Please write clearly in block capitals.

Centre number | | | | | |  Candidate number | | | | |

Surname _____

Forename(s) _____

Candidate signature _____
I declare this is my own work.

# GCSE
# COMPUTER SCIENCE

## Paper 1 Computational thinking and programming skills – Python

Time allowed: 2 hours

**Materials**
- There are no additional materials required for this paper.
- You must **not** use a calculator.

**Instructions**
- Use black ink or black ball-point pen. Use pencil only for drawing.
- Answer **all** questions.
- You must answer the questions in the spaces provided.
- If you need extra space for your answer(s), use the lined pages at the end of this book. Write the question number against your answer(s).
- Do all rough work in this book. Cross through any work you do not want to be marked.
- Questions that require a coded solution must be answered in Python.
- You should assume that all indexing in code starts at 0 unless stated otherwise.

**Information**
The total number of marks available for this paper is 90.

| For Examiner's Use | |
|---|---|
| Question | Mark |
| 1 | |
| 2–3 | |
| 4–5 | |
| 6–7 | |
| 8–9 | |
| 10 | |
| 11 | |
| 12 | |
| 13 | |
| 14 | |
| **TOTAL** | |

**Advice**

For the multiple-choice questions, completely fill in the lozenge alongside the appropriate answer.

CORRECT METHOD  WRONG METHODS

If you want to change your answer you must cross out your original answer as shown.

If you wish to return to an answer previously crossed out, ring the answer you now wish to select as shown.

J U N 2 2 8 5 2 5 1 B 0 1

IB/G/Jun22/E12

8525/1B

Do not write outside the box

Answer **all** questions.

**0 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`

**0 1 . 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 ⬭

Do not write outside the box

**0 1 . 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`

**0 1 . 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

**0 1 . 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

**Question 1 continues on the next page**

Do not write outside the box

**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
```

**0 1 . 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. ⬭

**0 1 . 6** **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.

**[1 mark]**

Do not write outside the box

| | | |
|---|---|---|
| **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` | ⬭ |

6

**Turn over for the next question**

Do not write outside the box

**0 2** **Figure 3** shows an algorithm, represented using pseudo-code, that calculates the delivery cost for an order from a takeaway company.

**Figure 3**

```
orderTotal ← USERINPUT
deliveryDistance ← USERINPUT
deliveryCost ← 0.0
messageOne ← "Minimum spend not met"
messageTwo ← "Delivery not possible"
IF deliveryDistance ≤ 5 AND orderTotal > 0.0 THEN
   IF orderTotal > 50.0 THEN
      deliveryCost ← 1.5
      OUTPUT deliveryCost
   ELSE IF orderTotal > 25.0 THEN
      deliveryCost ← (orderTotal / 10) * 2
      OUTPUT deliveryCost
   ELSE
      OUTPUT messageOne
   ENDIF
ELSE
   OUTPUT messageTwo
ENDIF
```

**0 2 . 1** Using **Figure 3**, complete the table.

**[2 marks]**

| **Input value of** `orderTotal` | **Input value of** `deliveryDistance` | **Output** |
|---|---|---|
| 55.5 | 2 | |
| 35.0 | 5 | |

**0 2 . 2** State how many possible values the result of the comparison `deliveryDistance ≤ 5` could have in the algorithm shown in **Figure 3**.

**[1 mark]**

Do not write outside the box

**0 2 . 3** State the most suitable data type for the following variables used in **Figure 3**.

**[2 marks]**

| Variable identifier | Data type |
|---|---|
| `deliveryCost` | |
| `messageOne` | |

**0 2 . 4** State **one** other common data type that you have **not** given in your answer to Question **02.3**.

**[1 mark]**

---

**Turn over for the next question**

Do not write outside the box

**0 3** **Figure 4** 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.

**Figure 4**

```
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)
```

**0 3 . 1** Rewrite **line 4** in **Figure 4** 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 mark]**

**0 3 . 2** The charge for parking for two or more hours is changed to include an additional £2 fee.

Rewrite **line 10** in **Figure 4** to show this change.

Your answer must be written in Python.

**[1 mark]**

8

Do not write outside the box

**0 4** The two Python programs in **Figure 5** 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`

**Figure 5**

**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)
```

**0 4 . 1** Shade **one** lozenge to indicate which of the statements is true about the programs in **Figure 5**.

**[1 mark]**

**A** Both programs are equally efficient. ⬭

**B** Program A is more efficient than Program B. ⬭

**C** Program B is more efficient than Program A. ⬭

**0 4 . 2** Justify your answer for Question **04.1**.

**[2 marks]**

_____

_____

_____

_____

Do not write outside the box

**0 5** A programmer has started to write a program using Python. 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` 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.

**Figure 6**

```
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])
```

**0 5 . 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. ⬭

Do not write outside the box

**0 5 . 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 Python.

**[2 marks]**

Line number _____

Corrected line _____

_____

**0 5 . 3** What type of data structure is the variable `numbers`?

**[1 mark]**

_____

8

**Turn over for the next question**

Do not write outside the box

**0 6** A program is being developed that allows users to rate and review movies. A user will enter their rating (out of 10) and a written review for each movie they have watched.

Computational thinking skills are used during the development of the program.

**0 6 . 1** Define the term **abstraction**.

**[1 mark]**

_____

_____

**0 6 . 2** A user will be able to register, log in and log out of the program. When registering, a new user will enter their details, before confirming their email address.

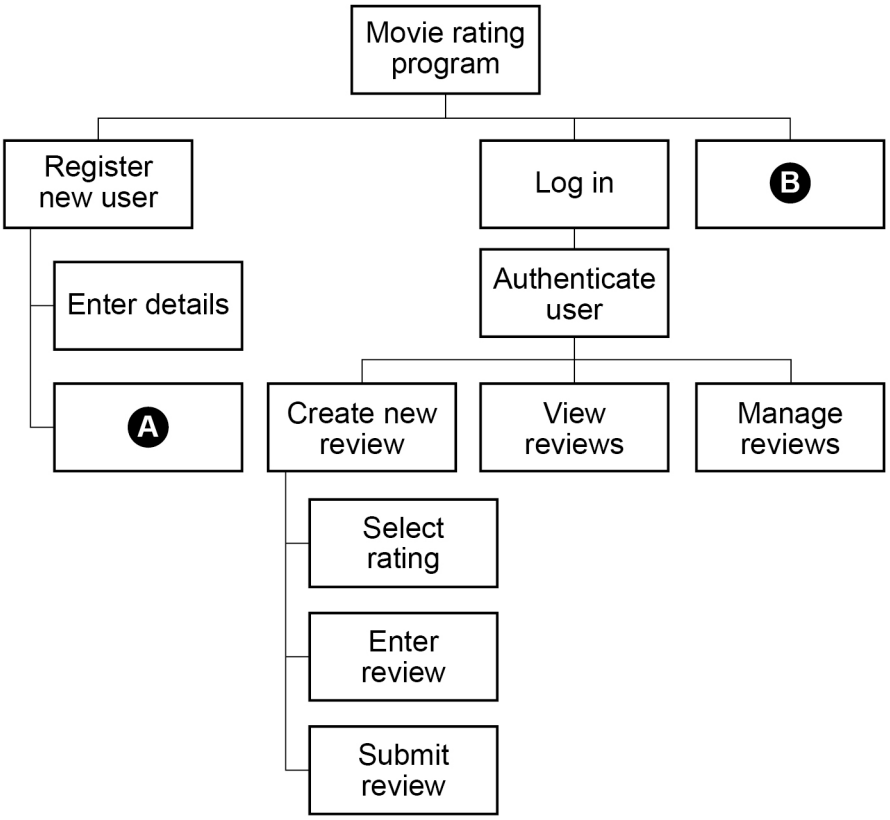
**Decomposition** has been used to break the problem down into smaller sub-problems.

**Figure 7** represents the design of the program.

Complete the decomposition of this program by stating what should be written in boxes **A** and **B**.

**[2 marks]**

**Figure 7**



*Do not write outside the box*

**A**

**B**

**Turn over for the next question**

Do not write outside the box

**0 7** 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.

**[5 marks]**

*Do not write outside the box*

8

**Turn over ►**

Do not write outside the box

**0 8** 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.

**[7 marks]**

*Do not write outside the box*

**Turn over ►**

Do not write outside the box

**0 9** **Figure 8** shows an algorithm represented using pseudo-code.

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

**Figure 8**

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

`SUBSTRING` returns part of a string.

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

**0 9 . 1** Shade **one** lozenge which shows the output of **line 4** from the algorithm shown in **Figure 8**.

**[1 mark]**

**A** `F` ⬭

**B** `Freddie` ⬭

**C** `Lily` ⬭

**D** `S` ⬭

**E** `Sarah` ⬭

Do not write outside the box

**0 9 . 2** Shade **one** lozenge which shows the output of **line 5** from the algorithm shown in **Figure 8**.

**[1 mark]**

**A** 1 ⬭

**B** 2 ⬭

**C** 4 ⬭

**D** 5 ⬭

**E** 10 ⬭

**0 9 . 3** State the output of **line 7** from the algorithm shown in **Figure 8**.

**[1 mark]**

_____

_____

**0 9 . 4** Two extra lines are being added to the end of the algorithm in **Figure 8**.

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

**[2 marks]**

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

OUTPUT names[ _______ ] + var
```

12

**Turn over for the next question**

Do not write outside the box

**1 0** **Figure 9** shows a subroutine represented using pseudo-code.

**Figure 9**

```
SUBROUTINE calculate(n)
   a ← n
   b ← 0
   REPEAT
      a ← a DIV 2
      b ← b + 1
   UNTIL a ≤ 1
   OUTPUT b
ENDSUBROUTINE
```

The `DIV` operator is used for integer division.

**1 0 . 1** Complete the trace table for the subroutine call `calculate(50)`

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

**[4 marks]**

| **n** | **a** | **b** | **OUTPUT** |
|---|---|---|---|
| 50 | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Do not write outside the box

**1 0 . 2** State the value that will be output for the subroutine call `calculate(1)`

**[1 mark]**

**1 0 . 3** The identifier for the variable `b` in **Figure 9** was not a good choice.

State a better identifier for this variable that makes the algorithm easier to read and understand.

**[1 mark]**

**Question 10 continues on the next page**

Do not write outside the box

**1 0 . 4** A `REPEAT…UNTIL` iteration structure was used in **Figure 9**.

**Figure 9** has been included again below.

**Figure 9**

```
SUBROUTINE calculate(n)
   a ← n
   b ← 0
   REPEAT
      a ← a DIV 2
      b ← b + 1
   UNTIL a ≤ 1
   OUTPUT b
ENDSUBROUTINE
```

**Figure 10** shows another subroutine called `calculate` that uses a `WHILE…ENDWHILE` iteration structure.

**Figure 10**

```
SUBROUTINE calculate(n)
   a ← n
   b ← 0
   WHILE a > 1
      a ← a DIV 2
      b ← b + 1
   ENDWHILE
   OUTPUT b
ENDSUBROUTINE
```

One difference in the way the subroutines in **Figure 9** and **Figure 10** work is:

- the `REPEAT…UNTIL` iteration structure in **Figure 9** loops until the condition is true
- the `WHILE…ENDWHILE` iteration structure in **Figure 10** loops until the condition is false.

Do not write outside the box

Describe **two** other differences in the way the subroutines in **Figure 9** and **Figure 10** work.

**[2 marks]**

1 _____

_____

_____

2 _____

_____

_____

8

**Turn over for the next question**

Do not write outside the box

**1 1 . 1** The size of a sound file is calculated using the following formula:

**size (in bits) = sampling rate * sample resolution * seconds**

To calculate the size **in bytes**, the number is divided by **8**

The algorithm in **Figure 12**, represented using pseudo-code, should output the size of a sound file in **bytes** that has been sampled 100 times per second, with a sample resolution of 16 bits and a recording length of 60 seconds.

A subroutine called `getSize` has been developed as part of the algorithm.

Complete **Figure 12** by filling in the gaps using the items in **Figure 11**.

You will not need to use all the items in **Figure 11**.

**[6 marks]**

**Figure 11**

| | | | |
|---|---|---|---|
| bit | byte | getSize | OUTPUT |
| rate | res | RETURN | sampRate |
| seconds | size | size + 8 | size * 8 |
| size / 8 | size MOD 8 | SUBROUTINE | USERINPUT |

**Figure 12**

```
SUBROUTINE getSize(____________, _____________, seconds)

    _______________ ← sampRate * res * seconds

    size ← ______________

    _______________ size
ENDSUBROUTINE


OUTPUT  ______________(100, 16, 60)
```

Do not write outside the box

**1 1 . 2** A local variable called `size` has been used in `getSize`.

Explain what is meant by a local variable in a subroutine.

**[1 mark]**

**1 1 . 3** State **three** advantages of using subroutines.

**[3 marks]**

1

2

3

10

**Turn over for the next question**

Do not write outside the box

**1 2** **Figure 13** shows an algorithm represented in pseudo-code. A developer wants to check the algorithm works correctly.

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

**Figure 13**

```
1   arr[0] ← 'c'
2   arr[1] ← 'b'
3   arr[2] ← 'a'
4   FOR i ← 0 TO 1
5      FOR j ← 0 TO 1
6         IF arr[j + 1] < arr[j] THEN
7            temp ← arr[j]
8            arr[j] ← arr[j + 1]
9            arr[j + 1] ← temp
10        ENDIF
11     ENDFOR
12  ENDFOR
```

Do not write outside the box

**1 2 . 1** Complete the trace table for the algorithm shown in **Figure 13**.

Some values have already been entered. You may not need to use all the rows in the table.

**[6 marks]**

| arr | | | i | j | temp |
|---|---|---|---|---|---|
| [0] | [1] | [2] | | | |
| c | b | a | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

**1 2 . 2** State the purpose of the algorithm.

**[1 mark]**

_____

_____

_____

**Question 12 continues on the next page**

Do not write outside the box

**1 2 . 3** **Figure 13** has been included again below.

**Figure 13**

```
1   arr[0] ← 'c'
2   arr[1] ← 'b'
3   arr[2] ← 'a'
4   FOR i ← 0 TO 1
5      FOR j ← 0 TO 1
6         IF arr[j + 1] < arr[j] THEN
7            temp ← arr[j]
8            arr[j] ← arr[j + 1]
9            arr[j + 1] ← temp
10        ENDIF
11     ENDFOR
12  ENDFOR
```

An earlier attempt at writing the algorithm in **Figure 13** had different code for **lines 4** and **5**.

**Lines 4** and **5** of the pseudo-code were:

```
FOR i ← 0 TO 2
   FOR j ← 0 TO 2
```

Explain why the algorithm did not work when the value 2 was used instead of the value 1 on these two lines.

**[1 mark]**

_____

_____

_____

8

Do not write outside the box

**1 3** 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.

**1 3 . 1** **Figure 14** shows part of the program that will get a player to enter the position of a card to swap.

**Figure 14**

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

Extend the program in **Figure 14**. 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 marks]**

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

**Question 13 continues on the next page**

Do not write outside the box

**1 3 . 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**

| | | | | | | |
|---|---|---|---|---|---|---|
| ... | 2 | 3 | 4 | 5 | 6 | ... |

**Valid run example 2**

| | | | | | | |
|---|---|---|---|---|---|---|
| ... | 99 | 100 | 101 | 102 | 103 | ... |

Below are examples of invalid runs.

**Invalid run example 1**

| | | | | | | |
|---|---|---|---|---|---|---|
| ... | 4 | 5 | 9 | 10 | 11 | ... |

**Invalid run example 2**

| | | | | | | |
|---|---|---|---|---|---|---|
| ... | 51 | 52 | 52 | 53 | 54 | ... |

**Invalid run example 3**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ... | 2 | 3 | 3 | 4 | 5 | 6 | ... |

Do not write outside the box

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 marks]**

**Question 13 continues on the next page**

*Do not write outside the box*

10

Do not write outside the box

**Turn over for the next question**

**DO NOT WRITE ON THIS PAGE**
**ANSWER IN THE SPACES PROVIDED**

Do not write outside the box

**1 4** 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.

Do not write outside the box

**1 4 . 1** **Figure 16** shows an incomplete Python 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 Python 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  ticket = [["","",""],
             ["","",""],
             ["","",""]]

2  i = 0
3  while i < 3:

4      j = ____
5      while j < 3:

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

7          ______________

8      ______________
```

**Question 14 continues on the next page**

Do not write outside the box

**1 4 . 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 Python 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 Python.

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.

**[8 marks]**

*Do not write outside the box*

**END OF QUESTIONS**

12

Do not write outside the box

**There are no questions printed on this page**

**DO NOT WRITE ON THIS PAGE**
**ANSWER IN THE SPACES PROVIDED**

*Do not write outside the box*

| Question number | **Additional page, if required.**<br>**Write the question numbers in the left-hand margin.** |
|---|---|
| | |

*Do not write outside the box*

| Question number | **Additional page, if required.**<br>**Write the question numbers in the left-hand margin.** |
|---|---|
| | |

**Copyright information**

For confidentiality purposes, all acknowledgements of third-party copyright material are published in a separate booklet. This booklet is published after each live examination series and is available for free download from www.aqa.org.uk.

Permission to reproduce all copyright material has been applied for. In some cases, efforts to contact copyright-holders may have been unsuccessful and AQA will be happy to rectify any omissions of acknowledgements. If you have any queries please contact the Copyright Team.

Copyright © 2022 AQA and its licensors. All rights reserved.