AS
COMPUTER SCIENCE

Paper 1

Tuesday 12 May 2026 Afternoon Time allowed: 1 hour 45 minutes

Materials
For this paper you must have:
- a computer
- a printer
- appropriate software
- the Electronic Answer Document
- an electronic version and a hard copy of the Skeleton Program
- an electronic version and a hard copy of the Preliminary Material

You must **not** use a calculator.

Instructions
- Type the information required on the front of your Electronic Answer Document.
- Before the start of the examination make sure your **Centre Number**, **Candidate Name** and **Candidate Number** are shown clearly **in the footer** of every page (not the front cover) of your Electronic Answer Document.
- Enter your answers into the Electronic Answer Document.
- Answer **all** questions.
- Save your work at regular intervals.

Information
- The marks for questions are shown in brackets.
- The maximum mark for this paper is 75.
- No extra time is allowed for printing and collating.
- The Question Paper is divided into **three** sections.

Advice
You are advised to allocate time to each section as follows:
Section A – 20 minutes; **Section B** – 25 minutes; **Section C** – 60 minutes.

At the end of the examination
Tie together all your printed Electronic Answer Document pages and hand them to the Invigilator.

Warning
It may not be possible to issue a result for this paper if your details are not on every page of your Electronic Answer Document.

Section A

You are advised to spend no more than **20 minutes** on this section.

Enter your answers to **Section A** in your Electronic Answer Document. You **must save** this document at regular intervals.

Question **03** in this section asks you to write program code **starting from a new program/project/file**.

You are advised to save your program at regular intervals.

0 1 The algorithm, represented using pseudo-code in **Figure 1**, describes a method to issue change from a vending machine.

The parameter `cost` contains the price, in pence, of a purchased item. The parameter `paid` contains the amount of money, in pence, inserted into the machine. Both parameters are integers, and `paid` will always be greater than or equal to `cost`

Coins available are worth 50, 20, 10, 5, 2, and 1 pence.

Figure 1

```
SUBROUTINE IssueChange(cost, paid)
  change ← paid - cost
  coinValues ← [50, 20, 10, 5, 2, 1]
  FOR i ← 0 TO LEN(coinValues) - 1
    IF change ≥ coinValues[i] THEN
      change ← change - coinValues[i]
      ReleaseCoin(coinValues[i])
    ENDIF
  ENDFOR
  RETURN Change
ENDSUBROUTINE
```

The `ReleaseCoin` subroutine causes the machine to release a coin of the specified amount.

0 1 . 1 Complete **Table 1** by hand-tracing the algorithm in **Figure 1** when the following statement is executed.

```
remaining ← IssueChange(24, 100)
```

Table 1

change	i	coinValues[i]

Copy the contents of all the unshaded cells in **Table 1** into your Electronic Answer Document.

[5 marks]

0 1 . 2 If the algorithm returns zero, then all the necessary change has been issued. Under some circumstances, however, not all necessary change is issued.

Give **one** example of a call to `IssueChange` that would result in some of the necessary change not being issued **and** explain why this is the case.

In your example, both parameters should be positive integer values less than 100.

[3 marks]

0 2 State **two** examples of tasks that would be undertaken during the analysis stage of software development.

[2 marks]

0 3 **Figure 2** shows an algorithm represented using pseudo-code.

Figure 2

```
OUTPUT "Enter the first number: "
INPUT in1
OUTPUT "Enter the second number: "
INPUT in2
IF in1 < 1 OR in2 < 1 THEN
  OUTPUT "Invalid input"
ELSE
  m ← 1
  found ← False
  WHILE found = False
    IF m MOD in1 = 0 AND m MOD in2 = 0 THEN
      found ← True
    ELSE
      m ← m + 1
    ENDIF
  ENDWHILE
  OUTPUT m / in1
  OUTPUT m / in2
  OUTPUT m
ENDIF
```

Table 2 lists the `MOD` operators for each of the available programming languages. You should refer to the row for your programming language.

Table 2

Programming language	**MOD**
C#	%
Java	%
Python	%
VB.NET	Mod

What you need to do:

Task 1
Write a program to implement the algorithm in **Figure 2**.

Task 2
Test that your program works:
- run your program
- enter `16` at the first prompt
- enter `22` at the second prompt.

Evidence that you need to provide
Include the following evidence in your Electronic Answer Document.

0 3 . 1 Your PROGRAM SOURCE CODE for **Task 1**.

[9 marks]

0 3 . 2 SCREEN CAPTURE(S) showing the test described in **Task 2**.

[1 mark]

Turn over for the next section

Section B

You are advised to spend no more than **25 minutes** on this section.

Enter your answers to **Section B** in your Electronic Answer Document. You **must save** this document at regular intervals.

These questions refer to the **Preliminary Material** and the **Skeleton Program**, but do **not** require any additional programming.

Refer **either** to the **Preliminary Material** issued with this question paper **or** your electronic copy.

0 4 State the identifier for:

0 4 . 1 a variable used to store a Boolean value.

[1 mark]

0 4 . 2 a user-defined subroutine that returns a string value.

[1 mark]

0 4 . 3 a user-defined subroutine that contains nested indefinite iteration.

[1 mark]

0 5 Explain the difference between definite and indefinite iteration.

[2 marks]

0 6 The **Skeleton Program** uses global variables.

State **one** advantage and **two** disadvantages of using global variables.

[3 marks]

0 7 The subroutine `GetRandomTile` uses random number generation to return either `TILE` or `NO_TILE`

0 7 . 1 State the probability of `NO_TILE` being returned when the subroutine `GetRandomTile` is called.

You should assume that each number within the random number generator's range is equally likely to be selected.

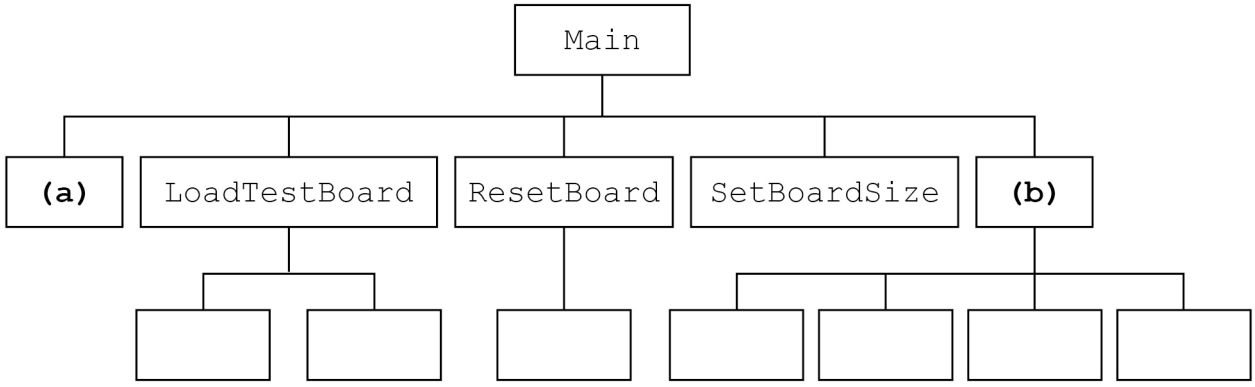
[1 mark]

0 7 . 2 Describe a change that could be made to the subroutine `GetRandomTile` that would increase the probability of `NO_TILE` being returned.

[2 marks]

0 8 **Figure 3** shows a partial hierarchy chart for the **Skeleton Program**.

Figure 3



0 8 . 1 What does each rectangle in a hierarchy chart represent?

[1 mark]

0 8 . 2 What should be written in rectangle **(a)** in **Figure 3**?

[1 mark]

0 8 . 3 What should be written in rectangle **(b)** in **Figure 3**?

[1 mark]

Turn over for the next question

0 9 This question refers to the subroutine `ProcessMove`

0 9 . 1 The subroutine `ProcessMove` correctly handles a single reference such as `A1` and two references connected by a dash, such as `B2-B4`

Describe how the subroutine would handle an input of three references connected by two dashes such as `C1-C2-C3`

[1 mark]

0 9 . 2 Explain why the input of `A1-A10` causes identical program behaviour to the input of `A1` on a 10 by 10 board.

[2 marks]

0 9 . 3 Describe the purpose of the third selection statement in the subroutine `ProcessMove`

[2 marks]

0 9 . 4 Describe how the variable `ToRemove` is used in the subroutine `ProcessMove`

[4 marks]

1 0 Describe how the subroutine `ConvertRefToCoords` would convert the reference `C6` to column 2 and row 5

[2 marks]

Section C

You are advised to spend no more than **60 minutes** on this section.

Enter your answers to **Section C** in your Electronic Answer Document. You **must save** this document at regular intervals.

These questions require you to load the **Skeleton Program** and to make programming changes to it.

1 1 This question refers to the subroutine `SetBoardSize`

Currently, when the board is resized, any width and any height can be specified.

Change the subroutine `SetBoardSize` so that both width and height can only be given a minimum value of 3 and a maximum value of 9

If the player specifies both a width and a height between 3 and 9, the size of the board should be set using these values.

If either of the values input by the player are greater than 9 or less than 3:

- the size of the board should remain unchanged
- the message `Invalid input - size of board unchanged` should be displayed.

You do **not** need to make the player re-enter a new height and width if either of the values that they enter are invalid.

Test your changes by running the **Skeleton Program** and then:

- enter `2` to set the board size
- enter `9` for the width
- enter `10` for the height
- enter `2` to set the board size
- enter `3` for the width
- enter `3` for the height.

Evidence that you need to provide
Include the following evidence in your Electronic Answer Document.

1 1 . 1 Your PROGRAM SOURCE CODE for the amended subroutine `SetBoardSize`

[6 marks]

1 1 . 2 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

1 2 This question refers to the subroutine `PlayGame` and to a new subroutine `InvertBoard`

Task 1
Create a new subroutine `InvertBoard` that iterates over each square on the board. All squares that contain a tile should be set to empty, and all squares that are empty should be set to contain a tile.

Figure 4 shows an example of how the board might appear before a call to the subroutine `InvertBoard` has been made. **Figure 5** shows the same board after a call to `InvertBoard` has been made.

Figure 4

```
    A   B   C   D
1  [X] [X] [X] [X]
2  [X] [X] [X] [X]
3  [ ] [ ] [ ] [ ]
4  [ ] [ ] [ ] [X]
```

Figure 5

```
    A   B   C   D
1  [ ] [ ] [ ] [ ]
2  [ ] [ ] [ ] [ ]
3  [X] [X] [X] [X]
4  [X] [X] [X] [ ]
```

Task 2
Modify the subroutine `PlayGame` so that when the player is prompted for a move and they enter the text `invert` the subroutine `InvertBoard` will be called. After this, the board will have been inverted, and it will be the turn of the next player.

Task 3
Test your changes by running the **Skeleton Program** and then:
- enter `4` to load the test board
- enter `invert`

Evidence that you need to provide
Include the following evidence in your Electronic Answer Document.

1 2 . 1 Your PROGRAM SOURCE CODE for the new subroutine `InvertBoard`

[5 marks]

1 2 . 2 Your PROGRAM SOURCE CODE for the amended subroutine `PlayGame`

[2 marks]

1 2 . 3 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

Turn over for the next question

1 3 This question refers to the subroutine `PlayGame` and to a new subroutine `PlayRandomMove`

This question involves creating a computer-generated move which will be legal but randomly selected.

Task 1
Create a new subroutine `PlayRandomMove` with no parameters and no return value. This new subroutine should behave as follows:

- A start square is selected at random.
- Either a horizontal or vertical move is selected, with a 1 in 2 chance of each option:
  - for a horizontal move, a random end square in the same row as the start square, **at or to the right of** the start square, is selected
  - for a vertical move, a random end square in the same column as the start square, **at or below** the start square, is selected.
- If the start and end square locations are in coordinate form, they should be converted to reference form by calling the subroutine `ConvertCoordsToRef`
- The start and end square references are concatenated with a dash to form a move such as `B3-B4` or `A5-A5`, which is passed to the subroutine `ProcessMove`
- The above steps should be repeated until a valid move has been made.
- Finally, the move is displayed as follows:
  - as a range if multiple tiles have been removed, eg `B3-B4`
  - as a single reference if only one tile has been removed, eg `A5`

Task 2
Modify the subroutine `PlayGame` so that, when the player is prompted for a move, if they enter the text `random`, the subroutine `PlayRandomMove` will be called.

Task 3
Test your changes by running the **Skeleton Program** and then:
- enter `1` to start the game
- enter `random` when prompted for a move.

Task 4
Test your changes by running the **Skeleton Program** and then:
- enter `4` to load the test board
- enter `random` when prompted for a move.

Evidence that you need to provide
Include the following evidence in your Electronic Answer Document.

1 3 . 1 Your PROGRAM SOURCE CODE for the new subroutine `PlayRandomMove`

[12 marks]

1 3 . 2 Your PROGRAM SOURCE CODE for the amended subroutine `PlayGame`

[1 mark]

1 3 . 3 SCREEN CAPTURE(S) showing the results of the requested test in **Task 3**.

[1 mark]

1 3 . 4 SCREEN CAPTURE(S) showing the results of the requested test in **Task 4**.

[1 mark]

END OF QUESTIONS

There are no questions printed on this page

There are no questions printed on this page

There are no questions printed on this page

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 © 2026 AQA and its licensors. All rights reserved.