
INTERNATIONAL GCSE

COMPUTER SCIENCE

Paper 1 Programming

Friday 31 October 2025

07:00 UK Time

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 subroutine in the **Skeleton Program** that has no parameters.
[1 mark]

0	1
---	---

.

2

 State the name of a subroutine in the **Skeleton Program** that contains nested iteration.
[1 mark]

0	1
---	---

.

3

 The **Skeleton Program** is divided into subroutines.
State **three** advantages of dividing a program into subroutines.
[3 marks]

0	1
---	---

.

4

 The subroutine `playGame` uses a `WHILE` loop.
Explain why a `WHILE` loop is used instead of a `FOR` loop.
[1 mark]

0	1
---	---

.

5

 Describe the purpose of the last `IF` statement in the subroutine `drawStreet`.
[2 marks]

0	1
---	---

.

6

 The subroutines `getPickUpLocation` and `getDropOffLocation` both return a place chosen at random from the `places` data structure.
Describe the purpose of the statements in `getDropOffLocation` that do **not** also exist in `getPickUpLocation`.
[2 marks]

0	2
---	---

The subroutine `processBids` chooses the winning bid.

0	2	.	1
---	---	---	---

Explain how the code causes Player 1's bid to win if both players in a two-player game make the same bid.

[1 mark]

0	2	.	2
---	---	---	---

Describe the effect of changing `processBids` so that the `<` operator in the `IF` statement is replaced by a `<=` operator.

[1 mark]

0	2	.	3
---	---	---	---

Describe how the subroutine `processBids` could be changed so that Player 1's bid sometimes wins and Player 2's bid sometimes wins when the bids are equal in a two-player game.

You should **not** make any changes to the **Skeleton Program** when answering this question.

[2 marks]

Turn over for the next question

Turn over ►

0 3 . 1 Three computer science terms are:

- A** constant
- B** declaration
- C** iteration

Complete **Table 1** by adding the label (**A–C**) of the most suitable computer science term to each row.

You **must** only use each letter once.

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

[2 marks]

Table 1

	Label
The repetition of statements in an algorithm	
The process of creating a new variable	
A named value that cannot change once set	

0 3 . 2 Give **one** example of a statement from the **Skeleton Program** in which an empty string is assigned to a variable.

[1 mark]

0 3 . 3 Define the term **meaningful identifier names**.

[1 mark]

0 4

Part of the subroutine `processBids` is represented using pseudocode in **Figure 1**.

Figure 1

```

winningBid ← players[0].bid
winningIndex ← 0
FOR p ← 1 TO LEN(players) - 1
    IF players[p].bid < winningBid THEN
        winningBid ← players[p].bid
        winningIndex ← p
    ENDIF
ENDFOR
OUTPUT "Player " + (winningIndex + 1) + " wins the job"

```

0 4 . 1

The game could be played by three players.

The three players place bids of 5, 4 and 4. These values are stored in the `bid` field of each player's record in the `players` data structure.

Complete the trace table for the value 5 stored in `players[0].bid`, 4 in `players[1].bid` and 4 in `players[2].bid`

Copy the contents of all the unshaded cells in **Table 2** into your **Electronic Answer Document**.

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

Table 2

winningBid	winningIndex	p

[3 marks]

0 4 . 2

State the message that would be output by the pseudocode in **Figure 1** when three players place bids of 5, 4 and 4.

[1 mark]

Turn over for the next section

Turn over ►

Section B (Short programming questions)

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

0	5
---	---

The **Skeleton Program** is to be improved so that it displays a new message showing the winning bid after the existing message showing the winning player.

Change the subroutine `processBids` to display the new message on the line below the existing message. For example, if Player 1 wins, the output should be:

```
Player 1 wins the job  
The bid was X
```

where X was the amount bid.

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

- enter 3
- enter 4
- enter 5

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 subroutine `processBids`

[3 marks]

0	5
---	---

2

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

[1 mark]

0	6
---	---

The **Skeleton Program** is to be improved so that an address is shown after each place. All the addresses are on North Street and have odd numbers, starting at 1

Modify the subroutine `drawStreet` so that each place includes an address in the format:

(X North Street)

where X is the next odd number in sequence.

For example, if the first three places are Gym, Cafe and Office, the output should be as follows:

```
Gym (1 North Street)
Cafe (3 North Street)
Office (5 North Street)
```

There should be a space between the place name and the opening bracket.

Your code should be written so that all places will have an address, even if the number of places changes.

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

- enter 3

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 subroutine `drawStreet`

[5 marks]

0	6	.	2
---	---	---	---

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

[1 mark]

Turn over for the next question

Turn over ►

0	7
---	---

The **Skeleton Program** is to be improved so that bids are limited. Currently there is no upper limit to the value of a bid.

Change the subroutine `collectBids` so that:

- the bid entered by each player is only accepted if it is less than or equal to **twice** the number of places in the program. For example, if the `places` data structure contains 6 places, the highest valid bid is 12
- if a player enters a bid that is too large:
 - the message `Bid too high` is displayed
 - the player is asked to enter another bid until a valid bid has been entered.

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

- enter 3
- enter 15
- enter 14
- enter 15

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 subroutine `collectBids`

[7 marks]

0	7	.	2
---	---	---	---

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

[1 mark]

0 8

The **Skeleton Program** is to be improved so that each journey uses up fuel. Currently, each player starts the game with 30 units of fuel. This value never changes.

Modify the subroutine `processBids` so that:

- for the player who makes the winning bid, their fuel is reduced by the cost of the journey. For example, if a journey has a cost of 5, then 5 should be subtracted from the fuel of the player who placed the winning bid. The subroutine already calculates the cost of a journey
- after the journey has been made:
 - if a player's fuel is between 1 and 5 inclusive, the message `Player X is low on fuel` should be displayed. X is the number of the player
 - if a player's fuel is 0 or lower, the message `Player X is out of fuel` should be displayed. X is the number of the player.

Modify the subroutine `playGame` so that:

- within the `FOR` loop, a check is made for each player having no fuel
- if a player has no fuel, the game should end.

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

- enter 3
- repeatedly enter 5 until a player is out of fuel.

Evidence that you need to provide

Include the following evidence in your Electronic Answer Document.

0 8 . 1

All the PROGRAM SOURCE CODE for the subroutine `processBids`

[7 marks]

0 8 . 2

All the PROGRAM SOURCE CODE for the subroutine `playGame`

[2 marks]

0 8 . 3

SCREEN CAPTURE(S) showing the results of the requested test. Your screen capture only needs to include output from the last display of the line `Player 1 - Balance` to the end of the output on screen.

[1 mark]

Turn over for the next section

Turn over ►

Section C (Longer programming questions)

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

0	9
---	---

The **Skeleton Program** is to be improved so that all players are offered a chance to move to a new location after a job has been completed.

Task 1

Create a new subroutine called `movePlayers` that:

- takes `places` and `players` as parameters
- for each player, displays the message `Player X, move to?` where `X` is the player number
- if a player presses the Enter key without having typed any text, the player remains in their current position
- if a player enters a place name, the program checks whether that place name exists within `places`
 - if it does exist, the player's position changes to match the place name they entered
 - if it does not exist, the message `No such place` is output. The player should **not** be given another opportunity to enter a place, and they should remain in the same position.

Your code should give each player the chance to move to a new position, even if the number of players is greater than two.

Task 2

Modify the subroutine `playGame` so that after `processBids` has been called, `drawStreet` is called, then the new subroutine `movePlayers` is called.

Task 3

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

- enter 3
- enter 5
- enter 5 again
- press the Enter key without having typed any text
- enter `Cafe`

Evidence that you need to provide

Include the following evidence in your Electronic Answer Document.

0 9 . 1 All the PROGRAM SOURCE CODE for the new subroutine `movePlayers` **[12 marks]**

0 9 . 2 All the PROGRAM SOURCE CODE for the subroutine `playGame` **[2 marks]**

0 9 . 3 SCREEN CAPTURE(S) showing the results of the requested test. **[1 mark]**

Turn over for the next question

Turn over ►

1	0
---	---

The **Skeleton Program** is to be improved. Each player should be shown the value of the lowest bid they can make which will **not** lose them any money. A player may still bid less than this value.

The code for this question only needs to work for a two-player game.

Task 1

Create a new subroutine called `getLowestBid` that:

- has four parameters – `places`, `current`, `pickUp` and `dropOff`
 - `places` is the data structure containing the names of the places
 - `current` is the integer representing a player's current position
 - `pickUp` is the name of the place where the package is to be picked up
 - `dropOff` is the name of the place where the package is to be dropped off
- finds the positions of `pickUp` and `dropOff` in the `places` data structure
- calculates the distance from the player's current position to the pick-up location **and then** to the drop-off location
- returns the calculated distance.

Task 2

Modify the subroutine `playGame` so that:

- **after** the text beginning `Transport a package` is displayed, and **before** the players' bids are entered, the new subroutine `getLowestBid` is called once for each player
- the value returned from each call is stored in a list or array called `lowestBids`
- `lowestBids` is passed as a new parameter in the call to `collectBids`

Task 3

Modify the subroutine `collectBids` so that:

- the list or array `lowestBids` is added as a parameter
- the lowest bid for a player is displayed before their bid is entered. The format should be `Minimum for player N: B` where `N` is the player's number and `B` is the lowest bid the player can make which will **not** lose them any money.

Task 4

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

- enter 3
- enter 5

Evidence that you need to provide

Include the following evidence in your Electronic Answer Document.

1 0 . 1 All the PROGRAM SOURCE CODE for the new subroutine `getLowestBid` **[6 marks]**

1 0 . 2 All the PROGRAM SOURCE CODE for the subroutine `playGame` **[4 marks]**

1 0 . 3 All the PROGRAM SOURCE CODE for the subroutine `collectBids` **[4 marks]**

1 0 . 4 SCREEN CAPTURE(S) showing the results of the requested test. **[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.oxfordaqa.com

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

Copyright © 2025 OxfordAQA International Examinations and its licensors. All rights reserved.



2 5 B Y 9 2 1 0 / 1