0 1 The algorithm in **Figure 1** has been developed to automate the quantity of dog biscuits to put in a dog bowl at certain times of the day. The algorithm contains an error.

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

Figure 1

```
1     time ← USERINPUT
2     IF time = 'breakfast' THEN
3        q ← 1
4     ELSE IF time = 'lunch' THEN
5        q ← 4
6     ELSE IF time = 'dinner' THEN
7        a ← 2
8     ELSE
9        OUTPUT 'time not recognised'
10    ENDIF
11    FOR n ← 1 TO q
12       IF n < 3 THEN
13          DISPENSE_BISCUIT('chewies')
14       ELSE
15          DISPENSE_BISCUIT('crunchy')
16       ENDIF
17    ENDFOR
```

0 1 . 1 Shade **one** lozenge which shows the line number where selection is **first** used in the algorithm shown in **Figure 1**.

[1 mark]

A Line number 2 ⬭

B Line number 4 ⬭

C Line number 9 ⬭

D Line number 12 ⬭

0 1 . 2 Shade **one** lozenge which shows the line number where iteration is **first** used in the algorithm shown in **Figure 1**.

[1 mark]

A Line number 1 ⬭

B Line number 8 ⬭

C Line number 11 ⬭

D Line number 13 ⬭

0 1 . 3 Shade **one** lozenge which shows how many times the subroutine `DISPENSE_BISCUIT` would be called if the user input is `'breakfast'`.

[1 mark]

A 1 subroutine call ⬭

B 2 subroutine calls ⬭

C 3 subroutine calls ⬭

D 4 subroutine calls ⬭

0 1 . 4 Shade **one** lozenge which shows the data type of the variable `time` in the algorithm shown in **Figure 1**.

[1 mark]

A Date/Time ⬭

B String ⬭

C Integer ⬭

D Real ⬭

0 1 . 5 State how many times the subroutine `DISPENSE_BISCUIT` will be called with the parameter `'chewies'` if the user input is `'lunch'`.

[1 mark]

0 1 . 6 State how many possible values the result of the comparison `time = 'dinner'` could have in the algorithm shown in **Figure 1**.

[1 mark]

0 1 . 7 The programmer realises they have made a mistake. State the line number of the algorithm shown in **Figure 1** where the error has been made.

[1 mark]

0 1 . 8 Write **one** line of code that would correct the error found in the algorithm in **Figure 1**.

[1 mark]

0 2 The subroutine `CHAR_TO_CODE(character)` returns the integer ASCII value of a character. For example,

`CHAR_TO_CODE('a')` returns the value 97
`CHAR_TO_CODE('z')` returns the value 122
`CHAR_TO_CODE('`')` returns the value 96
`CHAR_TO_CODE('{')` returns the value 123

Develop an algorithm, using either pseudo-code or a flowchart, that:

- asks the user to enter a character
- outputs '`LOWER`' if the user has entered a lowercase character
- outputs '`NOT LOWER`' if the user has entered any other character.

You **must** use the built-in `CHAR_TO_CODE` subroutine in your answer.

[7 marks]

Turn over for the next question

0 3 The algorithm in **Figure 2** is a sorting algorithm.

- Array indexing starts at 0.
- Line numbers are included but are not part of the algorithm.

Figure 2

```
1   arr ← [4, 1, 6]
2   sorted ← false
3   WHILE sorted = false
4      sorted ← true
5      i ← 0
6      WHILE i < 2
7         IF arr[i+1] < arr[i] THEN
8            t ← arr[i]
9            arr[i] ← arr[i+1]
10           arr[i+1] ← t
11           sorted ← false
12        ENDIF
13        i ← i + 1
14     ENDWHILE
15  ENDWHILE
```

0 3 . 1 State the data type of the variable `sorted` in the algorithm shown in **Figure 2**.

[1 mark]

0 3 . 2 The identifier `sorted` is used in the algorithm shown in **Figure 2**.

Explain why this is a better choice than using the identifier `s`.

[2 marks]

0 3 . 3 Shade **one** lozenge to show which of the following contains the **false** statement about the algorithm in **Figure 2**.

[1 mark]

A The algorithm uses a named constant ⬭

B The algorithm uses indefinite iteration ⬭

C The algorithm uses nested iteration ⬭

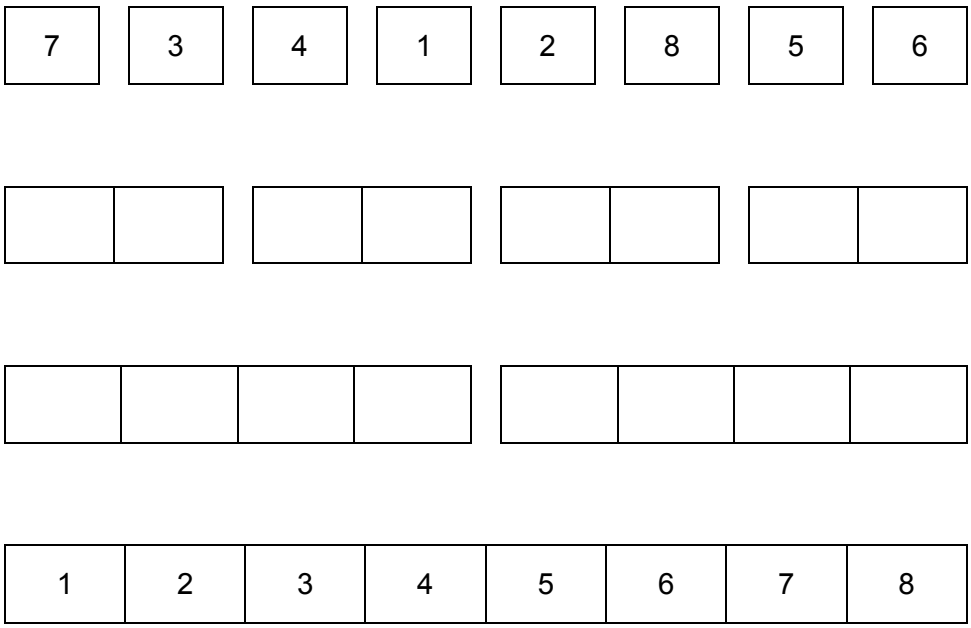
0 3 . 4 Complete the trace table for the algorithm shown in **Figure 2**. Some values have already been entered.

[6 marks]

arr			sorted	i	t
[0]	[1]	[2]			
4	1	6	false		

0 3 . 5 Fill in the values in the boxes to show how the merge part of the merge sort algorithm operates. The first and last rows have been completed for you.

[3 marks]



0 3 . 6 State **one** advantage of the merge sort algorithm compared to the sorting algorithm in **Figure 2**.

[1 mark]

0 3 . 7 A programmer implementing the algorithm in **Figure 2** decided to create it as a subroutine. Line 1 was removed and the array `arr` was made a parameter of the subroutine.

State **two** reasons why the programmer decided to implement the algorithm as a subroutine.

[2 marks]

Reason 1:

Reason 2:

0 4 Develop an algorithm using either pseudo-code or a flowchart that allows a taxi company to calculate how much a taxi fare should be.

The algorithm should:

- prompt the user to enter the journey distance in kilometres
  - the distance entered must be greater than zero
  - the user should be made to re-enter the distance until the distance entered is valid
- prompt the user to enter the number of passengers (no validation is required)
- calculate the taxi fare by
  - charging £2 for every passenger regardless of the distance
  - charging a further £1.50 for every kilometre regardless of how many passengers there are
- output the final taxi fare.

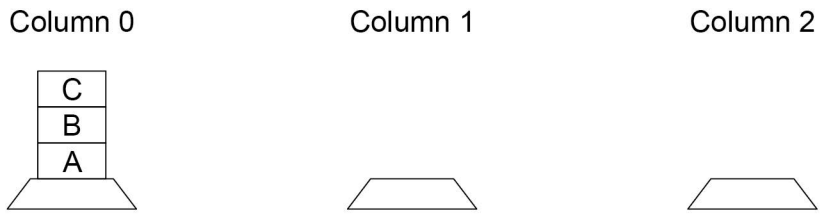
[8 marks]

Turn over for the next question

0 5 The following subroutines control the way that labelled blocks are placed in different columns.

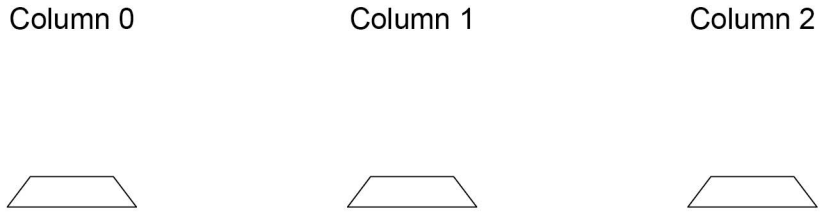
`BLOCK_ON_TOP(column)`	returns the label of the block on top of the column given as a parameter.
`MOVE(source, destination)`	moves the block on top of the `source` column to the top of the `destination` column.
`HEIGHT(column)`	returns the number of blocks in the specified column.

0 5 . 1 This is how the blocks A, B and C are arranged at the start.



Draw the final arrangement of the blocks after the following algorithm has run.

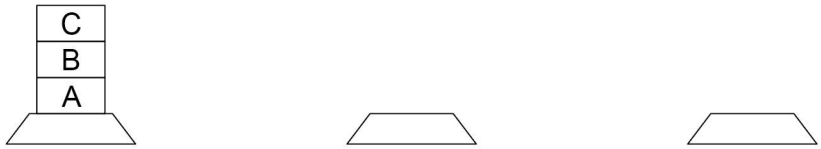
```
MOVE(0, 1)
MOVE(0, 2)
MOVE(0, 2)
```



[3 marks]

0 5 . 2 This is how the blocks A, B and C are arranged at the start.

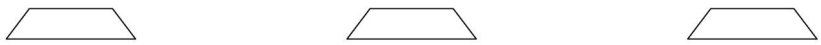
Column 0 Column 1 Column 2



Draw the final arrangement of the blocks after the following algorithm has run.

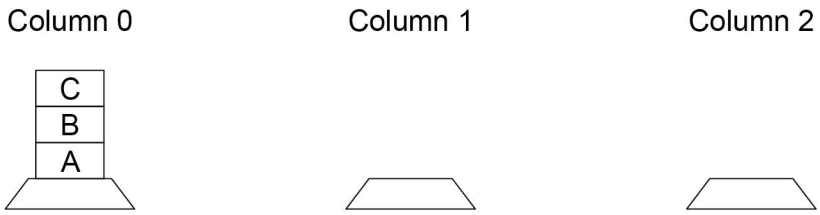
```
WHILE HEIGHT(0) > 1
   MOVE(0, 1)
ENDWHILE
MOVE(1, 2)
```

Column 0 Column 1 Column 2



[3 marks]

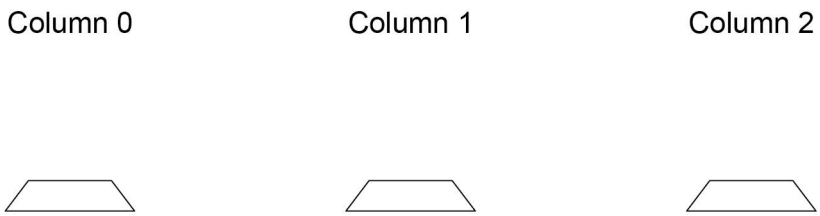
0 5 . 3 This is how the blocks A, B and C are arranged at the start.



Draw the final arrangement of the blocks after the following algorithm has run.

```
FOR c ← 0 TO 2
   IF BLOCK_ON_TOP(0) = 'B' THEN
      MOVE(0, (c+1) MOD 3)
   ELSE
      MOVE(0, (c+2) MOD 3)
   ENDIF
ENDFOR
```

This algorithm uses the MOD operator which calculates the remainder resulting from integer division. For example, `13 MOD 5 = 3`.



[3 marks]

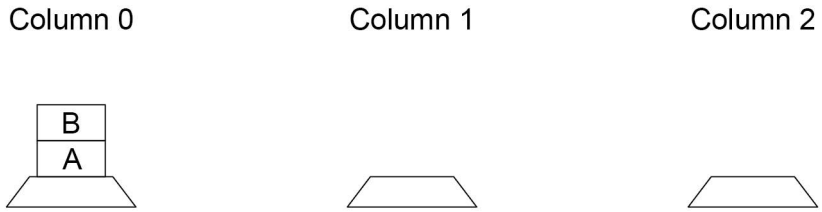
0 5 . 4 Develop an algorithm using either pseudo-code or a flowchart that will move every block from column 0 to column 1.

Your algorithm should work however many blocks start in column 0. You may assume there will always be at least one block in column 0 at the start and that the other columns are empty.

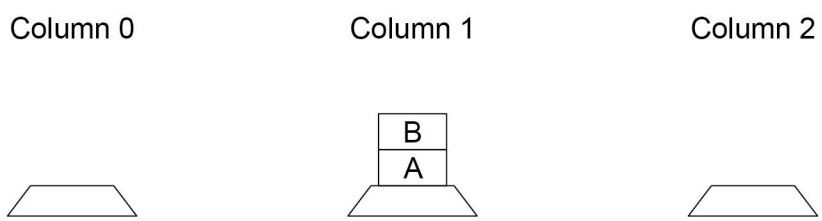
The order of the blocks must be preserved.

The `MOVE` subroutine must be used to move a block from one column to another. You should also use the `HEIGHT` subroutine in your answer.

For example, if the starting arrangement of the blocks is:



Then the final arrangement should have block B above block A:



[5 marks]

Turn over for the next question

0 6 A cake recipe uses 100 grams of flour and 50 grams of sugar for every egg used in the recipe.

Figure 3 shows the first line of an algorithm that will be used to calculate the amount of flour and sugar required based on the number of eggs being used. The number of eggs is entered by the user.

Figure 3

```
eggsUsed ← USERINPUT
```

0 6 . 1 Shade **one** lozenge to show which of the following lines of code correctly calculates the amount of flour needed in grams.

[1 mark]

A `flourNeeded ← USERINPUT` ⬭

B `flourNeeded ← eggsUsed * USERINPUT` ⬭

C `flourNeeded ← eggsUsed * 100` ⬭

D `flourNeeded ← eggsUsed * 50` ⬭

0 6 . 2 Shade **one** lozenge to show which programming technique has been used in all of the lines of code in **Question 03.1**.

[1 mark]

A Assignment ⬭

B Indefinite iteration ⬭

C Nested iteration ⬭

D Selection ⬭

0 6 . 3 The developer wants to use validation to ensure that the user can only enter a positive number of eggs, ie one egg or more. The maximum number of eggs that can be used in the recipe is eight.

Develop an algorithm, using either pseudo-code or a flowchart, so that the number of eggs is validated to ensure the user is made to re-enter the number of eggs used until a valid number is entered.

You should assume that the user will always enter an integer.

[4 marks]

0 7 . 1 Complete the trace table for the algorithm shown in **Figure 4** for when the user enters the value 750 when prompted.

[4 marks]

Figure 4

```
constant PAYLOAD_SIZE ← 250
constant HEADER_SIZE ← 50
OUTPUT 'Enter the number of bits of data to be sent'
dataToBeSent ← USERINPUT
totalSize ← PAYLOAD_SIZE + HEADER_SIZE
numberOfPackets ← 0
REPEAT
   dataToBeSent ← dataToBeSent - totalSize
   numberOfPackets ← numberOfPackets + 1
UNTIL dataToBeSent ≤ 0
```

totalSize	dataToBeSent	numberOfPackets
	750	

0 7 . 2 State why both PAYLOAD_SIZE and HEADER_SIZE from the algorithm in **Figure 4** did not need to be included in the trace table.

[1 mark]

0 7 . 3 Shade **one** lozenge to show which of the following best represents the input and output to/from the algorithm in **Figure 4**.

[1 mark]

A Input: dataToBeSent, output: numberOfPackets ⬭

B Input: numberOfPackets, output: totalSize ⬭

C Input: totalSize, output: dataToBeSent ⬭

0 7 . 4 A developer looks at the algorithm in **Figure 4** and realises that the use of iteration is unnecessary if they use a combination of the `DIV` and `MOD` operators.

- `DIV` calculates integer division, eg `11 DIV 4 = 2`
- `MOD` calculates the remainder after integer division, eg `11 MOD 4 = 3`

The programmer realises that she can rewrite the algorithm by replacing the `REPEAT-UNTIL` structure with code that uses selection, `MOD` and `DIV` instead.

Complete this new algorithm by stating the code that should be written in the boxes labelled **A**, **B** and **C**. This new algorithm should calculate the same final result for the variable `numberOfPackets` as the original algorithm in **Figure 4**.

[3 marks]

```
constant PAYLOAD_SIZE ← 250
constant HEADER_SIZE ← 50
OUTPUT 'Enter the number of bits of data to be sent'
dataToBeSent ← USERINPUT
totalSize ← PAYLOAD_SIZE + HEADER_SIZE
numberOfPackets ← dataToBeSent DIV totalSize
IF [ A ] MOD [ B ] > 0 THEN
  numberOfPackets ← [ C ]
ENDIF
```

A _____

B _____

C _____

Turn over for the next question

0 8 A developer creates the algorithm shown in **Figure 8** to provide support for users of a new brand of computer monitor (display).

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

Figure 8

```
1   OUTPUT 'Can you turn it on?'
2   ans ← USERINPUT
3   IF ans = 'no' THEN
4      OUTPUT 'Is it plugged in?'
5      ans ← USERINPUT
6      IF ans = 'yes' THEN
7         OUTPUT 'Contact supplier'
8      ELSE
9         OUTPUT 'Plug it in and start again'
10     ENDIF
11  ELSE
12     OUTPUT 'Is it connected to the computer?'
13     ans ← USERINPUT
14     IF ans = 'yes' THEN
15        OUTPUT 'Contact supplier'
16     ELSE
17        OUTPUT 'Connect it to the computer'
18     ENDIF
19  ENDIF
```

0 8 . 1 Shade **one** lozenge to show which programming technique is used on line 3 of the algorithm in **Figure 8**.

[1 mark]

A Assignment ⬭

B Iteration ⬭

C Selection ⬭

0 8 . 2 Shade **one** lozenge to show the data type of the variable `ans` in the algorithm in **Figure 8**.

[1 mark]

A Date ⬭

B Integer ⬭

C Real ⬭

D String ⬭

0 8 . 3 Regardless of what the user inputs, the same number of `OUTPUT` instructions will always execute in the algorithm shown in **Figure 8**.

State how many `OUTPUT` instructions will execute whenever the algorithm is run.

[1 mark]

0 8 . 4 The phrase `'Contact supplier'` appears twice in the algorithm in **Figure 8**.

State the **two** possible sequences of user input that would result in `'Contact supplier'` being output.

[2 marks]

Sequence 1:

Sequence 2:

0 8 . 5 Another developer looks at the algorithm shown in **Figure 8** and makes the following statement.

> "At the moment if the user enters 'y' or 'n' they will sometimes get unexpected results. This problem could have been avoided."

Explain why this problem has occurred and describe what would happen if a user entered 'y' or 'n' instead of 'yes' or 'no'.

You may include references to line numbers in the algorithm where appropriate. You do **not** need to include any additional code in your answer.

[3 marks]

0 9 The algorithms shown in **Figure 4** and **Figure 5** both have the same purpose.

The operator `LEFTSHIFT` performs a binary shift to the left by the number indicated.

For example, `6 LEFTSHIFT 1` will left shift the number `6` by one place, which has the effect of multiplying the number `6` by two giving a result of 12

Figure 4

```
result ← number LEFTSHIFT 2
result ← result - number
```

Figure 5

```
result ← 0
FOR x ← 1 TO 3
   result ← result + number
ENDFOR
```

0 9 . 1 Complete the trace table for the algorithm shown in **Figure 4** when the initial value of `number` is 4

You may not need to use all rows of the trace table.

[2 marks]

result

0 9 . 2 Complete the trace table for the algorithm shown in **Figure 5** when the initial value of `number` is 4

You may not need to use all rows of the trace table.

[2 marks]

x	result

0 9 . 3 The algorithms in **Figure 4** and **Figure 5** have the same purpose.

State this purpose.

[1 mark]

0 9 . 4 Explain why the algorithm shown in **Figure 4** can be considered to be a more efficient algorithm than the algorithm shown in **Figure 5**.

[1 mark]

Turn over for the next question

1 0 An application allows only two users to log in. Their usernames are stated in **Table 1** along with their passwords.

Table 1

username	password
`gower`	`9Fdg3`
`tuff`	`888rG`

Develop an algorithm, using either pseudo-code **or** a flowchart, that authenticates the user. The algorithm should:

- get the user to enter their username and password
- check that the combination of username and password is correct and, if so, output the string `'access granted'`
- get the user to keep re-entering their username and password until the combination is correct.

[6 marks]

1 1 Develop an algorithm, using either pseudo-code **or** a flowchart, that helps an ice cream seller in a hot country calculate how many ice creams they are likely to sell on a particular day. Your algorithm should:

- get the user to enter whether it is the weekend or a weekday
- get the user to enter the temperature forecast in degrees Celsius (they should enter a number between 20 and 45 inclusive; if the number falls outside of this range then they should be made to re-enter another number until they enter a valid temperature)
- calculate the number of ice creams that are likely to be sold using the following information:
  - 100 ice creams are likely to be sold if the temperature is between 20 and 30 degrees inclusive,
  - 150 ice creams are likely to be sold if the temperature is between 31 and 38 degrees inclusive,
  - and 120 ice creams are likely to be sold if the temperature is higher than 38 degrees
- double the estimate if it is a weekend
- output the estimated number of ice creams that are likely to be sold.

[9 marks]

1 2 A developer has written a set of subroutines to control an array of lights. The lights are indexed from zero. They are controlled using the subroutines in **Table 2**.

Table 2

Subroutine	Explanation
`SWITCH(n)`	If the light at index `n` is on it is set to off. If the light at index `n` is off it is set to on.
`NEIGHBOUR(n)`	If the light at index `(n+1)` is on, the light at index `n` is also set to on. If the light at index `(n+1)` is off, the light at index `n` is also set to off.
`RANGEOFF(m, n)`	All the lights between index `m` and index `n` (but **not** including `m` and `n`) are set to off.

Array indices are shown above the array of lights.

For example, if the starting array of the lights is

0	1	2	3
off	on	off	on

Then after the subroutine call `SWITCH(2)` the array of lights will become

0	1	2	3
off	on	on	on

And then after the subroutine call `NEIGHBOUR(0)` the array of lights will become

0	1	2	3
on	on	on	on

Finally, after the subroutine call `RANGEOFF(0, 3)` the array of lights will become

0	1	2	3
on	off	off	on

1 2 . 1 If the starting array of lights is

0	1	2	3	4	5	6
on	off	off	on	off	off	on

What will the array of lights become after the following algorithm has been followed?

```
a ← 2
SWITCH(a)
SWITCH(a + 1)
NEIGHBOUR(a - 2)
```

Write your final answer in the following array

[3 marks]

0	1	2	3	4	5	6

1 2 . 2 If the starting array of lights is

0	1	2	3	4	5	6
off	off	on	off	on	on	on

What will the array of lights become after the following algorithm has been followed?

```
FOR a ← 0 TO 2
   SWITCH(a)
ENDFOR
b ← 8
RANGEOFF((b / 2), 6)
NEIGHBOUR(b - 4)
```

Write your final answer in the following array

[3 marks]

0	1	2	3	4	5	6

1 2 . 3 If the starting array of lights is

0	1	2	3	4	5	6
off	on	off	on	off	on	off

What will the array of lights become after the following algorithm has been followed?

```
a ← 0
WHILE a < 3
   SWITCH(a)
   b ← 5
   WHILE b ≤ 6
      SWITCH(b)
      b ← b + 1
   ENDWHILE
   a ← a + 1
ENDWHILE
```

Write your final answer in the following array

[3 marks]

0	1	2	3	4	5	6

1 2 . 4 If the starting array of lights is

0	1	2	3	4	5	6
on	on	on	on	on	on	on

Write an algorithm, using **exactly three** subroutine calls, that means the final array of lights will be

0	1	2	3	4	5	6
off	off	off	off	off	off	off

You must use each of the subroutines `SWITCH`, `NEIGHBOUR` and `RANGEOFF` **exactly once** in your answer. If you do not do this you may still be able to get some marks.

[3 marks]

1 3 An algorithm, that uses the modulus operator, has been represented using pseudo-code in **Figure 1**.

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

Figure 1

```
1  i ← USERINPUT
2  IF i MOD 2 = 0 THEN
3     OUTPUT i * i
4  ELSE
5     OUTPUT i
6  ENDIF
```

The modulus operator is used to calculate the remainder after dividing one integer by another.

For example:
- `14 MOD 3` evaluates to `2`
- `24 MOD 5` evaluates to `4`

1 3 . 1 Shade **one** lozenge that shows the line number where selection is **first** used in the algorithm in **Figure 1**.

[1 mark]

A Line number 1 ⬭

B Line number 2 ⬭

C Line number 3 ⬭

D Line number 4 ⬭

1 3 . 2 Shade **one** lozenge that shows the output from the algorithm in **Figure 1** when the user input is `4`

[1 mark]

A `0` ⬭

B `2` ⬭

C `4` ⬭

D `8` ⬭

E `16` ⬭

1 3 . 3 Shade **one** lozenge that shows the line number where assignment is **first** used in the algorithm in **Figure 1**.

[1 mark]

A Line number 1 ⬭

B Line number 2 ⬭

C Line number 3 ⬭

D Line number 4 ⬭

1 3 . 4 Shade **one** lozenge that shows the line number that contains a relational operator in the algorithm in **Figure 1**.

[1 mark]

A Line number 1 ⬭

B Line number 2 ⬭

C Line number 3 ⬭

D Line number 4 ⬭

Figure 1 has been included again below.

Figure 1

```
1  i ← USERINPUT
2  IF i MOD 2 = 0 THEN
3     OUTPUT i * i
4  ELSE
5     OUTPUT i
6  ENDIF
```

1 3 . 5 Shade **one** lozenge to show which of the following is a **true** statement about the algorithm in **Figure 1**.

[1 mark]

A This algorithm uses a Boolean operator. ⬭

B This algorithm uses a named constant. ⬭

C This algorithm uses iteration. ⬭

D This algorithm uses the multiplication operator. ⬭

1 3 . 6 **Figure 2** shows an implementation of the algorithm in **Figure 1** using the C# programming language.

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

Figure 2

```
1  Console.Write("Enter a number: ");
2  int i = Convert.ToInt32(Console.ReadLine());
3  if (i % 2 == 0) {
4     Console.WriteLine(i * i);
5  }
6  else {
7     Console.WriteLine(i);
8  }
```

The program in **Figure 2** needs to be changed so that it repeats five times using **definite** (count controlled) iteration.

Shade **one** lozenge next to the program that does this correctly.

[1 mark]

A

```
for (int x = 0; x < 5; x++) {
   Console.Write("Enter a number: ");
   int i = Convert.ToInt32(Console.ReadLine());
   if (i % 2 == 0) {
       Console.WriteLine(i * i);
   }
   else {
       Console.WriteLine(i);
   }
}
```

○

B

```
for (int x = 0; x < 6; x++) {
   Console.Write("Enter a number: ");
   int i = Convert.ToInt32(Console.ReadLine());
   if (i % 2 == 0) {
       Console.WriteLine(i * i);
   }
   else {
      Console.WriteLine(i);
   }
}
```

○

C

```
int x = 1;
while (x != 6) {
   Console.Write("Enter a number: ");
   int i = Convert.ToInt32(Console.ReadLine());
   if (i % 2 == 0) {
       Console.WriteLine(i * i);
   }
   else {
       Console.WriteLine(i);
   }
   x = x + 1;
}
```

○

D

```
int x = 6;
while (x != 0) {
   Console.Write("Enter a number: ");
   int i = Convert.ToInt32(Console.ReadLine());
   if (i % 2 == 0) {
      Console.WriteLine(i * i);
   }
   else {
       Console.WriteLine(i);
   }
   x = x - 1;
}
```

○

1 4 Write a C# program to check if an email address has been entered correctly by a user.

Your program must:

- get the user to input an email address
- get the user to input the email address a second time
- output the message `Match` **and** output the email address if the email addresses entered are the same
- output the message `Do not match` if the email addresses entered are not the same.

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.

[5 marks]

1 5 Write a C# program that calculates the value of a bonus payment for an employee based on how many items they have sold and the number of years they have been employed.

The program should:

- get the user to input the number of items sold
- get the user to input the number of years employed
- output the value of the bonus payment:
  - if the years of employment is less than or equal to 2 **and** the number of items sold is greater than 100, then the bonus will be the number of items sold multiplied by 2
  - if the years of employment is greater than 2, then the bonus will be the number of items sold multiplied by 10
  - otherwise, the bonus is 0

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.

[7 marks]

1 6 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.

1 6 . 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]

1 6 . 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.

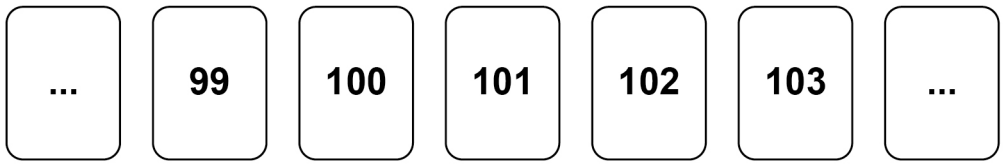
When a player has a valid run of five cards within their 100 cards they have won the game.

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.

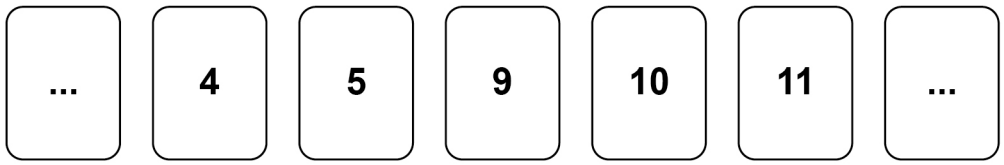
Valid run example 1

Valid run example 2

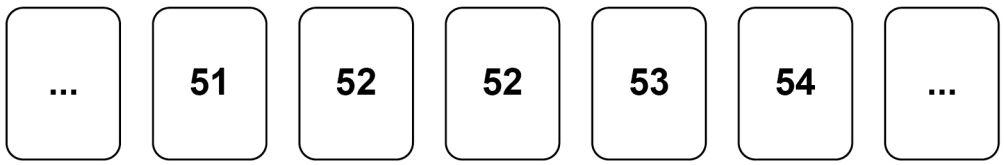


Below are examples of invalid runs.

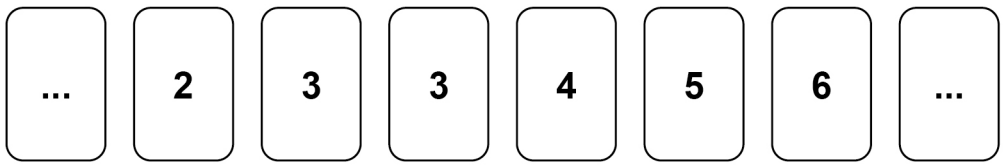
Invalid run example 1



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]

1 7 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.

1 7 . 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 }
```

1 7 . 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]

1 8 **Figure 2** shows an algorithm that uses integer division which has been represented using pseudo-code.

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

Figure 2

```
1   again ← True
2   WHILE again = True
3      a ← USERINPUT
4      IF a > 0 THEN
5         counter ← 0
6         WHILE a > 0
7            a ← a DIV 3
8            counter ← counter + 1
9         ENDWHILE
10     ELSE
11        again ← False
12     ENDIF
13     OUTPUT a
14  ENDWHILE
```

Integer division is the number of times one integer divides into another, with the remainder ignored.

For example:

- `14 DIV 5` evaluates to 2
- `25 DIV 3` evaluates to 8

1 8 . 1 Where is iteration **first** used in the algorithm in **Figure 2**?

Shade **one** lozenge.

[1 mark]

A Line number 2 ⬭

B Line number 4 ⬭

C Line number 6 ⬭

D Line number 11 ⬭

1 8 . 2 In the algorithm in **Figure 2**, what will be output when the user input is `10`?

Shade **one** lozenge.

[1 mark]

A `0` ⬭

B `1` ⬭

C `2` ⬭

D `4` ⬭

1 8 . 3 In the algorithm in **Figure 2**, what is the largest possible value of the variable `counter` when the user input is `36`?

Shade **one** lozenge.

[1 mark]

A `0` ⬭

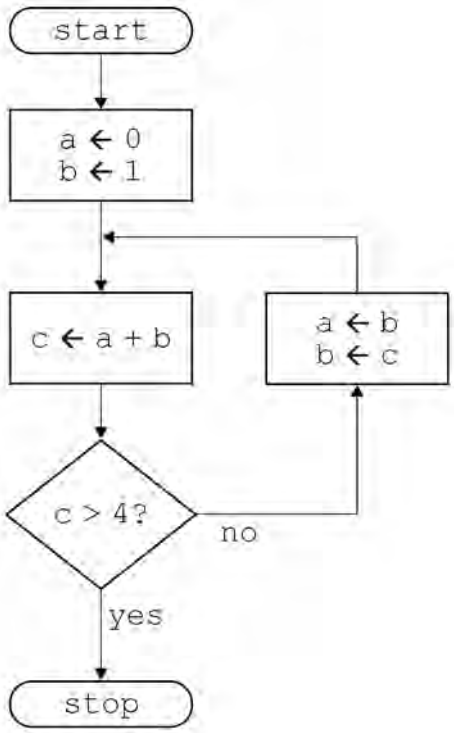
B `2` ⬭

C `4` ⬭

D `5` ⬭

1 9 **Figure 4** shows an algorithm presented as a flowchart.

Figure 4



Complete the trace table for the algorithm in **Figure 4**.

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

[3 marks]

a	b	c

2 0 A theme park charges £15 per person for a daily ticket. If there are six or more people in a group, the group is given a £5 discount.

Write a C# program to calculate the total charge for a group of people visiting the theme park.

The program must:
- get the user to enter the number of people in a group
- calculate the total charge by:
  - charging £15 per person
  - reducing the total charge by £5 if there are six or more people
- output the total charge.

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]

2 1 **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
```

2 1 . 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

2 1 . 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` ⬭

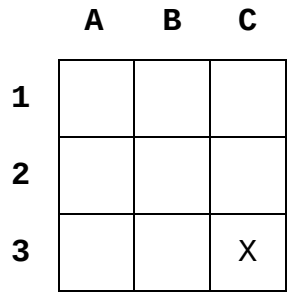
C Change line number 7 to: `FOR j ← 0 TO 2` ⬭

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

Turn over for the next question

2 2 A programmer is writing a game. The game uses a 3 x 3 grid containing nine squares.

Figure 14



In the game, a square on the grid is referred to by a letter and a number.
For example, square **C3** in **Figure 14** contains an X.

Figure 15 shows part of a C# program that checks the grid reference entered by a player.
The grid reference is valid if:

- there are exactly two characters
- the first character entered is A, B or C
- the second character entered is 1, 2 or 3.

Figure 15

```
bool check = false;
while (check == false) {
   string square = "";
   while (square.Length != 2) {
      Console.Write("Enter grid reference (eg C2): ");
      square = Console.ReadLine();
      square = square.ToUpper();
   }
```

The C# function `ToUpper()` converts letters into uppercase, eg `b1` would be converted to `B1`

Extend the program from **Figure 15** so it completes the other checks needed to make sure a valid grid reference is entered.

Your extended program must:

- use the variable `check`
- repeat the following steps until a valid grid reference is entered:
  - get the user to enter a grid reference
  - output an appropriate message if the grid reference entered is not valid.

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

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

[6 marks]

```
bool check = false;
while (check == false) {
    string square = "";
    while (square.Length != 2) {
        Console.Write("Enter grid reference (eg C2): ");
        square = Console.ReadLine();
        square = square.ToUpper();
    }
}
```

2 3 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 _____

Turn over for the next question

2 4 A group of people have a meal in a restaurant. Instead of one person paying for the whole meal, each person will pay for what they eat.

Write a C# program that asks each person in the group how much they are paying towards the meal and works out when the bill is fully paid. Each person can pay a different amount.

The program should:

- get the user to enter the total amount of the bill
- get a person to enter how much they are paying towards the bill
- subtract the amount entered from the bill:
  - if the amount left to pay is more than `0`, output how much is left to pay and repeat until the amount left to pay is `0` or less
  - if the amount left to pay is `0`, then output the message `Bill paid`
  - if the amount left to pay is less than `0`, then output the message `Tip is` and the difference between the amount left to pay and `0`

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]

2 5 Question **16** is about a dice game played against a computer.

The aim of the game is to get as close to a score of 21 as you can, without going over 21. If your score goes over 21 then you lose.

The player's score starts at 0.

For each turn:

- two dice (each numbered from 1 to 6) are rolled
- the total of the two dice rolls is added to the player's score
- the value of each dice and the player's new total score is output
- if the current score is less than 21, the player is asked if they would like to roll the dice again: if the player says yes, they get another turn; otherwise, the game ends.

At the end of the game, the program should work as follows:

- if the final score is 21, output a message to say the player has won
- if the final score is greater than 21, output a message to say the player has lost
- if the final score is less than 21, the program generates a random number between 15 and 21 inclusive:
  - if this random number is greater than the player's final score, output a message to say the player has lost
  - otherwise, output a message to say the player has won.

Figure 17 shows the output of a program that plays this dice game.

Figure 17

```
Roll 1: 1
Roll 2: 4
Current score: 5
Would you like to roll again? yes

Roll 1: 1
Roll 2: 6
Current score: 12
Would you like to roll again? yes

Roll 1: 1
Roll 2: 2
Current score: 15
Would you like to roll again? yes

Roll 1: 6
Roll 2: 1
Current score: 22
You lost!
```

Write a C# program to simulate this game.

The first line has been written for you in the answer grid.

The dice rolls are carried out by the program generating random numbers between 1 and 6. You will need to use the C# function `r.Next(a, b)` which generates a random integer in the range `a` to `b` starting at `a` but finishing one before `b`.

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.

[11 marks]

```
Random r = new Random();
```


2 6 The algorithm in **Figure 2** has been developed to automate the quantity of dog biscuits to put in a dog bowl at certain times of the day.

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

Figure 2

```
1     time ← USERINPUT
2     IF time = 'breakfast' THEN
3        q ← 1
4     ELSE IF time = 'lunch' THEN
5        q ← 4
6     ELSE IF time = 'dinner' THEN
7        q ← 2
8     ELSE
9        OUTPUT 'time not recognised'
10    ENDIF
11    FOR n ← 1 TO q
12       IF n < 3 THEN
13          DISPENSE_BISCUIT('chewies')
14       ELSE
15          DISPENSE_BISCUIT('crunchy')
16       ENDIF
17    ENDFOR
```

2 6 . 1 Shade **one** lozenge which shows the line number where selection is **first** used in the algorithm shown in **Figure 2**.

[1 mark]

A Line number 2 ⬭

B Line number 4 ⬭

C Line number 9 ⬭

D Line number 12 ⬭

2 6 . 2 Shade **one** lozenge which shows the line number where iteration is **first** used in the algorithm shown in **Figure 2**.

[1 mark]

A Line number 1 ⬭

B Line number 8 ⬭

C Line number 11 ⬭

D Line number 13 ⬭

2 6 . 3 Shade **one** lozenge which shows how many times the subroutine `DISPENSE_BISCUIT` would be called if the user input is `'breakfast'` in **Figure 2**.

[1 mark]

A 1 subroutine call ⬭

B 2 subroutine calls ⬭

C 3 subroutine calls ⬭

D 4 subroutine calls ⬭

2 6 . 4 Shade **one** lozenge which shows the data type of the variable `time` in the algorithm shown in **Figure 2**.

[1 mark]

A Date/Time ⬭

B String ⬭

C Integer ⬭

D Real ⬭

2 6 . 5 State how many times the subroutine `DISPENSE_BISCUIT` will be called with the parameter `'chewies'` if the user input is `'lunch'` in **Figure 2.**

[1 mark]

Turn over for the next question

2 7 A programmer has written a C# program that asks the user to input two integers and then output which of the two integers is the largest.
Complete the program by filling in the gaps using the information in **Figure 3**.
Each item in **Figure 3** should only be used once.

[5 marks]

Figure 3

Console.Write	num1	num2	output
else	<	>	else if
string	double	int	

```
int num1;

__________ num2;

Console.WriteLine("Enter a number: ");

num1 = int.Parse(Console.ReadLine());

Console.WriteLine("Enter another number: ");

num2 = int.Parse(Console.ReadLine());

if (num1 > num2)

{

    Console.WriteLine("    _________ is bigger.");

}

else

if (num1 _____________ num2)

{

    Console.WriteLine("    _________ is bigger.");

}

____________________

{

    Console.WriteLine("The numbers are equal.");

}
```

2 8 Write a C# program that allows a taxi company to calculate how much a taxi fare should be.

The program should:

- allow the user to enter the journey distance in kilometres (no validation is required)
- allow the user to enter the number of passengers (no validation is required)
- calculate the taxi fare by
  - charging £2 for every passenger regardless of the distance
  - charging a further £1.50 for every kilometre regardless of how many passengers there are
- output the final taxi fare.

You **should** use meaningful variable name(s), correct syntax and indentation in your answer.

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

[7 marks]

2 9 Write a C# program that inputs a password and checks if it is correct.

Your program should work as follows:

- input a password and store it in a suitable variable
- if the password entered is equal to `secret` display the message `Welcome`
- if the password entered is not equal to `secret` display the message `Not welcome`.

You **should** use meaningful variable name(s), correct syntax and indentation in your answer.

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

[5 marks]

3 0 The algorithm in **Figure 4** is a sorting algorithm.

- Array indexing starts at 0.
- Line numbers are included but are not part of the algorithm.

Figure 4

```
1   arr ← [4, 1, 6]
2   swapsMade ← false
3   WHILE swapsMade = false
4      swapsMade ← true
5      i ← 0
6      WHILE i < 2
7         IF arr[i+1] < arr[i] THEN
8            t ← arr[i]
9            arr[i] ← arr[i+1]
10           arr[i+1] ← t
11           swapsMade ← false
12        ENDIF
13        i ← i + 1
14     ENDWHILE
15  ENDWHILE
```

3 0 . 1 State the data type of the variable `swapsMade` in the algorithm shown in **Figure 4**.

[1 mark]

3 0 . 2 The identifier `swapsMade` is used in the algorithm shown in **Figure 4**.

Explain why this is a better choice than using the identifier `s`.

[2 marks]

3 0 . 3 Shade **one** lozenge to show which of the following contains the **false** statement about the algorithm in **Figure 4**.

[1 mark]

A The algorithm uses a named constant. ⬭

B The algorithm uses indefinite iteration. ⬭

C The algorithm uses nested iteration. ⬭

3 0 . 4 Complete the trace table for the algorithm shown in **Figure 4**. Some values have already been entered.

[6 marks]

arr			swapsMade	i	t
[0]	[1]	[2]			
4	1	6	false		

3 1 Write a C# program that inputs a character and checks to see if it is lowercase or not.

Your program should work as follows:

- gets the user to enter a character and store it in a suitable variable
- determines if the entered character is a lowercase character
- outputs `LOWER` if the user has entered a lowercase character
- outputs `NOT LOWER` if the user has entered any other character.

You **should** use meaningful variable name(s), correct syntax and indentation in your answer.

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

[7 marks]

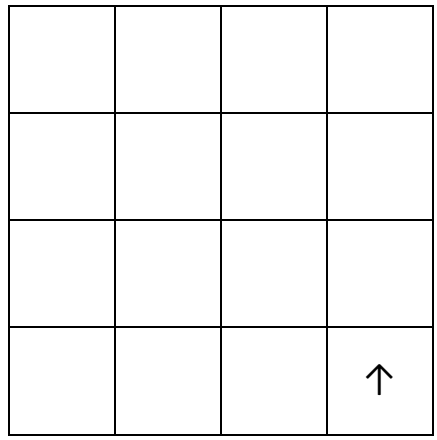
3 2 Four separate subroutines have been written to control a robot.

- `Forward(n)` moves the robot `n` squares forward.
- `TurnLeft()` turns the robot 90 degrees left.
- `TurnRight()` turns the robot 90 degrees right.
- `ObjectAhead()` returns `true` if the robot is facing an object in the next square or returns `false` if this square is empty.

3 2 . 1 Draw the path of the robot through the grid below if the following program is executed (the robot starts in the square marked by the ↑ facing in the direction of the arrow).

```
Forward(2)
TurnLeft()
Forward(1)
TurnRight()
Forward(1)
```

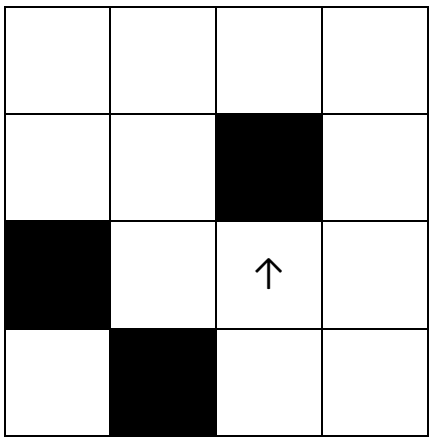
[3 marks]



3 2 . 2 Draw the path of the robot through the grid below if the following program is executed (the robot starts in the square marked by the ↑ facing in the direction of the arrow). If a square is black then it contains an object.

```
WHILE ObjectAhead() = true
  TurnLeft()
    IF ObjectAhead() = true THEN
      TurnRight()
      TurnRight()
    ENDIF
  Forward(1)
ENDWHILE
Forward(1)
```

[3 marks]

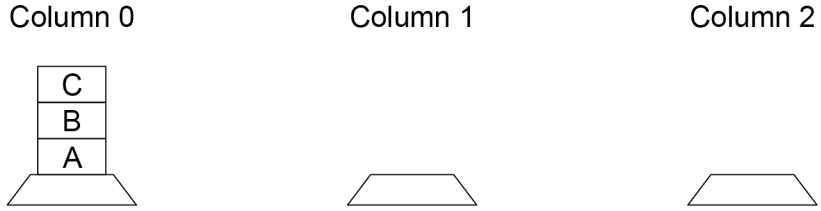


Turn over for the next question

3 3 The following subroutines control the way that labelled blocks are placed in different columns.

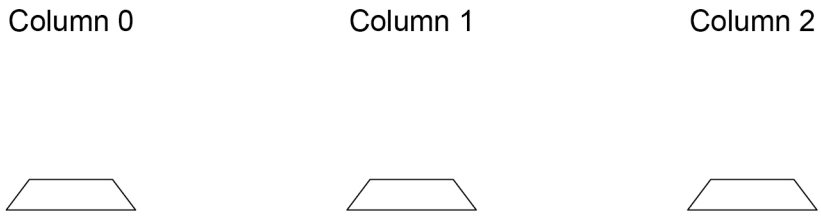
`BLOCK_ON_TOP(column)`	returns the label of the block on top of the column given as a parameter.
`MOVE(source, destination)`	moves the block on top of the `source` column to the top of the `destination` column.
`HEIGHT(column)`	returns the number of blocks in the specified column.

3 3 . 1 This is how the blocks A, B and C are arranged at the start.



Draw the final arrangement of the blocks after the following algorithm has run.

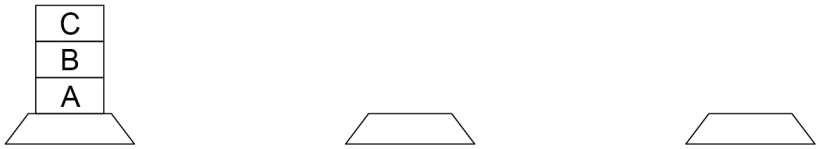
```
MOVE(0, 1)
MOVE(0, 2)
MOVE(0, 2)
```



[3 marks]

3 3 . 2 This is how the blocks A, B and C are arranged at the start.

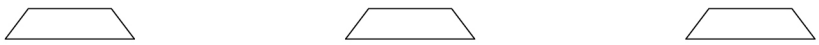
Column 0 Column 1 Column 2



Draw the final arrangement of the blocks after the following algorithm has run.

```
WHILE HEIGHT(0) > 1
   MOVE(0, 1)
ENDWHILE
MOVE(1, 2)
```

Column 0 Column 1 Column 2



[3 marks]

Turn over for the next question

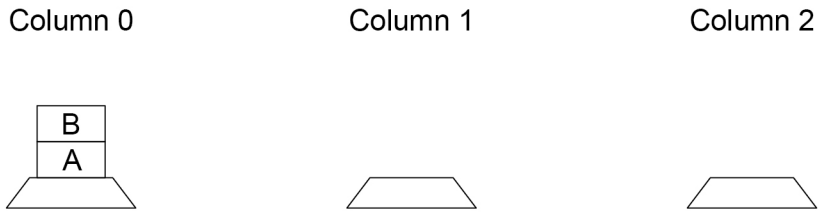
3 3 . 3 Develop an algorithm using either pseudo-code or a flowchart that will move every block from column 0 to column 1.

Your algorithm should work however many blocks start in column 0. You may assume there will always be at least one block in column 0 at the start and that the other columns are empty.

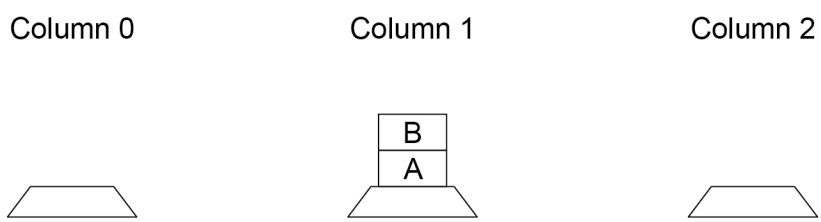
The order of the blocks must be preserved.

The `MOVE` subroutine must be used to move a block from one column to another. You should also use the `HEIGHT` subroutine in your answer.

For example, if the starting arrangement of the blocks is:



Then the final arrangement should have block B above block A:



[4 marks]

3 4 A programmer has written the C# program in **Figure 5** to add up the numbers between one and five.

Figure 5

```
int total = 0;
for (int number = 1; number < 6; number++)
{
  total = total + number;
}
Console.WriteLine(total);
```

The program needs to be changed so that it also multiplies all of the numbers between one and five.

Shade **one** lozenge next to the program that will do what the programmer wants.

[1 mark]

A

```
int total = 0;
int product = 1;
for (int number = 1; number < 6; number++)
{
  total = total + number;
  product = total * number;
}
Console.WriteLine(total);
Console.WriteLine(product);
```

B

```
int total = 0;
int product = 1;
for (int number = 1; number < 6; number++)
{
  total = total + number;
  product = product * number;
}
Console.WriteLine(total);
Console.WriteLine(product);
```

C

```
int total = 0;
int product = 1;
for (int number = 1; number < 6; number++)
{
  total = total + number;
  product = product * total;
}
Console.WriteLine(total);
Console.WriteLine(product);
```

D

```
int total = 0;
int product = 1;
for (int number = 1; number < 6; number++)
{
  total = total + number;
  product = (total + product) * number;
}
Console.WriteLine(total);
Console.WriteLine(product);
```

5 A program has been written in C# to display all the odd integers between 1 and the largest odd number smaller than an integer entered by the user. The program is shown in **Figure 6**.

Figure 6

```
int odd = 1;
int number;
Console.Write("Enter an integer: ");
number = Convert.ToInt32(Console.ReadLine());
while (odd != Number)
{
  Console.WriteLine(odd);
  odd = odd + 2;
}
Console.WriteLine("Finished!");
```

The program works correctly if the integer entered by the user is an odd, positive integer. For example, if 7 is entered the program correctly displays the values 1, 3 and 5

The program does not work correctly if an odd integer less than 1 is entered by the user. For example, when −7 is entered the program should display the values 1, -1, -3 and -5 but it doesn't do this.

Using C# only, change the program code inside the while loop so that it will work correctly for any odd integer entered by the user.

[4 marks]

3 6 **Figure 1** shows an algorithm, represented using pseudo-code.

The algorithm assigns different values to two variables, then asks the user to input a letter.

Figure 1

```
film ← "Godzilla vs. Kong"
year ← 2021
OUTPUT "Please guess a letter"
letter ← USERINPUT
```

3 6 . 1 Which pseudo-code statement assigns the length of the string `film` to a variable called `value`?

Shade **one** lozenge.

[1 mark]

A `film ← LEN(value)` ⬭

B `film ← film + value` ⬭

C `value ← film` ⬭

D `value ← LEN(film)` ⬭

3 6 . 2 The `POSITION` subroutine returns the position of the first occurrence of a character in a string.

For example:

- `POSITION("Godzilla vs. Kong", "o")` would return `1`
- `POSITION("Godzilla vs. Kong", "z")` would return `3`

`letter` and `film` are variables used in the algorithm in **Figure 1**.

Complete the pseudo-code statement to find the position of the first occurrence of the contents of `letter` in `film` and store this position in the variable `location`

You **must** use the `POSITION` subroutine in your answer.

[1 mark]

`location ←` _____

_____

3 6 . 3 Which of the following would be the most suitable data type for the variable `year`?

Shade **one** lozenge.

[1 mark]

A Boolean ⬭

B character ⬭

C integer ⬭

D real ⬭

3 6 . 4 Describe what is meant by an assignment statement in a program.

[1 mark]

_____

_____

3 6 . 5 Write a C# program that:

- gets the user to enter the name of a film
- displays `You entered` followed by the name of the film entered by the user.

The output from the program **must** be on one line.

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.

[2 marks]

3 7 **Figure 2** shows an algorithm, represented using pseudo-code.

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

Figure 2

```
1     num ← USERINPUT
2     IF NOT(num > 1) OR num > 20 THEN
3        OUTPUT "False"
4     ELSEIF num > 1 AND num < 15 THEN
5        OUTPUT "Almost"
6     ELSEIF num MOD 5 = 0 THEN
7        OUTPUT "True"
8     ELSE
9        OUTPUT "Unknown"
10    ENDIF
```

The modulus operator is used to calculate the remainder after dividing one integer by another.

For example:
- `14 MOD 3` evaluates to `2`
- `24 MOD 5` evaluates to `4`

3 7 . 1 Where is a relational operator **first** used in the algorithm in **Figure 2**?

Shade **one** lozenge.

[1 mark]

A Line number 1 ⬭

B Line number 2 ⬭

C Line number 3 ⬭

D Line number 6 ⬭

3 7 . 2 In the algorithm in **Figure 2**, what will be the output when the user input is `5`?

Shade **one** lozenge.

[1 mark]

A `Almost` ⬭

B `False` ⬭

C `True` ⬭

D `Unknown` ⬭

3 7 . 3 Which value input by the user would result in `True` being output by the algorithm in **Figure 2**?

Shade **one** lozenge.

[1 mark]

A `-1` ⬭

B `10` ⬭

C `20` ⬭

D `21` ⬭

3 7 . 4 Rewrite **line 2** from the algorithm in **Figure 2 without** using the `NOT` operator.

The algorithm must still have the same functionality.

[1 mark]

3 7 . 5 A user inputs a value into the algorithm in **Figure 2**.

State **one** value that the user could input that would result in an output of `Unknown`

[1 mark]

3 8 A university is writing a program to calculate a student's total mark for three essays.

If any essays are handed in late, the total mark is reduced.

Write a C# program to calculate the total mark.

You should assume there are three integer variables called `e1`, `e2` and `e3` which have already been given values to represent the marks of the three essays.

The program should:

- get the user to enter the number of essays handed in late and store the number in a variable
- calculate the total mark for the three essays
  - if only one essay is handed in late, the total mark is reduced by 10
  - if more than one essay is handed in late, the total mark should be halved
  - the total mark should **not** be less than 0
- output the total mark.

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**.**

[7 marks]

Turn over for the next question

3 9 A shop owner wants to create stock codes for each type of sweet they sell.

Figure 5 shows some of the sweets.

Figure 5

sweetID	sweetName	brand
S1	WINE GUMS	MAYNARDS
S2	COLA CUBES	BERRYMANS
S3	STARBURST	WRIGLEY

A stock code is made up of the:
- `sweetID`
- first letter and the second letter in `sweetName`
- first letter of the `brand`

For example:
- the stock code for `WINE GUMS` would be `S1WIM`
- the stock code for `STARBURST` would be `S3STW`

Write a C# program to create the stock code for a sweet.

The program should:
- get the user to enter the `sweetID`, `sweetName` and `brand`
- create the stock code
- assign the stock code to a variable called `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.

[4 marks]

Turn over for the next question

4 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 ⬭

4 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)
```

4 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

4 1 **Figure 8** shows a C# program.

Figure 8

```
static void First(int p1, int p2, int p3)
{
   int v1 = p2 + p3;
   Console.WriteLine(Second(v1, p1));
}

static int Second(int p1, int p2)
{
   int v1 = p1 + p2;
   if (v1 > 12)
   {
      v1 = v1 + Third(p1);
   }
   return v1;
}

static int Third(int p1)
{
   if (p1 > 3)
   {
      return 2;
   }
   else
   {
      return 0;
   }
}
```

4 1 . 1 State what will be displayed by the `Console.WriteLine` statement when the subroutine `First` is called with the values `3`, `4` and `4` for the parameters `p1`, `p2` and `p3`

[1 mark]

4 1 . 2 State what will be displayed by the `Console.WriteLine` statement when the subroutine `First` is called with the values `3`, `4` and `8` for the parameters `p1`, `p2` and `p3`

[1 mark]

4 2 A program is to be written to authenticate a username and password entered by the user.

Figure 9 shows the only two pairs of valid usernames and passwords.

Figure 9

Username	Password
Yusuf5	33kk
Mary80	af5r

Write a C# program to authenticate a username and password.

The program should:
- get the user to enter a username
- get the user to enter a password
- display the message `Access denied` if the username and password pair entered is not valid
- display the message `Access granted` if the username and password pair entered is valid
- repeat until a valid username and password pair is entered.

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.

[7 marks]

4 3 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

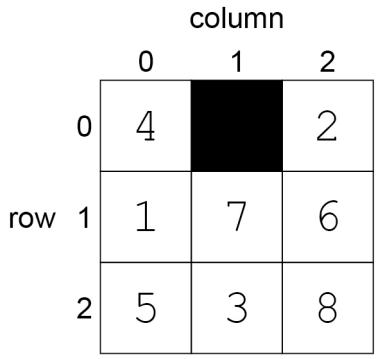


Figure 11

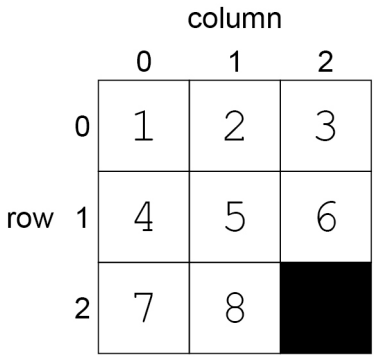


Table 3 describes the purpose of three subroutines the program uses.

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

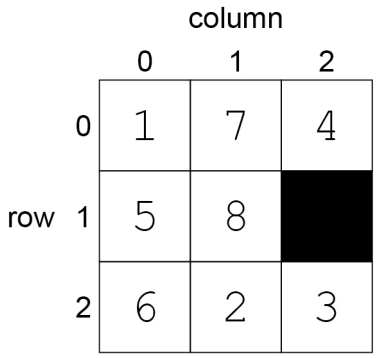
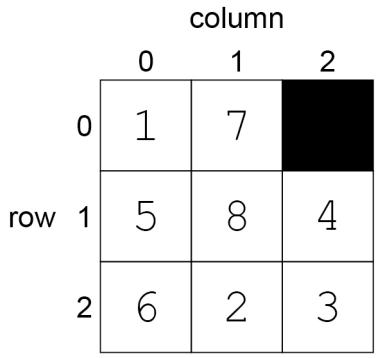


Figure 13



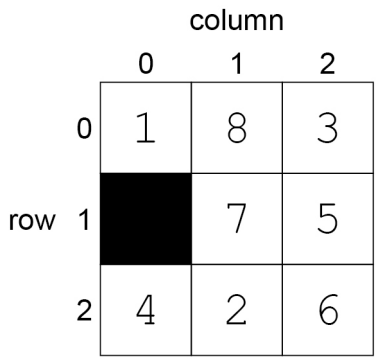
4 3 . 1 The C# program shown in **Figure 14** uses the subroutines in **Table 3**, on page 25.

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]

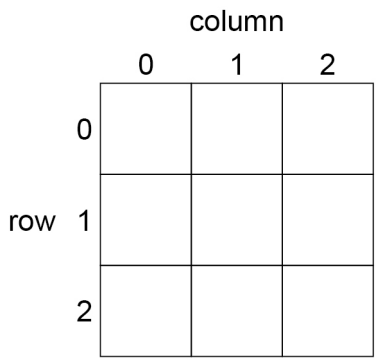


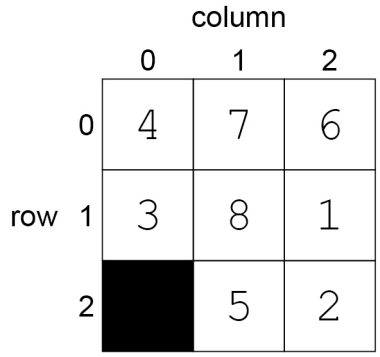
Figure 16 shows part of a C# program that uses the `getTile` subroutine from **Table 3**, on page 25.

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



4 3 . 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. ⬭

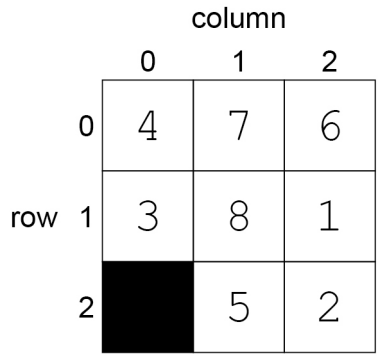
E The values of `i` and `j` do not change when the program is executed. ⬭

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



4 3 . 3 Explain the purpose of the **first** iteration structure in the program in **Figure 16**.

[1 mark]

4 3 . 4 Explain the purpose of the **second** iteration structure in the program in **Figure 16**.

[1 mark]

4 3 . 5 State the purpose of the program in **Figure 16**.

[1 mark]

4 3 . 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

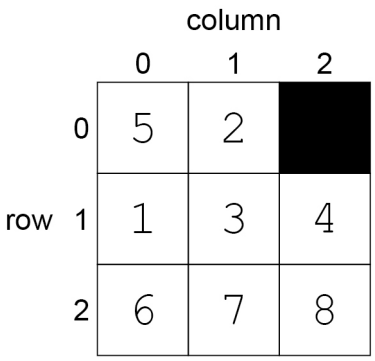
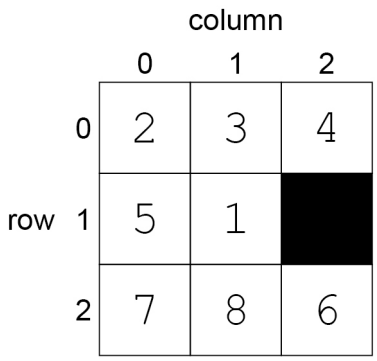


Figure 19



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]

4 3 . 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]

4 4 A programmer is writing a game.

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

Figure 20

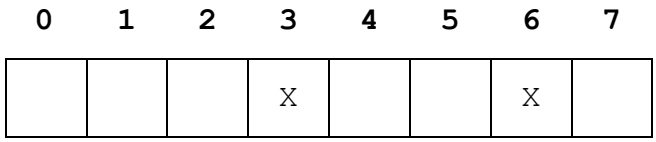


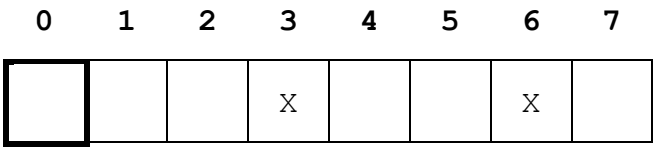
Figure 21 describes how the game is to be played.

Figure 21

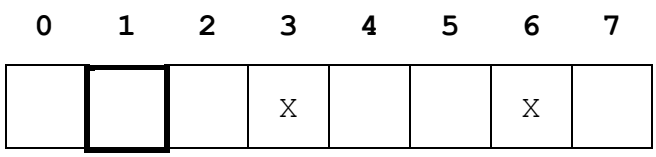
- The player starts at position `0` in a row of cells.
- The aim of the game is for the player to reach the end of the row.
- At each turn the player must enter either `1` or `2`
  - if the player enters `1`, the player's position increases by 1
  - if the player enters `2`, the player's position increases 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**:

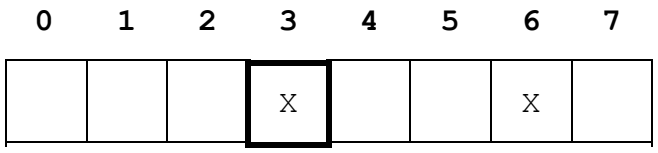
- the player starts in position `0`



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

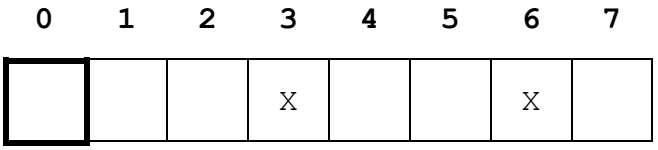


- if the player then enters a `2`, `Bad Move` is displayed as position `3` contains an `X`

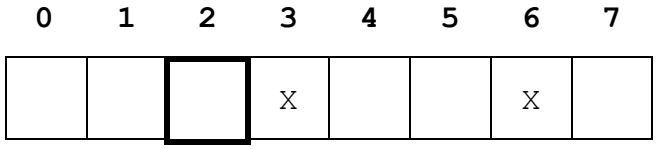



```
Bad move
```

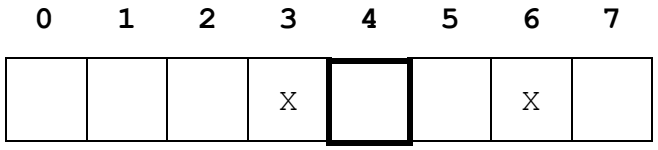
- the player then goes back to position `0`



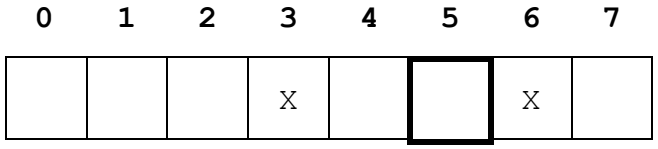
- if the player then enters a `2`, they move to position `2`



- 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.

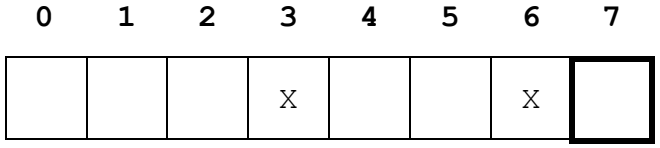


Figure 22 shows part of a C# program that will be used for the game.

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)
{
```

}				