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 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 3 The subroutine in **Figure 3** is used to authenticate a username and password combination.

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

Figure 3

```
1     SUBROUTINE Authenticate(user, pass)
2        us ← ['dave', 'alice', 'bob']
3        ps ← ['abf32', 'woof2006', '!@34E$']
4        z ← 0
5        correct ← false
6        WHILE z < 3
7           IF user = us[z] THEN
8              IF pass = ps[z] THEN
9                 correct ← true
10             ENDIF
11          ENDIF
12          z ← z + 1
13       ENDWHILE
14       RETURN correct
15    ENDSUBROUTINE
```

0 3 . 1 Complete the trace table for the following subroutine call:

```
Authenticate('alice', 'woof2006')
```

[3 marks]

z	correct

0 3 . 2 State the value that is returned by the following subroutine call:

```
Authenticate('bob', 'abf32')
```

[1 mark]

0 3 . 3 Lines 7 and 8 in **Figure 3** could be replaced with a single line. Shade **one** lozenge to show which of the following corresponds to the correct new line.

[1 mark]

A `IF user = us[z] OR pass = ps[z] THEN` ⬭

B `IF user = us[z] AND pass = ps[z] THEN` ⬭

C `IF NOT (user = us[z] AND pass = ps[z]) THEN` ⬭

0 3 . 4 A programmer implements the subroutine shown in **Figure 3**. He replaces line 9 with

```
RETURN true
```

He also replaces line 14 with

```
RETURN false
```

Explain how the programmer has made the subroutine more efficient.

[2 marks]

0 4 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 4 . 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 4 . 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 4 . 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 5 Run length encoding (RLE) is a form of compression that creates frequency/data pairs to describe the original data.

For example, an RLE of the bit pattern `00000011101111` could be `6 0 3 1 1 0 4 1` because there are six `0`s followed by three `1`s followed by one `0` and finally four `1`s.

The algorithm in **Figure 7** is designed to output an RLE for a bit pattern that has been entered by the user.

Five parts of the code labelled **L1**, **L2**, **L3**, **L4** and **L5** are missing.

- Note that indexing starts at zero.

Figure 7

```
pattern ← L1
i ← L2
count ← 1
WHILE i < LEN(pattern)-1
   IF pattern[i] L3 pattern[i+1] THEN
      count ← count + 1
   ELSE
      L4
      OUTPUT pattern[i]
      count ← 1
   ENDIF
   L5
ENDWHILE
OUTPUT count
OUTPUT pattern[i]
```

0 5 . 1 Shade **one** lozenge to show what code should be written at point **L1** of the algorithm.

[1 mark]

A `OUTPUT` ⬭

B `'RLE'` ⬭

C `True` ⬭

D `USERINPUT` ⬭

0 5 . 2 Shade **one** lozenge to show what value should be written at point **L2** of the algorithm.

[1 mark]

A -1 ⬭

B 0 ⬭

C 1 ⬭

D 2 ⬭

0 5 . 3 Shade **one** lozenge to show what operator should be written at point **L3** of the algorithm.

[1 mark]

A = ⬭

B ≤ ⬭

C < ⬭

D ≠ ⬭

0 5 . 4 Shade **one** lozenge to show what code should be written at point **L4** of the algorithm.

[1 mark]

A `count` ⬭

B `count ← count - 1` ⬭

C `count ← USERINPUT` ⬭

D `OUTPUT count` ⬭

0 5 . 5 Shade **one** lozenge to show what code should be written at point **L5** of the algorithm.

[1 mark]

A `i ← i * 2` ⬭

B `i ← i + 1` ⬭

C `i ← i + 2` ⬭

D `i ← i DIV 2` ⬭

0 5 . 6 State a run length encoding of the series of characters `ttjjeeess`

[2 marks]

0 5 . 7 A developer implements the algorithm shown in **Figure 7** and tests their code to check that it is working correctly. The developer tests it only with the input bit pattern that consists of six zeros and it correctly outputs `6 0`.

Using example test data, state **three** further tests that the developer could use to improve the testing of their code.

[3 marks]

0 6 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 6 . 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 6 . 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 6 . 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 6 . 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 6 . 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 7 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]

0 8 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]

0 9 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 9 . 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 9 . 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 9 . 3 What type of data structure is the variable `numbers`?

[1 mark]

_____

Turn over for the next question

1 0 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 1 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 1 . 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 1 . 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.

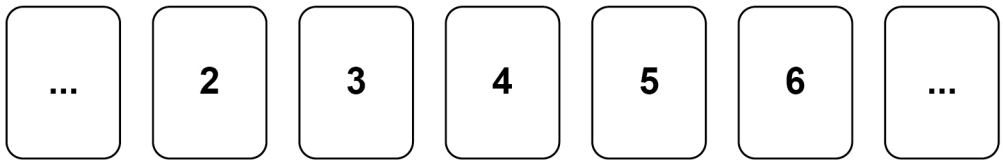
The player's 100 cards are always stored in numerical order.

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

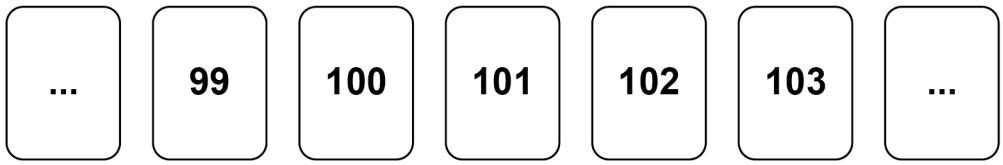
A valid run:
- consists of five cards
- can start from any position in the player's 100 cards
- the second card's value is one more than the first card's value, the third card's value is one more than the second card's value, the fourth card's value is one more than the third card's value, and the fifth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

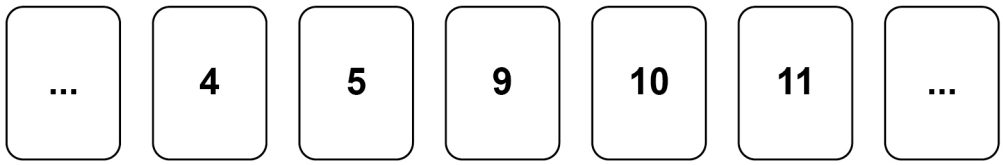


Valid run example 2

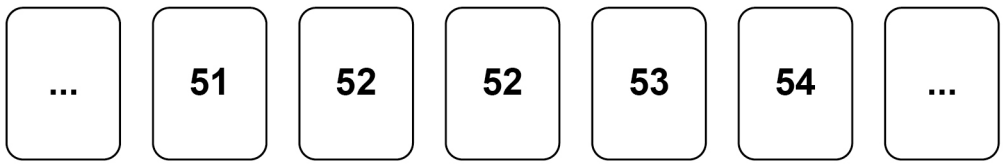


Below are examples of invalid runs.

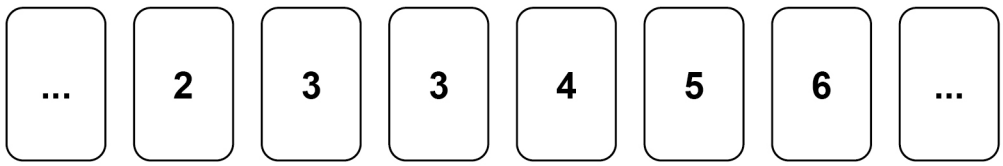
Invalid run example 1



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 2 **Figure 3** shows a program written in C# that calculates the area of a rectangle or the volume of a box from the user inputs.

Figure 3

```
public static int calculate(int width, int length,
int height) {
   if (height == -1)
   {
      return width * length;
   } else
   {
      return width * length * height;
   }
}

public static void Main() {
   int numOne, numTwo, numThree, answer;
   Console.Write("Enter width: ");
   numOne = Convert.ToInt32(Console.ReadLine());
   Console.Write("Enter length: ");
   numTwo = Convert.ToInt32(Console.ReadLine());
   Console.Write("Enter height, -1 to ignore: ");
   numThree = Convert.ToInt32(Console.ReadLine());

   answer = calculate(numOne, numTwo, numThree);

   if (numThree == -1)
   {
      Console.WriteLine($"Area {answer}");
   } else
   {
      Console.WriteLine($"Volume {answer}");
   }
}
```

1 2 . 1 Complete the trace table using the program in **Figure 3**.

[3 marks

numOne	numTwo	numThree	Final output
5	6	−1	
10	4	0	
3	5	10	

1 2 . 2 Describe **one** way that the program in **Figure 3** could be made more robust.

[1 mark]

Turn over for the next question

1 3 **Figure 5** shows an algorithm represented using pseudo-code.

The algorithm is for a simple authentication routine.

The pseudo-code uses a subroutine `getPassword` to check a username:

- If the username exists, the subroutine returns the password stored for that user.
- If the username does not exist, the subroutine returns an empty string.

Parts of the algorithm are missing and have been replaced with the labels L1 to L4.

Figure 5

```
login ← False
REPEAT
   username ← ''
   WHILE username = ''
      OUTPUT 'Enter username: '
      username ← L1
   ENDWHILE
   password ← ''
   WHILE password = ''
      OUTPUT 'Enter password: '
      password ← USERINPUT
   ENDWHILE
   storedPassword ← getPassword( L2 )
   IF storedPassword =  L3  THEN
      OUTPUT ' L4 '
   ELSE
      IF password = storedPassword THEN
         login ← True
      ELSE
         OUTPUT 'Try again.'
      ENDIF
   ENDIF
UNTIL login = True
OUTPUT 'You are now logged in.'
```

Figure 6

-1	OUTPUT	0
username	True	SUBROUTINE
1	User not found	''
USERINPUT	password	Wrong password

State the items from **Figure 6** that should be written in place of the labels in the algorithm in **Figure 5**.

You will not need to use all the items in **Figure 6**.

[4 marks]

L1 _____

L2 _____

L3 _____

L4 _____

Turn over for the next question

1 4 **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
```

1 4 . 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

1 4 . 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

1 5 **Figure 10** shows part of an algorithm that has been written in pseudo-code.

There is an error in the algorithm.

The algorithm should:
- get the start year and end year from the user
- check that the start year is before the end year
- check that the start year is before 2000
- calculate the difference between the two years after a valid start year has been entered.

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

Figure 10

```
1   validChoice ← False
2   REPEAT
3      difference ← –1
4      OUTPUT 'Enter a start year '
5      startYear ← USERINPUT
6      OUTPUT 'Enter an end year '
7      endYear ← USERINPUT
8      IF startYear ≥ endYear THEN
9         OUTPUT 'Start year must be before end year'
10     ELSE
11        IF startYear < 2000 THEN
12           OUTPUT 'Start year must be before 2000'
13        ELSE
14           validChoice ← True
15        ENDIF
16     ENDIF
17  UNTIL validChoice = True
18  difference ← endYear - startYear
19  OUTPUT difference
```

1 5 . 1 **Table 1** shows three tests used to check the algorithm in **Figure 10**.

Complete the table to show what the values of the `validChoice` and `difference` variables would be for the given test data.

[4 marks]

Table 1

Test type	**Test data**		`validChoice`	`difference`
Normal	`startYear`	`1995`		
	`endYear`	`2010`		
Erroneous	`startYear`	`2015`		
	`endYear`	`2000`		
Boundary	`startYear`	`2000`		
	`endYear`	`2023`		

1 5 . 2 The algorithm in **Figure 10** contains a logic error on **line 11**.

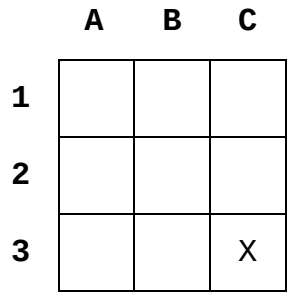
Describe how the error on **line 11** can be corrected.

[1 mark]

Turn over for the next question

1 6 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();
    }
}
```

1 7 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]

1 8 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]

9 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]

2 0 **Figure 7** shows part of a program written in C#.

Figure 7

```
bool validChoice;
int choice;
validChoice = false;
while (validChoice == false)
{
   Console.Write("Enter your choice [1 - 10] ");
   choice = int.Parse(Console.ReadLine());
   if (choice >= 1 & choice <= 10)
   {
      validChoice = true;
   }
   else
   {
      Console.WriteLine("Invalid choice");
   }
}
Console.WriteLine("Valid choice");
```

Complete the following test plan for the code shown in **Figure 7**.

Test type	Test data	Expected result
Normal data	5	`Valid choice` message displayed
Invalid data		
Boundary data		

[2 marks]

2 | 1 **Figure 8** shows a C# program that is being developed.

It is supposed to calculate and display the highest common factor of two numbers entered by the user.

The highest common factor of two numbers is the largest number that both numbers can be divided by without leaving a remainder.

Examples:

- the highest common factor of the numbers 6 and 9 is 3
- the highest common factor of 2 and 5 is 1

Line numbers are shown but are not part of the program code.

Figure 8

```
1  int num1 = Convert.ToInt32(Console.ReadLine());
2  int num2 = Convert.ToInt32(Console.ReadLine());
3  int hcf = 1;
4  int count = 1;
5  while (count < num1)
6  {
7    if (num1 % count == 0 && num2 % count == 0)
8    {
9      hcf = count;
10   }
11   count = count + 1;
12 }
13 Console.WriteLine(hcf);
```

The highest common factor of two numbers is the largest number that both numbers can be divided by without leaving a remainder.

Examples:

- the highest common factor of the numbers 6 and 9 is 3
- the highest common factor of 2 and 5 is 1

The program in **Figure 8** works correctly sometimes but not always. When the user enters the numbers 4 and 6 it correctly outputs 2, but when the user enters the numbers 4 and 4 it should output 4 but it does not.

2 1 . 1 State the output from the program in **Figure 8** when the user enters the numbers 4 and 4

[1 mark]

2 1 . 2 State the line number from the program in **Figure 8** which contains the error that stops the program from sometimes working correctly.

[1 mark]

2 1 . 3 Describe how the line of code identified in your answer to **17.2** should be changed so that the program in **Figure 8** will work correctly.

[1 mark]

Turn over for the next question

2 2 Write a C# program that calculates an estimate of the braking distance in metres for a new model of go-kart that is travelling between `10` and `50` kilometres per hour (kph).

Your program should:

- keep asking the user to enter a speed for the go-kart until they enter a speed that is between `10` and `50` (inclusive)
- calculate the braking distance in metres by dividing the speed by 5
- ask the user if the ground is wet (expect the user to enter `yes` if it is)
- if the ground is wet, multiply the braking distance by 1.5
- output the final calculated braking distance.

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 you code accurately.

[8 marks]

2 3 **Figure 3** shows an incomplete C# program for a number guessing game.

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

Figure 3

```
1     Random rGen = new Random();
2     int randomNumber;
3
4     Console.WriteLine("Enter a number");
5     int userNumber = Convert.ToInt32(Console.ReadLine());
6     while (userNumber < 1 || userNumber > 100)
7     {
8        Console.WriteLine("Invalid number");
9        userNumber = Convert.ToInt32(Console.ReadLine());
10    }
11    Console.WriteLine("Valid number entered");
12    if (randomNumber == userNumber)
13    {
14       Console.WriteLine("Number guessed correctly");
15    }
```

2 3 . 1 The program should generate a random number between 1 and 100 (including 1 and 100). This will be the number the user has to guess.

Write the C# code that should be used on **line 3** in **Figure 3** to:
- generate a random number between 1 and 100 inclusive
- assign this number to the appropriate variable from the program.

You **must** use `rGen.Next(a, b)` in your C# code.

`rGen.Next(a, b)` generates a random integer in the range `a` to `b` starting at `a` but finishing one before `b`

[2 marks]

2 3 . 2 Complete the test plan in **Table 1** to test the validation of `userNumber` in the program in **Figure 3**.

[2 marks]

Table 1

Test number	Test type	Test data	Expected result
1	Erroneous	`150`	
2	Boundary		
3	Normal		`Valid number entered`

2 3 . 3 In an earlier version of the program in **Figure 3**, **line 6** contained one syntax error and one logic error:

```
whil (userNumber < 1 || userNumber >= 100)
```

Complete the table to describe the errors in the program on **line 6**.

[2 marks]

Error type	Description
Syntax error	
Logic error	

2 4 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]
