        Console.WriteLine("No");
    }
}
else {
    Console.WriteLine("No");
}
```

(Part B, Part C)

(Part B, Part C)

(Part D)

(Part D)

(Part D)

I. Indentation in C#

A. `Write` in place of `WriteLine`

Note to examiners: in a nested `if` statement, all pathways must be present to award Mark D (including the part shaded yellow above).

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

```
if (getTile(0, 0) + 1 == getTile(0, 1)) {
   if (getTile(0, 0) + 2 == getTile(0, 2)) {
      Console.WriteLine("Yes");
   }
   else {
      Console.WriteLine("No");
   }
}
else {
   Console.WriteLine("No");
}
```

(Part B, Part C)
(Part B, Part C)
(Part D)
(Part D)
(Part D)

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

Note to examiners: in a nested `if` statement, all pathways must be present to award Mark D (including the part shaded yellow above).

C# Example 3 (fully correct)
Design mark is achieved (**Mark A**)

```
if ((getTile(0, 1) - getTile(0, 0) == 1) &&
(getTile(0, 2) - getTile(0, 1) == 1)) {
   Console.WriteLine("Yes");
}
else {
   Console.WriteLine("No");
}
```

(Part B, Part C)
(Part D)
(Part D)

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

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

```
if getTile(0, 0) + 1 == getTile(0, 1):          (Part B, Part C)
    if getTile(0, 1) + 1 == getTile(0, 2):      (Part B, Part C)
        print("Yes")                            (Part D)
    else:
        print("No")                             (Part D)
else:
    print("No")                                 (Part D)
```

Note to examiners: in a nested `if` statement, all pathways must be present to award Mark D (including the part shaded yellow above).

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

```
if getTile(0, 0) + 1 == getTile(0, 1):          (Part B, Part C)
    if getTile(0, 0) + 2 == getTile(0, 2):      (Part B, Part C)
        print("Yes")                            (Part D)
    else:
        print("No")                             (Part D)
else:
    print("No")                                 (Part D)
```

Note to examiners: in a nested `if` statement, all pathways must be present to award Mark D (including the part shaded yellow above).

Python Example 3 (fully correct)
Design mark is achieved (**Mark A**)

```
if getTile(0, 1) - getTile(0, 0) == 1 and getTile(0, 2) - getTile(0, 1) == 1:
    print("Yes")
else:
    print("No")
```

(Part B, Part C)
(Part D)
(Part D)

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

```
If getTile(0, 0) + 1 = getTile(0, 1) Then
    If getTile(0, 1) + 1 = getTile(0, 2) Then
        Console.WriteLine("Yes")
    Else
        Console.WriteLine("No")
    End If
Else
    Console.WriteLine("No")
End If
```

(Part B, Part C)
(Part B, Part C)
(Part D)
(Part D)
(Part D)

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

Note to examiners: in a nested `if` statement, all pathways must be present to award Mark D (including the part shaded yellow above).

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

```
If getTile(0, 0) + 1 = getTile(0, 1) Then
   If getTile(0, 0) + 2 = getTile(0, 2) Then
      Console.WriteLine("Yes")
   Else
      Console.WriteLine("No")
   End If
Else
   Console.WriteLine("No")
End If
```

(Part B, Part C)
(Part B, Part C)
(Part D)
(Part D)
(Part D)

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

Note to examiners: in a nested `if` statement, all pathways must be present to award Mark D (including the part shaded yellow above).

VB.NET Example 3 (fully correct)
Design mark is achieved (**Mark A**)

```
If getTile(0, 1) - getTile(0, 0) = 1 And
getTile(0, 2) - getTile(0, 1) = 1 Then
   Console.WriteLine("Yes")
Else
   Console.WriteLine("No")
End If
```

(Part B, Part C)
(Part D)
(Part D)

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

Question	Part	Marking guidance	Total marks
11	7	**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 an indefinite iteration structure that exists within their language; **Mark B** for the use of a selection structure or equivalent to check for a blank space; **Program Logic** **Mark C** for using user input and storing the result in two variables correctly for the row and column; **Mark D** for code that uses **both** the `solved` subroutine and the `checkSpace` subroutine in logically correct locations; **Mark E** for calling the `move` subroutine in a pathway following an `= True` condition (or equivalent) with the row and column from the user input as parameters; **Mark F** for outputting `Invalid move` when the tile does not get moved **and** asking the user to input row and column again in logically correct locations; **R**. if user is asked to re-input after the problem is solved. **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 5 marks** if any errors in code. **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**)

```
while (!solved()) {                                    (Part D)
   row =                                               (Part C)
Convert.ToInt32(Console.ReadLine());
   col =                                               (Part C)
Convert.ToInt32(Console.ReadLine());
   if (checkSpace(row, col)) {                         (Part D)
      move(row, col);                                  (E)
   }
   else {
      Console.WriteLine("Invalid move");               (F)
   }
}
```

I. Indentation in C#

A. `Write` in place of `WriteLine`

C# Example 2 (fully correct)

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

```
do {
   row =                                               (Part C)
Convert.ToInt32(Console.ReadLine());
   col =                                               (Part C)
Convert.ToInt32(Console.ReadLine());
   if (checkSpace(row, col)) {                         (Part D)
      move(row, col);                                  (E)
   }
   else {
      Console.WriteLine("Invalid move");               (F)
   }
} while (!solved);                                     (Part D)
```

I. Indentation in C#

A. `Write` in place of `WriteLine`

Python Example 1 (fully correct)

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

```
while not solved():                          (Part D)
    row = int(input())                       (Part C)
    col = int(input())                       (Part C)
    if checkSpace(row, col):                 (Part D)
        move(row, col)                       (E)
    else:
        print("Invalid move")                (F)
```

Python Example 2 (fully correct)

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

```
while solved() == False:                     (Part D)
    row = int(input())                       (Part C)
    col = int(input())                       (Part C)
    if checkSpace(row, col) == True:         (Part D)
        move(row, col)                       (E)
    else:
        print("Invalid move")                (F)
```

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

```
While Not solved()                                  (Part D)
   row = Console.ReadLine()                         (Part C)
   col = Console.ReadLine()                         (Part C)
   If checkSpace(row, col) Then                     (Part D)
      move(row, col)                                (E)
   Else
      Console.WriteLine("Invalid move")             (F)
   End If
End While
```

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**)

```
Do                                                  (Part D)
   row = Console.ReadLine()                         (Part C)
   col = Console.ReadLine()                         (Part C)
   If checkSpace(row, col) Then                     (Part D)
      move(row, col)                                (E)
   Else
      Console.WriteLine("Invalid move")             (F)
   End If
Loop Until solved()                                 (Part D)
```

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

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

```
Do While Not solved()                               (Part D)
   row = Console.ReadLine()                         (Part C)
   col = Console.ReadLine()                         (Part C)
   If checkSpace(row, col) Then                     (Part D)
      move(row, col)                                (E)
   Else
      Console.WriteLine("Invalid move")             (F)
   End If
Loop
```

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

Question	Part	Marking guidance	Total marks
12		**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());
    if (move == 1 || move == 2) {
        pos += move;
    }
    if (pos > lastPos) {
        pos = 0;
        Console.WriteLine("Bad move");
    }
    else if (row[pos] == "X") {
        pos = 0;
        Console.WriteLine("Bad move");
    }
```

(C)
(D)
(E)
(Part F)
(Part F)
(Part G, H)
(Part G)

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());
    if (pos + move > lastPos || row[pos +
move] == "X") {
        Console.WriteLine("Bad move");
        pos = 0;
    }
    else {
        pos = pos + move;
    }
}
```

(C)
(Part F, Part G, H)
(Part F, Part G)
(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`