OXFORD AQA

INTERNATIONAL QUALIFICATIONS

INTERNATIONAL GCSE
COMPUTER SCIENCE

Paper 1 Programming

Mock1 May 2025 Time allowed: 2 hours

Materials

For this paper you must have access to:

- a computer
- a printer
- appropriate software
- electronic versions of the **Preliminary Material** and the **Skeleton Program**
- a hard copy of the **Preliminary Material** and the **Skeleton Program**.

Information

- The marks for questions are shown in brackets.
- The maximum mark for this paper is 80.
- No extra time is allowed for printing and collating.
- The question paper is divided into **three** sections.
- You may use a bilingual dictionary.
- You must **not** use an English dictionary.
- You are **not** allowed to use a calculator.

Instructions

- Type the information required on the front of your **Electronic Answer Document**.
- Answer **all** questions.
- Enter your answers into the **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 of your **Electronic Answer Document** (not the front cover).
- Include the evidence required for your answers to **Sections B** and **C** in your **Electronic Answer Document**.
- You **must save** your **Electronic Answer Document** at regular intervals.
- The questions in **Sections B** and **C** require you to make changes to the **Skeleton Program**.
- All of the programming questions in **Sections B** and **C** can be answered independently of each other. If you cannot answer one of the questions you can still attempt to solve later questions.
- You are advised to keep a backup copy of the original **Skeleton Program** so that you can go back to it if you accidentally make changes to the program which means it can no longer be compiled/executed while answering the questions in **Sections B** and **C**.

Secure all your printed Electronic Answer Document pages together and hand them to the invigilator.

Section A (Non-programming questions)

You are advised to spend no more than **30 minutes** on this section.

Type your answers to **Section A** in your **Electronic Answer Document**.

You **must save** your **Electronic Answer Document** at regular intervals.

The questions in this section are about programming and how the **Skeleton Program** works. Do **not** make any changes to the **Skeleton Program** when answering these questions.

0 1 . 1 State the name of a variable in the **Skeleton Program** that is used to store multiple values at the same time.

[1 mark]

0 1 . 2 State the name of a user-defined subroutine in the **Skeleton Program** that does not return a value.

[1 mark]

0 1 . 3 Explain why subroutines often use parameters.

[2 marks]

0 1 . 4 Named constants are used to represent the four directions of turning.

State **two** advantages of using named constants.

[2 marks]

0 1 . 5 The subroutine `CreateBoard` uses FOR loops.

Explain why FOR loops are used rather than WHILE loops.

[2 mark]

0 1 . 6 The subroutine `Turn(rotation)` uses the data structures `directions`. These data structures are used to store data in four directions.

Explain the name of this data structure and the data types within it.

[2 marks]

Question 1 continues on the next page

0 1 . 7 State the purpose of the first statement inside the first FOR loop in the subroutine `CreateBoard()`.

[1 mark]

0 1 . 8 Explain why parameters are used in the `Turn()` subroutine.

[2 marks]

0 1 . 9 Explain the purpose of the `CheckForward()` subroutine.

[1 marks]

0 1 . 10 what is a local variable?
Identify the local variables in the `ProcessForward()` subroutine.

[3 marks]

0 2 . 1 In the subroutine `Turn`, Explain why the code uses `(directionIndex + 3) % 4` for left turns.

[3 marks]

0 2 . 2 What would happen if `(directionIndex -1)% 4` were used instead of `(directionIndex+3)% 4`?

[1 mark]

Turn over for the next section

Section B (Short programming questions)

You are advised to spend no more than **45 minutes** on this section.

0 3 The **Skeleton Program** is to be improved so that it displays a message when the user runs the program.

The message `POINTER NAVIGATION GAME` should be displayed when the program is run.

Change the subroutine `main` so that the message `POINTER NAVIGATION GAME` is displayed before any other output. This message should only be displayed once each time the program is run.

There must be a blank line after the message.

Test your changes by running the **Skeleton Program**.

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

0 3 . 1 **All** the PROGRAM SOURCE CODE for the amended subroutine `main`.

[3 marks]

0 3 . 2 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

0 4 The **Skeleton Program** is to be changed by adding an additional Fixed Bonus Cell.

- The current constants are:
  - EMPTY
  - OBSTACLE #
  - GOAL @

- The new constant pattern will be:
  - EMPTY
  - OBSTACLE #
  - GOAL @
  - CLEAR !

Add a new constant to store the character for a CLEAR.

Change the `CreateExampleBoard` subroutine so that a CLEAR is added at row 5, column 10 .

Modify `MoveForward` to check if the pointer lands on "!", then clear all non-wall obstacles.

- Display a message "Bonus collected! Obstacles cleared!"when triggered.

Test your changes by:
- Selecting option 2 (example board).
- Moving the pointer to (5, 10) and verifying obstacles disappear.

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

0 4 . 1 The PROGRAM SOURCE CODE that creates a new constant to represent a CLEAR. Change the `CreateExampleBoard` subroutine so that a CLEAR is added at row 5, column 10

[2 mark]

0 4 . 2 **All** the PROGRAM SOURCE CODE for the amended subroutine `MoveForward`.

[4 marks]

0 4 . 3 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

0 5 The **Skeleton Program** is to be improved that modify the `DrawBoard` subroutine to display:

1. Current pointer coordinates in the format (Row:X, Col:Y)
2. Total forward steps taken (Steps:M)
3. Total commands processed (Commands:K)

Requirements:

- Add these statistics **below** the board with a blank line separation

- Track forward steps (successful `FD` commands) and all valid commands

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

Running the program with the example board (option 2)

Executing mixed commands (e.g., `FD 2`, `RT`, inputs)

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

0 5 . 1 **All** the PROGRAM SOURCE CODE for the amended subroutine `DrawBoard`.

[4 marks]

0 5 . 2 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

Turn over for the next question

0 6 The **Skeleton Program** currently allows unlimited commands per game. Modify the `Main` subroutine to:

- Limit the total commands to 20
- If the player exceeds 20 commands before reaching the goal (@), display:
- `Game Over! Maximum 20 commands reached.`
- End the game immediately

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

- Select the example board (option 2).
- Entering 20+ commands (e.g., "FD 1" repeated) without reaching @.
- Verifying the game ends with the error message at the 21st command.

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

0 6 . 1 **All** the PROGRAM SOURCE CODE for the amended subroutine `main`.

[5 marks]

0 6 . 2 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

0 7 The `MoveForward` subroutine in the **Skeleton Program** moves the pointer forward based on the current direction. However, it does not currently check whether the move would result in the pointer going **off the board**.

Change the `MoveForward` subroutine so that:

-Calculating the target row and column before moving.

-Checking whether the target position is within the board boundary (i.e. not less than 0 and not greater than or equal to HEIGHT or WIDTH).

-If the move is outside the boundary, set errorCode = 1 and cancel the movement.

-If the move is valid, allow the pointer to move.

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

Run the program and select option 2
Enter LT
Enter FD 2
Enter GO

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

0 7 . 1 **All** the PROGRAM SOURCE CODE for the amended subroutine `MoveForward`.

[7 marks]

0 7 . 2 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

Turn over for the next question

Section C (Longer programming questions)

You are advised to spend no more than **45 minutes** on this section.

0 8 The **Skeleton Program** is to be modified so that Implement a new teleportation feature that allows the pointer to instantly move to any empty cell on the board.

Task 1
Create a new subroutine called `Teleport` that:

- Takes parameters `targetRow` and `targetColumn`.
- Checks if the target position is within board boundaries and empty.
  - If valid, moves the pointer to the target position.
  - If invalid (out of bounds or obstacle), sets errorCode = 1.

Task 2
Displays appropriate success/failure messages.

Task 3
Modify the `ProcessInput` subroutine to complete the transfer function

Test:
Test teleportation to:

A valid empty position (e.g., (5,5))

An invalid position (e.g., (0,0) or an obstacle)

0 8 . 1 **All** the PROGRAM SOURCE CODE for the new subroutine `Teleport`.

[9 marks]

0 8 . 2 **All** the PROGRAM SOURCE CODE for the amended subroutine `ProcessInput`.

[6 mark]

0 8 . 3 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

Turn over for the next question

0 9 The **Skeleton Program** is to be modified so that add command to reverse pointer direction.

Task 1
Create a new subroutine called `TurnAround` that:

- Reverses current direction:
  - UP ↔ DOWN
  - LEFT ↔ RIGHT
- Updates pointer display

Task 2
Modify the `ProcessInput` subroutine to:

Handle new **TURN** command

Call `TurnAround` when encountered

Maintain existing error checking

Task 3
Test by:

Facing each cardinal direction

Executing **TURN** command

Verifying correct reversal

0 9 . 1 **All** the PROGRAM SOURCE CODE for new subroutine `TurnAround`.

[7 marks]

0 9 . 2 **All** the PROGRAM SOURCE CODE for the amended subroutine `ProcessInput`.

[4 mark]

0 9 . 3 SCREEN CAPTURE(S) showing the results of the requested test.

[1 mark]

END OF QUESTIONS

There are no questions printed on this page