**1.** 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`

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

**A** Line number 1

**B** Line number 2

**C** Line number 3

**D** Line number 4

**(1)**

(b) Shade **one** lozenge that shows the output from the algorithm in **Figure 1** when the user input is `4`

**A** `0`

**B** `2`

**C** `4`

**D** `8`

**E** `16`

**(1)**

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

**A** Line number 1

**B** Line number 2

**C** Line number 3

**D** Line number 4

**(1)**

(d) Shade **one** lozenge that shows the line number that contains a relational operator in the algorithm in **Figure 1**.

**A** Line number 1

**B** Line number 2

**C** Line number 3

**D** Line number 4

**(1)**

**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
```

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

**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)**

(f) **Figure 2** shows an implementation of the algorithm in **Figure 1** using the Python programming language.

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

**Figure 2**

```
1      i = int(input("Enter a number: "))
2      if i % 2 == 0:
3        print(i * i)
4      else:
5        print(i)
```

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.

| | | |
|---|---|---|
| **A** | `for x in range(0, 5):`<br>    `i = int(input("Enter a number: "))`<br>    `if i % 2 == 0:`<br>        `print(i * i)`<br>    `else:`<br>        `print(i)` | ○ |
| **B** | `for x in range(0, 6):`<br>    `i = int(input("Enter a number: "))`<br>    `if i % 2 == 0:`<br>        `print(i * i)`<br>    `else:`<br>        `print(i)` | ○ |
| **C** | `x = 1`<br>`while x != 6:`<br>    `i = int(input("Enter a number: "))`<br>    `if i % 2 == 0:`<br>        `print(i * i)`<br>    `else:`<br>        `print(i)`<br>    `x = x + 1` | ○ |
| **D** | `x = 6`<br>`while x != 0:`<br>    `i = int(input("Enter a number: "))`<br>    `if i % 2 == 0:`<br>        `print(i * i)`<br>    `else:`<br>        `print(i)`<br>    `x = x – 1` | ○ |

**(1)**

**(Total 6 marks)**

**2.** The figure below shows a Python program that calculates car park charges.

The user inputs their car registration (eg `MA19 GHJ`) and the length of the stay. The program then outputs the charge.

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

```
1    charge = 0
2    carReg = input("Enter your car registration: ")
3    while len(carReg) > 8:
4       displayMessage = " is not valid"
5       carReg = input(displayMessage)
6    hours = int(input("Enter your stay in hours: "))
7    if hours < 2:
8       charge = 0
9    else:
10      charge = hours * 2
11   print(charge)
```

(a) Rewrite **line 4** in the figure above to **concatenate** the car registration with the string `" is not valid"`, and store the result in the variable `displayMessage`.

Your answer must be written in Python.

**(1)**

(b) The charge for parking for two or more hours is changed to include an additional £2 fee.

Rewrite **line 10** in the figure above to show this change.

Your answer must be written in Python.

**(1)**
**(Total 2 marks)**

**3.** The two Python programs in the figure below output the value that is equivalent to adding together the integers between 1 and an integer entered by the user.

For example, if the user entered the integer `5`, both programs would output `15`

**Program A**

```
print("Enter a number: ")
num = int(input())
total = 0
for i in range(1, num + 1):
   total = total + i
print(total)
```

**Program B**

```
print("Enter a number: ")
num1 = int(input())
num2 = num1 + 1
num2 = num1 * num2
num2 = num2 // 2
print(num2)
```

(a) Shade **one** lozenge to indicate which of the statements is true about the programs in the figure above.

| | | |
|---|---|---|
| **A** | Both programs are equally efficient. | ⬭ |
| **B** | Program A is more efficient than Program B. | ⬭ |
| **C** | Program B is more efficient than Program A. | ⬭ |

**(1)**

(b) Justify your answer for part (a).

**(2)**

**(Total 3 marks)**

**4.** A programmer has started to write a program using Python. Their program is shown in the figure below.

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` module.

For example, `random.randrange(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.

```
1    import random
2    numbers = [ 11, 14, 56, 4, 12, 6, 42, 2 ]
3    count = 0
4    while count < 10:
5       count = count + 1
6       number = random.randrange(0, 8
7       print(numbers[count])
```

(a) The program shown in the figure above contains a syntax error.

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

**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.

**(2)**

(b) The program shown in the figure above 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 Python.

**(2)**

(c) What type of data structure is the variable `numbers`?

**(1)**

**(Total 5 marks)**

**5.** Write a Python 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 indentation as appropriate, meaningful variable name(s) and Python syntax in your answer.

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

**(Total 5 marks)**

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**6.** Write a Python 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 indentation as appropriate, meaningful variable name(s) and Python syntax in your answer.

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

**(Total 7 marks)**

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**7.** A program is being developed in Python 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.

(a) The figure below shows part of the program that will get a player to enter the position of a card to swap.

```
position = int(input("Enter card position: "))
```

Extend the program in the figure above. Your answer must be written in Python.

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 indentation as appropriate, meaningful variable name(s) and Python syntax in your answer.

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

**(4)**

|  |  |  |  |  |
|---|---|---|---|---|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

(b) 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.

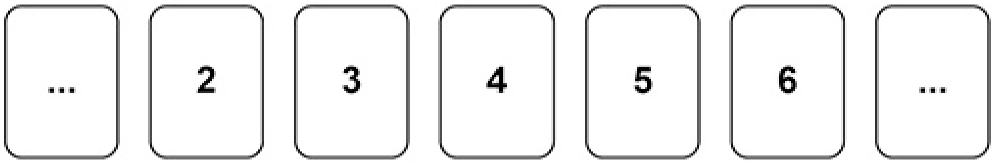
When a player has a valid run of five cards within their 100 cards they have won the game.

A valid run:

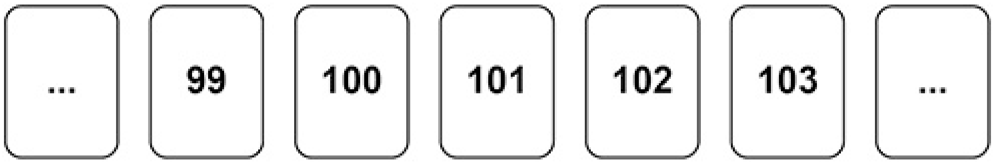
- consists of five cards
- can start from any position in the player's 100 cards
- the second card's value is one more than the first card's value, the third card's value is one more than the second card's value, the fourth card's value is one more than the third card's value, and the fifth card's value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

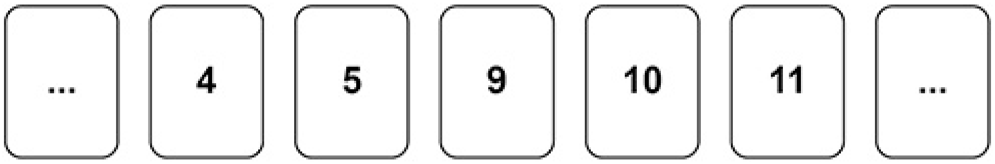


**Valid run example 2**

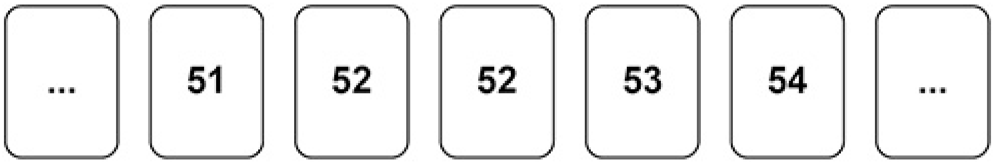


Below are examples of invalid runs.

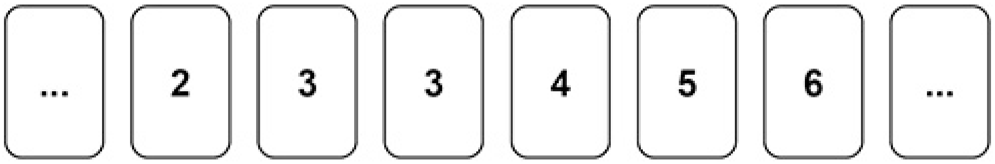
**Invalid run example 1**



**Invalid run example 2**



**Invalid run example 3**



Write a Python 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 indentation as appropriate, meaningful variable name(s) and Python syntax in your answer.

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

**(6)**

**(Total 10 marks)**

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**8.** A programmer has written a Python program that asks the user to input two integers and then output which of the two integers is the largest.
Complete the program below by filling in the gaps using the items in the box. You will not need to use all the items. Each item should only be used once.

| | | | |
|---|---|---|---|
| `print` | `num1` | `num2` | `output` |
| `else` | `<` | `>` | `elif` |
| `str` | `float` | `int` | |

```
num1 = int(input("Enter a number: "))

num2 = ____________________ (input("Enter a second number: "))

if num1 > num2:

     print(" ____________________ is bigger.")

elif num1 ____________________ num2:

     print(" ____________________ is bigger.")

____________________

     print("The numbers are equal.")
```

**(Total 5 marks)**

**9.** Write a Python 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.

**(Total 7 marks)**

|  |  |  |  |  |
|---|---|---|---|---|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**10.** Write a Python 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.

**(Total 5 marks)**

|  |  |  |  |  |
|---|---|---|---|---|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**11.** Write a Python 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.

**(Total 7 marks)**

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**12.** A programmer has written the Python program shown below to add up the numbers between one and five.

```
total = 0
for number in range(1, 6):
   total = total + number
print(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.

| | | |
|---|---|---|
| **A** | `total = 0`<br>`product = 1`<br>`for number in range(1, 6):`<br>`   total = total + number`<br>`   product = total * number`<br>`print(total)`<br>`print(product)` | ○ |
| **B** | `total = 0`<br>`product = 1`<br>`for number in range(1, 6):`<br>`   total = total + number`<br>`   product = product * number`<br>`print(total)`<br>`print(product)` | ○ |
| **C** | `total = 0`<br>`product = 1`<br>`for number in range(1, 6):`<br>`   total = total + number`<br>`   product = product * total`<br>`print(total)`<br>`print(product)` | ○ |
| **D** | `total = 0`<br>`product = 1`<br>`for number in range(1, 6):`<br>`   total = total + number`<br>`   product = (total + product) * number`<br>`print(total)`<br>`print(product)` | ○ |

**(Total 1 mark)**

**13.** A program has been written in Python 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 below.

```
odd = 1
number = int(input("Enter an integer: "))
while odd != number:
    print(odd)
    odd = odd + 2
print("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 does not do this.

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

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

**(Total 4 marks)**

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

**14.** Part of a program written in Python is shown below.

```
validChoice = False
while validChoice == False:
   choice = int(input('Enter your choice [1 - 10]'))
   if choice >= 1 and choice <= 10:
      validChoice = True
   else:
      print('Invalid choice')
print('Valid choice')
```

Complete the following test plan for the code shown above.

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

**(Total 2 marks)**

**15.** The code below shows a Python 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.

```
1   num1 = int(input())
2   num2 = int(input())
3   hcf = 1
4   count = 1
5   while count < num1:
6      if (num1 % count == 0 and num2 % count == 0):
7         hcf = count
8      count = count + 1
9   print(hcf)
```

The program 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.

(a) State the output from the program when the user enters the numbers 4 and 4

**(1)**

(b) State the line number from the program which contains the error that stops the program from sometimes working correctly.

**(1)**

(c) Describe how the line of code identified in your answer to (b) should be changed so that the program will work correctly.

**(1)**

**(Total 3 marks)**

**16.** Write a Python 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 your code accurately.

**(Total 8 marks)**

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

## Mark schemes

**1.** (a) **Mark is for AO2 (apply)**

**B** Line number 2;

**R.** If more than one lozenge shaded

**1**

(b) **Mark is for AO2 (apply)**

**E** `16;`

**R.** If more than one lozenge shaded

**1**

(c) **Mark is for AO2 (apply)**

**A** Line number 1;

**R.** If more than one lozenge shaded

**1**

(d) **Mark is for AO2 (apply)**

**B** Line number 2;

**R.** If more than one lozenge shaded

**1**

(e) **Mark is for AO2 (apply)**

**D** This algorithm uses the multiplication operator;

**R.** If more than one lozenge shaded

**1**

(f) **Mark is for AO3 (refine)**

**C#**
**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);
      }
  }
```

**Python**

**A**

```
   for x in range(0, 5):
      i = int(input("Enter a number: "))
      if i % 2 == 0:
        print(i * i)
      else:
        print(i)
```

**VB.NET**

**C**

```
   For x As Integer = 0 To 4
      Console.Write("Enter a number: ")
      Dim i As Integer = Console.ReadLine()
      If i Mod 2 = 0 Then
        Console.WriteLine(i * i);
      Else {
        Console.WriteLine(i)
      End If
   Next
```

**R.** If more than one lozenge shaded

1

**[6]**

**2.** (a) **Mark is for AO3 (refine)**

**C#**

```
string displayMessage = carReg + " is not valid";
```

**Python**

```
displayMessage = carReg + " is not valid"
```

**VB.NET**

```
Dim displayMessage As String = carReg + " is not valid" //
Dim displayMessage As String = carReg & " is not valid"
```

**I.** Case

**I**. Space between variable outputs

**I**. Order of strings

1

(b) **Mark is for AO3 (refine)**

**C#**
```
charge = hours * 2 + 2; //
charge = 2 + hours * 2;
```

**Python**
```
charge = hours * 2 + 2 //
charge = 2 + hours * 2
```

**VB.NET**
```
charge = hours * 2 + 2 //
charge = 2 + hours * 2
```

**I.** Case
**I.** Parentheses, unless altering result eg, `hours * (2 + 2)`

1

**[2]**

**3.** (a) **Mark is for AO2 (apply)**

**C** Program B is more efficient than Program A;

**R.** If more than one lozenge shaded

1

(b) **2 marks for AO2 (apply)**

It will take less time for the computer to execute program B;
because fewer lines of code will be executed;
//
The number of calculations performed is constant in Program B;
but increases as the number input gets bigger in Program A;

**A.** Program B has fewer variables; so would use less memory (when executing);

2

**[3]**

**4.** (a) **2 marks for AO1 (recall)**

**B** A syntax error is a mistake in the grammar of the code;

**D** A syntax error will stop a program from running;

**R.** If more than two lozenges shaded

2

(b) **Mark is for AO2 (apply)**
**Mark is for AO3 (refine)**

**C#**
`Line number: 7;`

Corrected line of code: `Console.WriteLine(numbers[number]);`

**Python**
`Line number: 7;`

Corrected line of code: `print(numbers[number])`

**VB.NET**
`Line number: 7;`

Corrected line of code: `Console.WriteLine(numbers(number))`

**A**. `WriteLine` changed to `Write` as long as all other required changes have been made

2

(c) **Mark is for AO2 (apply)**

Array // List (of integers);

1

**[5]**

**5.** **2 marks for AO3 (design), 3 marks for AO3 (program)**

**Program Design**
**Note** that AO3 (design) marks are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

**Mark A** for using meaningful variable names throughout and for using two variables to store the two email address inputs;
**Mark B** for the use of a selection construct with ELSE / ELSEIF // use of multiple selection constructs;

**Program Logic**
**Mark C** for using user input and storing the results in a variable correctly for the first email address and the second email address;
**Mark D** for a correct expression that checks if the first entered email address is equal to the second entered email address (or not equal to);
**Mark E** for outputting `Do not match` and `Match` in logically separate places such as the IF and ELSE part of selection, and for outputting the email address if both email addresses match;

**A.** Any suitable alternative messages.

**I**. Case
**I**. Messages or no messages with input statements

**Maximum 4 marks** if any errors in code.

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

```
string email1 = Console.ReadLine();                        (Part of C)

string email2 = Console.ReadLine();                        (Part of C)

if (email1 != email2) {                                    (D)

   Console.WriteLine("Do not match")                       (Part of E)
}
else {
   Console.WriteLine("Match");                             (Part of E)
   Console.WriteLine(email1);                              (Part of E)
}
```

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

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

if (em1 == em2)     {                                      (D)

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

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

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

if email1 == email2:                                      (D)

    print("Do not match")                                 (Part of E)

else:

    print("Match")                                        (Part of E)
    print(email1)                                         (Part of E)
```

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

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

if em1 == em2:                                            (D)

    print("Match")                                        (Part of E)
    print(em2)                                            (Part of E)

else:

    print("Do not match")                                 (Part of E)
```

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

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

if email1 == email2:                                      (D)

    print("Match")
```

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

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

If email1 <> email2 Then                                         (D)

   Console.WriteLine("Do not match")                             (Part of E)

Else

   Console.WriteLine("Match")                                    (Part of E)
   Console.WriteLine(email1)                                     (Part of E)

End If
```

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

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

If em1 = em2 Then                                                (D)

   Console.WriteLine("Match")                                    (Part of E)
   Console.WriteLine(em2)                                        (Part of E)

Else

   Console.WriteLine("Do not match")                             (Part of E)

End If
```

**[5]**

**6.** **3 marks for AO3 (design) and 4 marks for AO3 (program)**

**Program Design**

**Note** that AO3 (design) marks are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

**Mark A** for using meaningful variable names throughout;
**Mark B** for the use of a selection construct;
**Mark C** for the use of a nested selection construct or multiple conditions;

**Program Logic**

**Mark D** for using user input and storing the result in a variable correctly for the items sold **and** years of employment;
**Mark E** for correct expression that checks the years entered against the criteria for years employed;
**Mark F** for correct Boolean expressions throughout;
**Mark G** for outputting correct bonus depending on inputs entered in logically separate places such as IF, ELSE part of selection;

**I.** Case
**I.** Prompts

**Maximum 6 marks** if any errors in code

**C# Example 1 (fully correct)**

```
Console.Write("How many items?: ");
int items = Convert.ToInt32(Console.ReadLine());          (Part of A, D)
Console.Write("How many years employed?: ");
int years = Convert.ToInt32(Console.ReadLine());          (Part of A, D)
if (years <= 2)   {                                       (Part of B, E)
    if items > 100   {                                    (Part of C, F)
      Console.WriteLine(items * 2);                       (Part of G)
    }
    else {                                                (Part of B, E)
      Console.WriteLine(0);                               (Part of G)
    }
}
else {                                                    (Part of B, E)
      Console.WriteLine(items * 10);                      (Part of G)
}
```

**<u>Python Example 1 (fully correct)</u>**

```
items = int(input("How many items?: "))                  (Part of A, D)
years = int(input("How many years employed?: "))         (Part of A, D)
if years <= 2:                                           (Part of B, E)
   if items > 100:                                       (Part of C, F)
      print(items * 2)                                   (Part of G)
   else:                                                 (Part of C, F)
      print(0)                                           (Part of G)
else:                                                    (Part of B, E)
   print(items * 10)                                     (Part of G)
```

**<u>Python Example 2 (fully correct)</u>**

```
items = int(input("Enter items: "))                      (Part of A, D)
years = int(input("Enter years employed: "))             (Part of A, D)
if years <= 2: and items > 100:                          (Part of B, C, E, F)
   print(items * 2)                                      (Part of G)
elif years > 2:                                          (Part of B, C, E, F)
   print(items * 10)                                     (Part of G)
else:                                                    (Part of B, E)
   print(0)                                              (Part of G)
```

**<u>VB.NET Example 1 (fully correct)</u>**

```
Console.Write("Enter items: ")
Dim items As Integer = Console.ReadLine()                (Part of A, D)
Console.Write("Enter years: ")
Dim years As Integer = Console.ReadLine()                (Part of A and D)
If years <= 2 And items > 100 Then                       (Part of B, C, E, F)
   Console.WriteLine(items * 2)                          (Part of G)
ElseIf years > 2 Then                                    (Part of B, C, E, F)
   Console.WriteLine(items * 10)                         (Part of G)
Else                                                     (Part of B, E)
   Console.WriteLine(0)                                  (Part of G)
End If
```

**[7]**

**7.** (a) **2 marks for AO3 (design), 2 marks for AO3 (program)**

<u>**Program Design**</u>
**Note** that AO3 (design) marks are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

**Mark A** for the idea of inputting a number within the iteration/validation structure;
**Mark B** for the use of indefinite iteration;

<u>**Program Logic**</u>
**Mark C** for using a Boolean condition that checks the lower bound of `position`;
**Mark D** for using a Boolean condition that checks **BOTH** the lower and upper bounds of `position` correctly;
**Marks C** and **D** could be one expression eg `0 < position <= 100;`

**I.** Case
**I.** Missing prompts

**Maximum 3 marks** if any errors in code.

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

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

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

```
while (position <= 0 || position >= 101) {                  (C,D)
   Console.Write("Enter card position: ");
   position = Convert.ToInt32(Console.ReadLine());
   }
```

<u>**C# Example 3 (partially correct – 3 marks)**</u>
1 design mark achieved (**Mark A**)

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

**C# Example 4 (partially correct – 3 marks)**

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

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

**4**

**I.** Indentation in C#

**I.** `WriteLine` instead of `Write`

**Python Example 1 (fully correct)**

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

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

**Python Example 2 (fully correct)**

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

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

**Python Example 3 (partially correct – 3 marks)**

1 design mark achieved (Mark A)

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

**Python Example 4 (partially correct – 3 marks)**

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

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

**VB.NET Example 1 (fully correct)**

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

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

**VB.NET Example 2 (fully correct)**

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

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

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

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

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

**<u>VB.NET Example 4 (partially correct – 3 marks)</u>**
All design marks are achieved (**Marks A and B**)

```
Do While position < 1 or position >= 100                    (Mark C)
   Console.Write("Enter card position: ")
   position = Convert.ToInt32(Console.ReadLine())
Loop
```

**I.** Indentation in VB.NET
**I.** `WriteLine` instead of `Write`

(b) **2 marks for AO3 (design), 4 marks for AO3 (program)**
Any solution that does not map to the mark scheme refer to lead examiner

**<u>Program Design</u>**
**Note** that AO3 (design) marks are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

**Mark A** for the idea of using an iteration structure which attempts to access each element in the `cards` array; // attempts to repeat 100 times;
**Mark B** for the idea of using a selection structure which attempts to compare two cards;

**<u>Program Logic</u>**
**Mark C** for using a loop or similar to correctly iterate through the `cards` array using valid indices that do not go out of range;
**Mark D** for using a correct Boolean condition to access the values in the `cards` array;
**Mark E** for correctly checking if there are five values in the `cards` array that are in sequence;
**Mark F** for setting `gameWon` to `True` in the correct place;

**I.** Case

**Maximum 5 marks** if any errors in code.

**<u>C# Example 1 (fully correct)</u>**

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

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

**<u>C# Example 2 (fully correct)</u>**

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

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

**I.** Indentation in C#

**<u>Python Example 1 (fully correct)</u>**

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

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

**<u>Python Example 2 (fully correct)</u>**

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

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

**<u>Python Example 3 (fully correct)</u>**

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

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

**<u>VB.NET Example 1 (fully correct)</u>**

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

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

**VB.NET Example 2 (fully correct)**

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

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

**I.** Indentation in VB.NET

6

**[10]**

**8.** **5 marks for AO3 (program)**

1 mark for each correct item in the correct location.

```
num1 = int(input("Enter a number: "))
num2 =  int          (input("Enter a second number: "))
if num1 > num2:
     print("  num1          is bigger.")
elif num1  <          num2:
     print("  num2          is bigger.")
 else:
     print("The numbers are equal.")
```

**I.** Case of response

**R.** if any spelling mistakes

**[5]**

**9.** **2 marks for AO3 (design) and 5 marks for AO3 (program)**

**Program Design**
**Mark A** for using meaningful variable names throughout (even if logic is incorrect);
**Mark B** for using suitable data types throughout (distance can be real or integer, passengers must be integer);

**Program Logic**
**Mark C** for getting user input for the distance in an appropriate place;
**Mark D** for getting user input for the number of passengers in an appropriate place;
**Mark E** for a fare that correctly charges £2 per passenger;
**Mark F** for a fare that correctly charges £1.50 for every kilometre;
**Mark G** for outputting the correct final fare;

**I.** Case of program code

**Maximum 6 marks** if any errors in code.

**Python Example 1 (fully correct)**
**Mark A** awarded.

```
distance = float(input())             (Part of B, C)
passengers = int(input())             (Part of B, D)
fare = 2 * passengers                 (E)
fare = fare + (1.5 * distance)        (F)
print(fare)                           (G)
```

**Python Example 2 (partially correct – 6 marks)**
**Mark A** awarded. **Mark B** not awarded because float conversion missing.

```
dist = input()                        (C but NOT B)
pass = int(input())                   (Part of B, D)
fare = 2 * pass                       (E)
fare = 1.5 * dist                     (F)
print fare                            (G – still awarded even though
                                      parentheses missing in print
                                      command as logic still clear)
```

**[7]**

**10.** **2 marks for AO3 (design), 3 marks for AO3 (program)**

**Program Design**
**Mark A** for the use of a selection construct (even if the logic is incorrect);
**Mark B** for the correct, consistent use of meaningful variable names throughout (even if the code would not work);

**Program Logic**
**Mark C** for using user input and storing the result in a variable correctly;
**Mark D** for a correct expression that checks if the entered password is `'secret'` (even if the syntax is incorrect);
**Mark E** for outputting `Welcome` and `Not welcome` correctly in logically separate places such as the `IF` and `ELSE` part of selection;

**I.** Case of output strings for **Mark E**, but spelling must be correct
**I.** Case of program code

**Maximum 4 marks** if any errors in code.

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

```
password = input()                     (C)
if password == 'secret':               (D)
   print('Welcome')                    (Part of E)
else:
   print('Not welcome')                (Part of E)
```

**Python Example 2 (partially correct – 4 marks)**
**Mark A** is awarded. **Mark B** is not awarded.

```
p = input()                            (C)
if p == 'secret'                       (D)
   print('Welcome')                    (Part of E)
   else:
   print('Not welcome')                (Part of E)
```

**[5]**

**11.** **3 marks for AO3 (design), 4 marks for AO3 (program)**

**Program Design**
**Mark A** for the idea of inputting a character and checking if it is lower case (even if the code would not work);
**Mark B** for the use of a selection construct (even if the logic is incorrect);
**Mark C** for the correct, consistent use of meaningful variable names throughout (even if the code would not work);

**Program Logic**
**Mark D** for using user input correctly;
**Mark E** for storing the result of user input in a variable correctly;
**Mark F** for a correct expression/method that checks if the character is lowercase;
**Mark G** for outputting `LOWER` and `NOT LOWER` correctly in logically separate places such as the `IF` and `ELSE` part of selection;

**I.** Case of output strings for **Mark G**, but spelling must be correct
**I.** Case of program code

**Maximum 6 marks** if any errors in code.

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

```
character = input()                                    (D, E)
if (character >= 'a') and (character <= 'z'):         (F)
   print('LOWER')                                      (Part of G)
else:
   print('NOT LOWER')                                  (Part of G)
```

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

```
character = input()                                    (D, E)
if character.islower():                                (F)
   print('LOWER')                                      (Part of G)
else:
   print('NOT LOWER')                                  (Part of G)
```

**Python Example 3 (partially correct – 5 marks)**
All design marks are achieved **(Marks A, B and C)**

```
character = input()                                    (D, E)
if (character > 'a') or (character < 'z'):             (NOT F)
   print('NOT LOWER')                                  (NOT G)
else:
   print('LOWER')                                      (NOT G)
```

**[7]**

**12.** **1 mark for AO3 (refine)**

B;

**R.** if more than 1 lozenge shaded

**[1]**

**13.** **4 marks for AO3 (refine)**

**Program Logic**
**Mark A** for using a selection structure with `else` part **or** two selection structures (even if the syntax is incorrect)
**Mark B** for correct condition(s) in selection statement(s) (even if the syntax is incorrect)
**Mark C** for statement that subtracts two from `odd` under the correct conditions (even if the syntax is incorrect)
**Mark D** for `odd` being output and doing one of adding or subtracting two but not both each time loop repeats (even if the syntax is incorrect)

**I.** `while` loop from question if included in answer
**I.** case of program code

**Maximum 3 marks** if any errors in code.

**Python Example 1 (fully correct)**

```
print(odd)                               (Part of D)
if number < 0                            (A, B)
   odd = odd – 2                         (C, Part of D)
else:
   odd = odd + 2                         (Part of D)
```

**Python Example 2 (partially correct – 3 marks)**

```
print(odd)                               (Part of D)
if number != 0                           (A, NOT B)
   odd = odd – 2                         (C, Part of D)
else:
   odd = odd + 2                         (Part of D)
```

**[4]**

**14.** **2 marks for AO3 (test)**

| Test type | Test data | Expected result |
|---|---|---|
| Normal data | 5 | `Valid choice` message displayed |
| Invalid data | Any value other than the numbers 1 to 10 inclusive | `Invalid choice` (message displayed) |
| Boundary data | Any one of 0, 1, 10 or 11 | if 1 or 10 given as test data `Valid choice` (message displayed) if 0 or 11 given as test data `Invalid choice` (message displayed) |

1 mark for each completely correct row to a maximum of 2 marks.

**[2]**

**15.** (a) **1 mark for AO3 (test)**

2;

1

(b) **1 mark for AO3 (test)**

5;

1

(c) **1 mark for AO3 (refine)**

Change the < sign to <= // change `num1` to `num1 + 1`;

**A.** answers where line of code has been rewritten

1

**[3]**

**16.** **2 marks for AO3 (design) and 6 marks for AO3 (program)**

**<u>Program Design</u>**
**Mark A** for using an iterative structure to validate the user input of speed (even if logic is incorrect);
**Mark B** for using meaningful variable names **and** suitable data types throughout (speed can be real or integer, breaking distance must be real, the `IsWet` input must be string);

**Program Logic**
**Mark C** for getting user input for **both** the speed and `IsWet` in appropriate places;
**Mark D** for using a `WHILE` loop or similar to re-prompt for the user input (even if it would not work);
**Mark E** for using a correct Boolean condition with the validation structure;
**Mark F** for calculating the braking distance correctly (i.e. divided by 5);
**Mark G** for using a selection structure to adjust the braking distance calculation if the user input required it (even if it would not work);
**Mark H** for outputting the braking distance in a logically correct place;

**I.** Case of program code

**Maximum 7 marks** if any errors in code.

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

```
speed = float(input())                                       (Part of C)
while speed < 10 or speed > 50:                              (D, E)
   speed = float(input())                                    (Part of D)
braking_distance = speed / 5                                 (F)

IsWet = input()                                              (Part of C)
if IsWet == 'yes':                                           (Part of G)
   braking_distance = braking_distance * 1.5                 (Part of G)
print(braking_distance)                                      (H)
```

**Python Example (partially correct – 7 marks)**
All design marks are achieved **(Marks A and B)**

```
speed = float(input())                                       (Part of C)
while speed <= 10 and speed > 50                             (D, NOT E)
   speed = float(input())                                    (Part of D)
   braking_distance = speed / 5                              (F)

IsWet = input()                                              (Part of C)
if IsWet = 'yes'                                             (Part of G)
   braking_distance = braking_distance * 1.5                 (Part of G)
print(braking_distance)                                      (H)
```

**[8]**