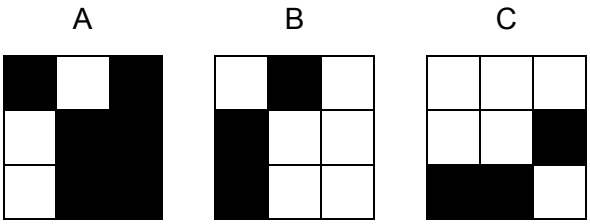
0 1 A black and white image can be represented as a two-dimensional array where:

- 0 represents a white pixel
- 1 represents a black pixel.

Two images are exact inverses of each other if:

- every white pixel in the first image is black in the second image
- every black pixel in the first image is white in the second image.

For example, B is the inverse of A but C is not the inverse of A:



A developer has started to create an algorithm that compares two 3x3 black and white images, `image1` and `image2`, to see if they are exact inverses of each other.

Complete the algorithm in pseudo-code, ensuring that, when the algorithm ends, the value of the variable `inverse` is `true` if the two images are inverses of each other or `false` if they are not inverses of each other.

The algorithm should work for any 3x3 black and white images stored in `image1` and `image2`.

- Note that indexing starts at zero.

```
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
```

[6 marks]

0 2 A developer needs to store data about thousands of songs in a program. She needs to be able to hold information on every song's title, singer and year of release.

Explain how the developer could use a combination of an array and records to store this information.

In your answer you should refer to the data types that would be used by the developer.

[4 marks]

0 3 A programmer has started to write a program using C#. Their program is shown in **Figure 6**.

The program should generate and output 10 numbers, each of which is randomly selected from the numbers in a data structure called `numbers`.

The program uses the `Random` class.

For example, `r.Next(0, 8)` would generate a random integer between 0 and 7 inclusive.

One possible output from the finished program would be 11, 14, 14, 42, 2, 56, 56, 14, 4, 2

- Line numbers are included but are not part of the program.

Figure 6

```
1   int[] numbers = { 11, 14, 56, 4, 12, 6, 42, 2 };
2   int count = 0;
3   Random r = new Random();
4   while (count < 10) {
5      count = count + 1;
6      int number = r.Next(0, 8;
7      Console.WriteLine(numbers[count]);
8   }
```

0 3 . 1 The program shown in **Figure 6** contains a syntax error.

Shade **two** lozenges to indicate the statements that are true about syntax errors.

[2 marks]

A	A syntax error can be found by testing boundary values in a program.	⬭
B	A syntax error is a mistake in the grammar of the code.	⬭
C	A syntax error is generally harder to spot than a logic error.	⬭
D	A syntax error will stop a program from running.	⬭
E	An example of a syntax error is trying to access the fifth character in a string which only contains four characters.	⬭

0 3 . 2 The program shown in **Figure 6** also contains a logic error.

Identify the line number that contains the logic error, and correct this line of the program.

Your corrected line must be written in C#.

[2 marks]

Line number _____

Corrected line _____

_____

0 3 . 3 What type of data structure is the variable `numbers`?

[1 mark]

_____

Turn over for the next question

0 4 **Figure 8** shows an algorithm represented using pseudo-code.

- Line numbers are included but are not part of the algorithm.

Figure 8

```
1   names ← ['Lily', 'Thomas']
2   name1 ← 'Sarah'
3   name2 ← 'Freddie'
4   OUTPUT name1[0]
5   OUTPUT LEN(names)
6   var ← SUBSTRING(0, 3, name1)
7   OUTPUT var
```

`SUBSTRING` returns part of a string.

For example, `SUBSTRING(3, 5, 'programming')` will return the string `'gra'`.

0 4 . 1 Shade **one** lozenge which shows the output of **line 4** from the algorithm shown in **Figure 8**.

[1 mark]

A `F` ⬭

B `Freddie` ⬭

C `Lily` ⬭

D `S` ⬭

E `Sarah` ⬭

0 4 . 2 Shade **one** lozenge which shows the output of **line 5** from the algorithm shown in **Figure 8**.

[1 mark]

A	1	⬭
B	2	⬭
C	4	⬭
D	5	⬭
E	10	⬭

0 4 . 3 State the output of **line 7** from the algorithm shown in **Figure 8**.

[1 mark]

_____

_____

0 4 . 4 Two extra lines are being added to the end of the algorithm in **Figure 8**.

Fill in the gaps so the output from the new final line will be the string `'Thomasrah'`.

[2 marks]

```
var ← SUBSTRING( _____ , _____ , name1)

OUTPUT names[ _______ ] + var
```

Turn over for the next question

0 5 A program is being developed in C# to simulate a card game.

Throughout the game each player always has 100 cards. Each card displays a number.

Players take it in turns to swap one of their cards with another random card from a set of cards until a player has a run of five numbers in sequence within their 100 cards.

0 5 . 1 **Figure 14** shows part of the program that will get a player to enter the position of a card to swap.

Figure 14

```
Console.Write("Enter card position: ");
int position = Convert.ToInt32(Console.ReadLine());
```

Extend the program in **Figure 14**. Your answer must be written in C#.

The program should keep getting the user to enter the card position until they enter a card position that is between 1 and 100 inclusive.

You **should** use meaningful variable name(s) and C# syntax in your answer.

The answer grid below contains vertical lines to help you indent your code.

[4 marks]

0 5 . 2 There are 500 cards within the game in total. Each card is numbered from 1 to 250 and each number appears twice in the whole set of cards.

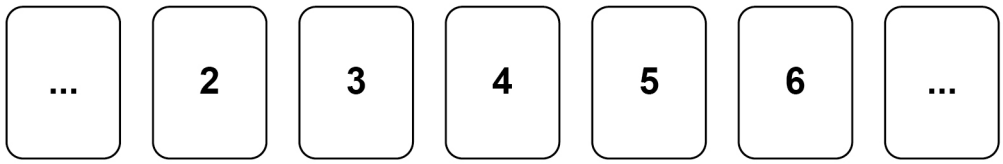
The player's 100 cards are always stored in numerical order.

When a player has a valid run of five cards within their 100 cards they have won the game.

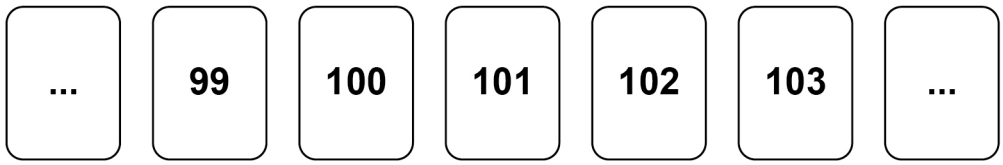
A valid run:
- consists of five cards
- can start from any position in the player's 100 cards
- the second card's value is one more than the first card's value, the third card's value is one more than the second card's value, the fourth card's value is one more than the third card's value, and the fifth card's value is one more than the fourth card's value.

Below are examples of valid runs which means a player has won.

Valid run example 1

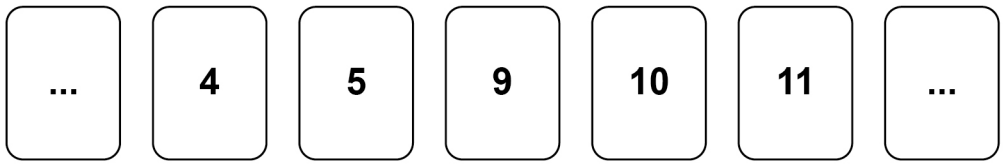


Valid run example 2

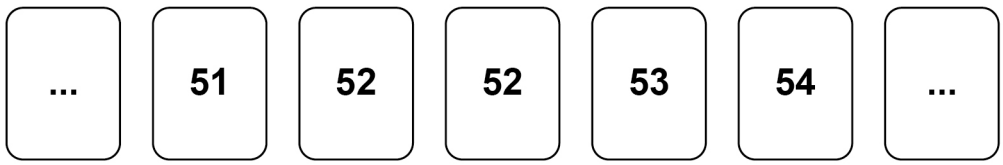


Below are examples of invalid runs.

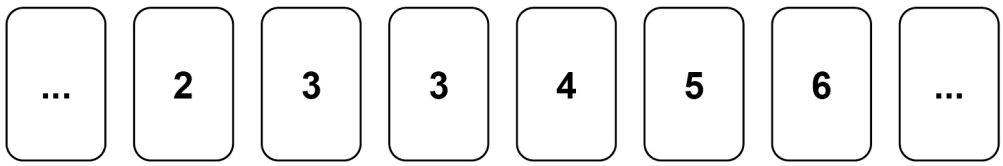
Invalid run example 1



Invalid run example 2



Invalid run example 3



Write a C# program to check if a player has a valid run of five cards within their 100 cards.

When writing your program you should assume:

- there is an array called `cards` that contains the values of the player's 100 cards
- `cards[0]` will contain the value of the first card and `cards[99]` will contain the value of the last card
- the values in `cards` are already stored in numerical order
- there is a Boolean variable called `gameWon` that has a value of `False`.

Your program should set `gameWon` to `True` if there is a valid run.

You **should** use meaningful variable name(s) and C# syntax in your answer.

The answer grid below contains vertical lines to help you indent your code.

[6 marks]

0 6 A program is being written to simulate a computer science revision game in the style of bingo.

At the beginning of the game a bingo ticket is generated with nine different key terms from computer science in a 3 x 3 grid. An example bingo ticket is provided in **Figure 15**.

Figure 15

CPU	ALU	Pixel
NOT gate	Binary	LAN
Register	Cache	Protocol

The player will then be prompted to answer a series of questions.

If an answer matches a key term on the player's bingo ticket, then the key term will be marked off automatically.

0 6 . 1 **Figure 16** shows an incomplete C# program to create a bingo ticket for a player.

The programmer has used a two-dimensional array called `ticket` to represent a bingo ticket.

The program uses a subroutine called `generateKeyTerm`. When called, the subroutine will return a random key term, eg `"CPU"`, `"ALU"`, `"NOT gate"` etc.

Complete the C# program in **Figure 16** by filling in the five gaps.

- Line numbers are included but are not part of the program.

[4 marks]

Figure 16

```
1  string[,] ticket = new string[,] {{"","",""},
                                     {"","",""},
                                     {"","",""}};

2  int i = 0;
3  while (i < 3) {

4      int j = ____ ;
5      while (j < 3) {

6          ticket[ ____ , ____ ] = generateKeyTerm();

7          _________________;
8      }

9      _________________;
10 }
```

0 6 . 2 Each time a player answers a question correctly the `ticket` array is updated; if their answer is in the `ticket` array then it is replaced with an asterisk (*).

An example of the `ticket` array containing key terms and asterisks is shown in **Figure 17**.

Figure 17

	0	1	2
0	CPU	ALU	*
1	*	*	LAN
2	Register	Cache	*

Write a subroutine in C# called `checkWinner` that will count the number of asterisks.

The subroutine should:
- take the `ticket` array as a parameter
- count the number of asterisks in the `ticket` array
- output the word `Bingo` if there are nine asterisks in the array
- output the total number of asterisks if there are fewer than nine asterisks in the array.

You **must** write your own count routine and not use any built-in count function that might be available in C#.

You **should** use meaningful variable name(s) and C# syntax in your answer.

The answer grid below contains vertical lines to help you indent your code.

[8 marks]

0 7 **Figure 8** shows an algorithm, written using pseudo-code, that uses a RECORD data structure for storing information about a film.

Each record stores four pieces of information about a film:

- film title
- certificate (eg 12A, PG)
- year the film was made
- if the film is currently being shown at a cinema.

There are records for three films and these films are stored alphabetically in an array called `filmCollection`.

The pseudo-code outputs the title of the newest of the three films.

- Part of the algorithm has been replaced by the label L1.

Figure 8

```
RECORD Film
   title : String
   certificate : String
   year : Integer
   beingShown : Boolean
ENDRECORD

hulk ← Film('Hulk', '12A', 2005, False)
ironMan ← Film('Iron Man', '12A', 2008, False)
antMan ← Film('Ant-Man', '12A', 2015, False)
filmCollection ← [antMan, hulk, ironMan]
year ← 0
position ← 0

FOR i ← 0 TO L1
   IF filmCollection[i].year > year THEN
      year ← filmCollection[i].year
      position ← i
   ENDIF
ENDFOR

OUTPUT filmCollection[position].title, ' is the
newest film'
```

0 7 . 1 How many different values can the field `beingShown` have?

Shade **one** lozenge.

[1 mark]

A 2 ⬭

B 3 ⬭

C 128 ⬭

D 256 ⬭

0 7 . 2 Which assignment statement changes the year the film *Hulk* was made to 2003?

Shade **one** lozenge.

[1 mark]

A `hulk.year ← 2003` ⬭

B `filmCollection[0].year ← 2003` ⬭

C `Film(year) ← 2003` ⬭

D `hulk(year) ← 2003` ⬭

0 7 . 3 What should the label L1 in **Figure 8** be replaced by?

Shade **one** lozenge.

[1 mark]

A 3 ⬭

B `LEN(filmCollection)` ⬭

C `LEN(filmCollection) – 1` ⬭

D `Position` ⬭

0 7 . 4 Write a pseudo-code statement that updates the `antMan` record to show that the film is currently being shown at the cinema.

[1 mark]

_____

0 8 **Figure 9** shows an algorithm, represented in pseudo-code, used to display students' test scores. The algorithm does not work as expected and the teacher wants to find the error.

The algorithm should display three test scores for each student:

- Natalie has results of 78, 81 and 72
- Alex has results of 27, 51 and 54
- Roshana has results of 52, 55 and 59.

- Line numbers are included but are not part of the algorithm.

Figure 9

```
1   names ← ['Natalie', 'Alex', 'Roshana']
2   scores ← [78, 81, 72, 27, 51, 54, 52, 55, 59]
3   count ← 0
4   FOR i ← 0 TO 2
5      person ← names[i]
6      OUTPUT 'Student: ', person
7      FOR j ← 0 TO 1
8         OUTPUT j + 1
9         result ← scores[i * 3 + j]
10        OUTPUT result
11        count ← count + 1
12     ENDFOR
13  ENDFOR
```

0 8 . 1 Complete the trace table for the algorithm shown in **Figure 9**.

You may not need to use all the rows in the table.

[5 marks]

count	i	person	j	result

0 8 . 2 How could the error in the algorithm in **Figure 9** be corrected?

Shade **one** lozenge.

[1 mark]

A Change line number 3 to: `count ← -1` ⬭

B Change line number 4 to: `FOR i ← 1 TO 4` ⬭

C Change line number 7 to: `FOR j ← 0 TO 2` ⬭

D Change line number 9 to: `result ← scores[j * 3 + i]` ⬭

Turn over for the next question

0 9 50 students have voted for the music genre they like best.

Figure 16 shows an **incomplete** algorithm, represented using pseudo-code, designed to output the highest or lowest results of the vote.

The programmer has used a two-dimensional array called `results` to store the genre and the number of votes for each genre.

Parts of the algorithm are missing and have been replaced with the labels L1 to L3.

Figure 16

```
SUBROUTINE showResults(method, numberOfGenres)
   results ← [['Pop', 'Post-Punk', 'Techno', 'Metal',
               'Dance'], ['7', '19', '14', '1', '9']]
   pos ← 0
   high ← -1
   IF method = 'HIGHEST' THEN
      FOR i ← 0 TO numberOfGenres - 1
         Votes ← STRING_TO_INT(results[L1][i])
         IF votes > high THEN
            high ← votes
            pos ← L2
         ENDIF
      ENDFOR
   ELSE
      OUTPUT 'not yet working'
   ENDIF
   IF high ≠ -1 THEN
      OUTPUT results[0][pos], ' with ', results[1][pos]
   ENDIF
ENDSUBROUTINE

OUTPUT 'Show the genre with the HIGHEST or LOWEST number
of votes? '
method ← USERINPUT
showResults( L3, 5)
```

State what should be written in place of the labels L1 to L3 in the algorithm in **Figure 16**.

[3 marks]

L1 _____

L2 _____

L3 _____

1 0 . 1 Which of the following best describes a **data structure**?

Shade **one** lozenge.

[1 mark]

A A number with a fractional part ⬭

B A value such as a whole number ⬭

C All of the data used and stored within a program ⬭

D An organised collection of values ⬭

1 0 . 2 **Figure 7** shows an **incomplete** algorithm, represented using pseudo-code.

The algorithm is used to store and manage books using records.

The algorithm should do the following:

- create a record definition called `Book` with the fields `bookName`, `author` and `price`
- create a variable for each book using the record definition.

Complete **Figure 7** by filling in the gaps using the items in **Table 2**.

- You may need to use some of the items in **Table 2** more than once.
- You will **not** need to use all the items in **Table 2**.

[3 marks]

Table 2

1	2	author
B1	B2	Book
bookName	i	Real
OUTPUT	String	Boolean

Figure 7

```
RECORD _____________

   bookName : String

   _____________ : String

   price : _____________

ENDRECORD

B1 ← Book("The Book Thief", "M Zusak", 9.99)

B2 ← _____________ ("Divergent", "V Roth", 6.55)
```

1 0 . 3 Write an algorithm using pseudo-code to display the name of the most expensive book.

The algorithm should:

- compare the price of `B1` and the price of `B2`
- output the book name of the most expensive book
- output `Neither` if the books are the same price.

The algorithm should work for any values stored in `B1` and `B2`

[3 marks]

Turn over for the next question

1 1 A program is being written to solve a sliding puzzle.

- The sliding puzzle uses a 3 x 3 board.
- The board contains eight tiles and one blank space.
- Each tile is numbered from 1 to 8
- On each turn, a tile can only move one position up, down, left, or right.
- A tile can only be moved into the blank space if it is next to the blank space.
- The puzzle is solved when the tiles are in the correct final positions.

Figure 10 shows an example of how the tiles might be arranged on the board at the start of the game with the blank space in the position (0, 1).

Figure 11 shows the correct final positions for the tiles when the puzzle is solved.

The blank space (shown in black) is represented in the program as number `0`

Figure 10

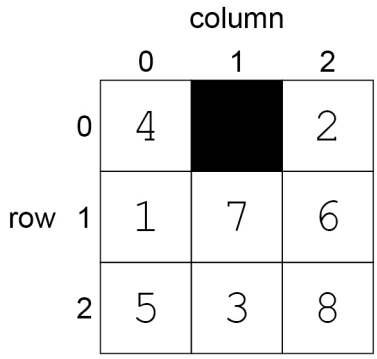


Figure 11

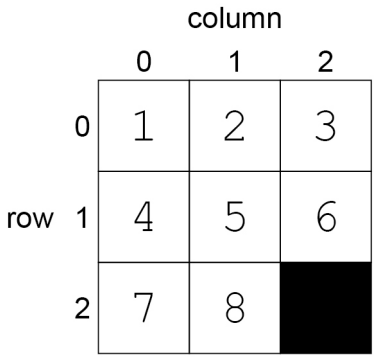


Table 3 describes the purpose of three subroutines the program uses.

Table 3

Subroutine	Purpose
`getTile(row, column)`	Returns the number of the tile on the board in the position `(row, column)` For example: • `getTile(1, 0)` will return the value `5` if it is used on the board in **Figure 12** • `getTile(1, 2)` will return the value `0` if it is used on the board in **Figure 12**.
`move(row, column)`	Moves the tile in position `(row, column)` to the blank space, if the blank space is next to that tile. If the position `(row, column)` is not next to the blank space, no move will be made. For example: • `move(0, 2)` would change the board shown in **Figure 12** to the board shown in **Figure 13** • `move(2, 0)` would not make a move if used on the board shown in **Figure 12**.
`displayBoard()`	Displays the board showing the current position of each tile.

Figure 12

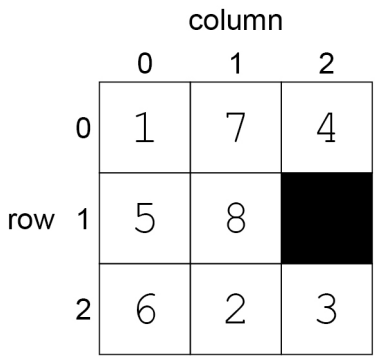
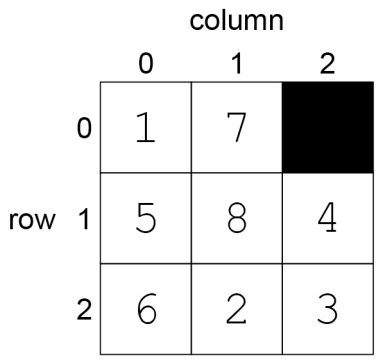


Figure 13



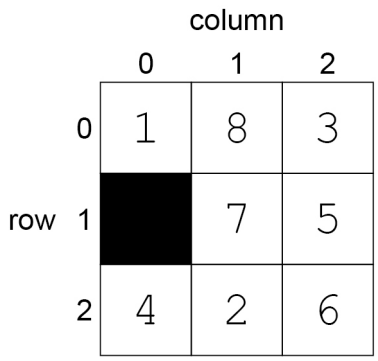
1 1 . 1 The C# program shown in **Figure 14** uses the subroutines in **Table 3**, on page 25.

The program is used with the board shown in **Figure 15**.

Figure 14

```
if (getTile(1, 0) == 0)
{
   move(2, 0);
}
if (getTile(2, 0) == 0)
{
   move(2, 1);
}
displayBoard();
```

Figure 15



Complete the board to show the new positions of the tiles after the program in **Figure 14** is run.

[2 marks]

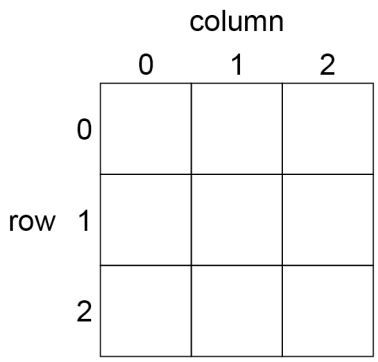


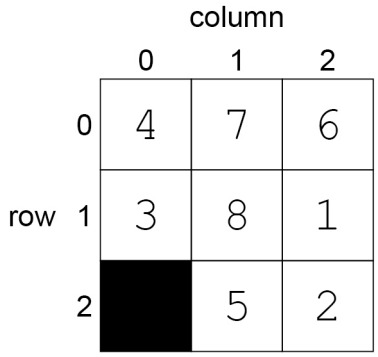
Figure 16 shows part of a C# program that uses the `getTile` subroutine from **Table 3**, on page 25.

The program is used with the board shown in **Figure 17**.

Figure 16

```
int ref1, ref2;
for (int i = 0; i < 3; i++)
{
   for (int j = 0; j < 3; j++)
   {
      if (getTile(i, j) == 0)
      {
         ref1 = i;
         ref2 = j;
      }
   }
}
```

Figure 17



1 1 . 2 Which **two** of the following statements about the program in **Figure 16** are **true** when it is used with the board in **Figure 17**?

Shade **two** lozenges.

[2 marks]

A Nested iteration is used. ⬭

B The final value of `ref1` will be `0` ⬭

C The number of comparisons made between `getTile(i, j)` and `0` will be nine. ⬭

D The outer loop, `for (int i = 0; i < 3; i++)`, will execute nine times. ⬭

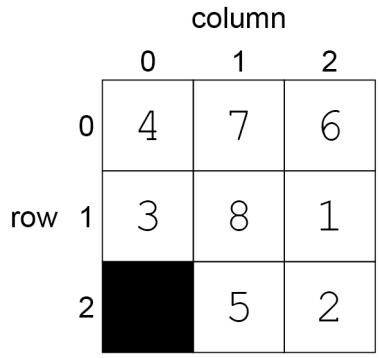
E The values of `i` and `j` do not change when the program is executed. ⬭

Figure 16 and **Figure 17** are repeated below.

Figure 16

```
int ref1, ref2;
for (int i = 0; i < 3; i++)
{
   for (int j = 0; j < 3; j++)
   {
      if (getTile(i, j) == 0)
      {
         ref1 = i;
         ref2 = j;
      }
   }
}
```

Figure 17



1 1 . 3 Explain the purpose of the **first** iteration structure in the program in **Figure 16**.

[1 mark]

1 1 . 4 Explain the purpose of the **second** iteration structure in the program in **Figure 16**.

[1 mark]

1 1 . 5 State the purpose of the program in **Figure 16**.

[1 mark]

1 1 . 6 **Table 4** shows a description of the `getTile` subroutine previously described in more detail in **Table 3**, on page 25.

Table 4

Subroutine	Purpose
`getTile(row, column)`	Returns the number of the tile on the board in the position `(row, column)`

Figure 18 and **Figure 19** show example boards.

Figure 18

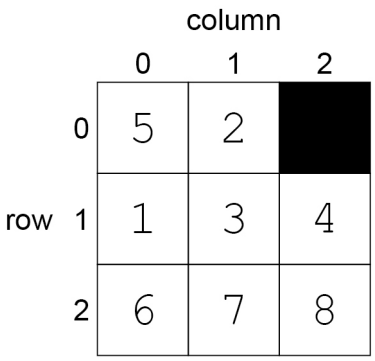
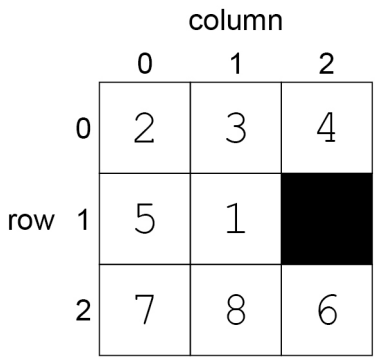


Figure 19



Write a C# program to:

- check that in the first row:
  - the second tile number is one more than the first tile number
  - the third tile number is one more than the second tile number
- display `Yes` when the row meets both conditions above
- display `No` when the row does not meet both conditions above.

For example:

- for the board in **Figure 18**, the program would display `No`
- for the board in **Figure 19**, the program would display `Yes`

You **must** use the `getTile` subroutine in your C# code.

You **should** use meaningful variable name(s) and C# syntax in your answer.

The answer grid below contains vertical lines to help you indent your code accurately.

[4 marks]

1 1 . 7 **Table 5** describes the purpose of another two subroutines the program uses.

Table 5

Subroutine	Purpose
`solved()`	Returns `true` if the puzzle has been solved. Otherwise returns `false`
`checkSpace(row, column)`	Returns `true` if there is a blank space next to the tile on the board in the position `(row, column)` Otherwise returns `false`

Table 6 shows a description of the `move` subroutine previously described in more detail in **Table 3**, on page 25.

Table 6

Subroutine	Purpose
`move(row, column)`	Moves the tile in position `(row, column)` to the blank space, if the blank space is next to that tile. If the position `(row, column)` is not next to the blank space, no move will be made.

Write a C# program to help the user solve the puzzle.

The program should:
- get the user to enter the row number of a tile to move
- get the user to enter the column number of a tile to move
- check if the tile in the position entered is next to the blank space
  - if it is, move that tile to the position of the blank space
  - if it is not, output `Invalid move`
- repeat these steps until the puzzle is solved.

You **must** use the subroutines in **Table 5** and **Table 6**.

You **should** use meaningful variable name(s) and C# syntax in your answer.

The answer grid opposite contains vertical lines to help you indent your code accurately.

[6 marks]

1 2 A programmer is writing a game.

The game uses a row of cells represented as an array. **Figure 20** shows an example.

Figure 20

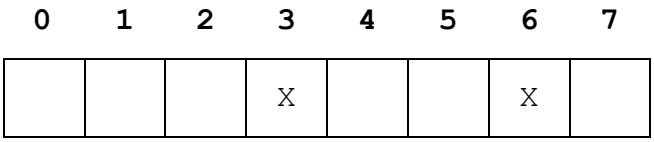


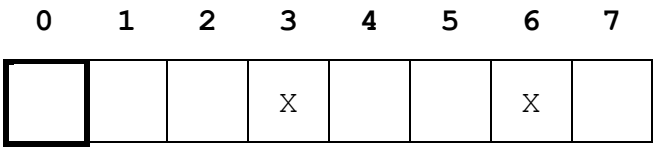
Figure 21 describes how the game is to be played.

Figure 21

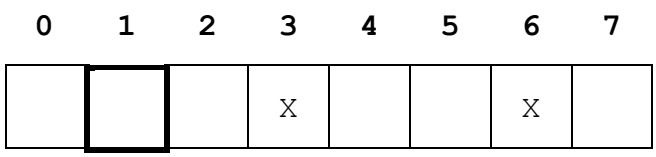
- The player starts at position `0` in a row of cells.
- The aim of the game is for the player to reach the end of the row.
- At each turn the player must enter either `1` or `2`
  - if the player enters `1`, the player's position increases by 1
  - if the player enters `2`, the player's position increases by 2
- If the player's position goes beyond the end of the row or contains an `X`:
  - the message `Bad move` is displayed
  - the player goes back to position `0`
- These steps are repeated until the player reaches the end of the row.
- If the player reaches the end of the row the game is finished.

For example, using the array in **Figure 20**:

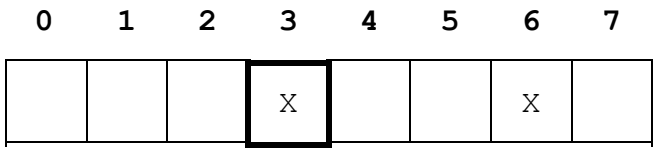
- the player starts in position `0`



- if the player enters a `1`, then they move to position `1`

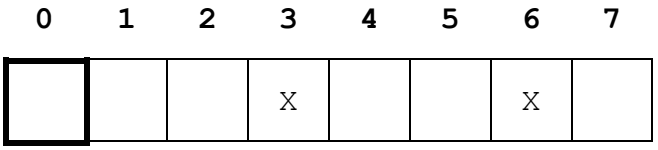


- if the player then enters a `2`, `Bad Move` is displayed as position `3` contains an `X`

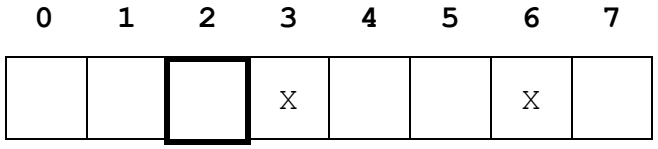



```
Bad move
```

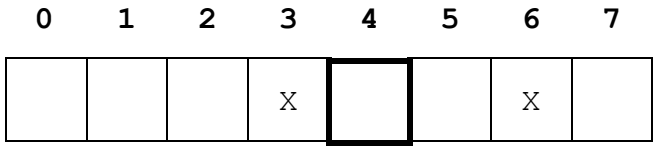
- the player then goes back to position `0`



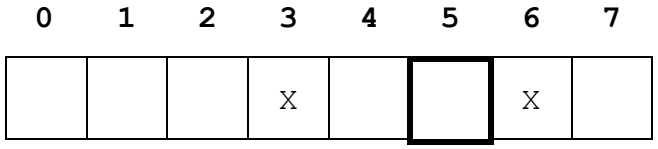
- if the player then enters a `2`, they move to position `2`



- if the player then enters a `2`, they move to position `4`



- if the player then enters a `1`, they move to position `5`



- if the player then enters a `2`, the game finishes.

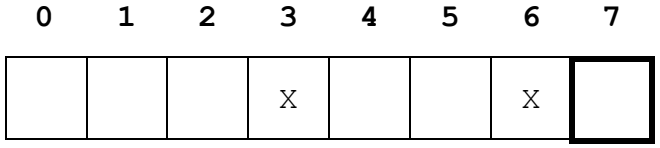


Figure 22 shows part of a C# program that will be used for the game.

Figure 22

```
int pos = 0;
int lastPos = row.Length - 1;
while (pos < lastPos)
{
```

`pos` is a variable that contains the player's current position.

Extend the program from **Figure 22** so that the game works as described in **Figure 21**, on page 37.

When writing your program you should assume:

- there is an array called `row`
- the number of `X` characters in `row` can vary
- the position of the `X` characters in `row` can vary
- the `X` characters have already been added to the array called `row`
- the `row` array can be of any length.

You **should** use meaningful variable name(s) and C# syntax in your answer.

The answer grid below contains vertical lines to help you indent your code.

[8 marks]

```
int pos = 0;
int lastPos = row.Length - 1;
while (pos < lastPos)
{
```

}				