OXFORD

INTERNATIONAL
AQA EXAMINATIONS

INTERNATIONAL GCSE
COMPUTER SCIENCE

Paper 1 Programming

Time allowed: 2 hours

Materials

For this paper you must have access to:

- a computer
- a printer
- appropriate software
- an electronic version of the **Skeleton Program**
- a hard copy of 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 The **Skeleton Program** contains a number of subroutines written by the programmer.

Describe **three** advantages of using subroutines.

[3 marks]

0 1 . 2 State the name of a variable in the **Skeleton Program** that holds an integer value.

[1 mark]

0 1 . 3 State the name of a programmer–written subroutine in the **Skeleton Program** that has **exactly** three parameters.

[1 mark]

0 1 . 4 The `GetGameState` subroutine uses a selection statement to determine the value to return.

Describe the circumstances under which the string `Won` is returned.

[1 mark]

0 1 . 5 The board is displayed as two rows, but a one-dimensional array has been chosen to represent this instead of a two-dimensional array.

State **two** reasons why using a one-dimensional array instead of a two-dimensional array makes it easier to program, moving from pit to pit.

[2 marks]

0 2 This question is about the `DropSeeds` subroutine.

0 2 . 1 Explain what is meant by indefinite iteration.

[1 mark]

0 2 . 2 The `DropSeeds` subroutine uses indefinite iteration.

Explain why this could be changed to make use of definite iteration.

[1 mark]

0 2 . 3 The MOD operator is written as `%` in C#, `%` in Python, and `Mod` in Visual Basic.

In the subroutine `DropSeeds` the MOD operator is used in two assignment statements.

Explain the purpose of these assignment statements and why the MOD operator has been used in them.

[2 marks]

0 2 . 4 Complete the trace table in **Table 1** for the subroutine call `DropSeeds(1)`.

In this game the board has been setup to have only four pits and the current values in `Board` have been filled in for you. The value of the parameter `Pit` has also been filled in for you.

You may not need all the rows in **Table 1**.

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

[6 marks]

Table 1

Board				Pit	Seeds	DropPit	PitCount
0	1	2	3				
1	4	1	2	1			

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 at the top of the main menu.

The message `Capture The Seeds` should be displayed with a blank line between it and the menu choices.

Change the subroutine `DisplayMenu` so that this message followed by a blank line is displayed **before** the menu choices.

Test your changes by running the **Skeleton Program** and selecting `Play the test board`.

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

0 3 . 1 Your PROGRAM SOURCE CODE for the subroutine `DisplayMenu`.

[3 marks]

0 3 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

0 4 The **Skeleton Program** is to be improved so that it can identify when a game is drawn.

A game is drawn when both players have captured half of the seeds.

Change the subroutine `GetGameState` so that:
- it tests to see if the game is drawn
- if the game is drawn it returns the string `Drawn`

Test your changes by running the **Skeleton Program** and then:
- select `Play the test board`
- enter a pit number of `2`
- enter a pit number of `10`

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

0 4 . 1 Your PROGRAM SOURCE CODE for the subroutine `GetGameState`.

[5 marks]

0 4 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Turn over for the next question

0 5 The **Skeleton Program** is to be changed so that if a player picks a pit with no seeds in it for their turn, they are asked to pick another pit.

Change the subroutine `GetMove` so that:

- if the pit the player picks has no seeds in it the message `No seeds - pick again` is displayed
- the player is asked again which pit to pick seeds from and this is repeated until a pit is picked that contains seeds.

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

- select `Play the test board`
- enter a pit number of `1`
- enter a pit number of `0`

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

0 5 . 1 Your PROGRAM SOURCE CODE for the subroutine `GetMove`.

[5 marks]

0 5 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

0 6 The **Skeleton Program** is to be improved so that when setting up the board the number of pits is set to an appropriate value.

The entered number of pits should be checked to make sure that:
- a number between 10 and 20, inclusive, is entered
- the number is even.

Change the `GetInitialValues` subroutine so that:
- the value entered is checked to make sure it is between 10 and 20
- the value entered is checked to make sure it is even
- if any of the checks are failed, the message `Invalid value` should be displayed and the user should be asked to enter another value.

Test your changes by running the **Skeleton Program** and then:
- select `Setup a basic board`
- enter a value for the number of pits of `15`
- enter a value for the number of pits of `24`
- enter a value for the number of pits of `14`

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

0 6 . 1 Your PROGRAM SOURCE CODE for the subroutine `GetInitialValues`.

[7 marks]

0 6 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Turn over for the next question

0 7 Currently, the **Skeleton Program** does not always distribute all of the seeds to pits when a board is set up. For example, for a board with 10 pits and 25 seeds, 2 seeds would be put into each pit and 5 seeds would not be distributed.

The **Skeleton Program** is to be changed so that when distributing the seeds during the board setup, it makes sure that all seeds are distributed.

The `CreateBoard` subroutine is to be changed so that:

- after the seeds have been distributed using the existing process it calculates how many seeds are left
- if any seeds are left, it asks `Which pit should get the remaining seeds?`
- the user's response is stored in an appropriately named variable
- the board is updated so that these remaining seeds are placed into the requested pit.

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

- select `Setup a basic board`
- enter a value for the number of pits of `10`
- enter a value for the number of seeds of `25`
- enter a pit number of `4`
- select `Play a basic game`

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

0 7 . 1 Your PROGRAM SOURCE CODE for the `CreateBoard` subroutine.

[6 marks]

0 7 . 2 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

There are no questions printed on this page

Turn over for Section C

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 extended so that the board can be set up to have the seeds randomly dropped into the pits. When a player chooses to have a random board they will be asked to enter the number of pits and the number of seeds as they do for a basic board.

Task One

Change the `Menu` subroutine so that it displays an additional option.

```
R - Setup a random board
```

Task Two

Change the `Main` subroutine so that when the user enters `R` it asks for the number of pits and the number of seeds to be entered. It should then call `SetupBoard` with the required parameters including `TypeOfBoard` set to `R`

Task Three

Change the `SetupBoard` subroutine so that if the `TypeOfBoard` parameter is `R` it:

- displays the message `Setting up random board`
- calls the `CreateBoard` subroutine so that the board has the number of pits entered by the user and each pit contains no seeds
- initialises the `State` variable
- drops each seed into a randomly chosen pit on the board until all seeds have been dropped.

Task Four

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

- select `Setup a random board`
- enter a value for the number of pits of `10`
- enter a value for the number of seeds of `20`
- select `Play a basic game`

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

0 8 . 1 Your PROGRAM SOURCE CODE for the `Menu` subroutine.

[1 mark]

0 8 . 2 Your PROGRAM SOURCE CODE for the `Main` subroutine.

[4 marks]

0 8 . 3 Your PROGRAM SOURCE CODE for the `SetupBoard` subroutine.

[10 marks]

0 8 . 4 SCREEN CAPTURE(S) to show the result of carrying out the test.

[1 mark]

Turn over for the next question

0 9 The **Skeleton Program** is to be extended so that there is a new way to collect seeds when making a move.

The seeds will be distributed by the player and if the last seed is dropped into an empty pit then the player collects all of the seeds from the **next** pit.

Example

In an example game the board is currently:

```
 11  10   9   8   7   6
+---+---+---+---+---+---+
| 2 | 0 | 5 | 4 | 3 | 2 |  Player 1 holds: 0
+---+---+---+---+---+---+
| 5 | 3 | 0 | 4 | 0 | 2 |  Player 2 holds: 0
+---+---+---+---+---+---+
  0   1   2   3   4   5
```

If Player 1 picks pit 9, they will pick up 5 seeds.

These seeds will be dropped into pits 10, 11, 0, 1 and 2.

As the last seed was dropped into an empty pit, pit 2, they will collect all of the seeds from pit 3.

The state of the board after this move is shown below:

```
 11  10   9   8   7   6
+---+---+---+---+---+---+
| 3 | 1 | 0 | 4 | 3 | 2 |  Player 1 holds: 4
+---+---+---+---+---+---+
| 6 | 4 | 1 | 0 | 0 | 2 |  Player 2 holds: 0
+---+---+---+---+---+---+
  0   1   2   3   4   5
```

Task One

Create a new subroutine `CollectNext` that:

- takes the parameters `Pit` and `Player`
- calls the `DropSeeds` subroutine to place the seeds into the pits
- checks whether the last seed was dropped into an empty pit and, if it was, collects all the seeds in the next pit
- displays the message `Collected Seeds` if any seeds were collected.

Task Two

Change the `PlayGame` subroutine so that after getting the move from the player:

- it asks, with an appropriate prompt, if the player wishes to use the original way of collecting seeds or the new way of collecting seeds
- it allows the user to enter `A` for the original way of collecting seeds or `B` for the new way of collecting seeds
- it checks the entered value and calls either `MakeMove` or `CollectNext`, as appropriate.

Task Three

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

- select `Play the test board`
- enter a pit number of `0`
- enter a value for a collection method of `B`
- enter a pit number of `11`
- enter a value for a collection method of `B`
- enter a pit number of `3`
- enter a value for the collection method of `A`

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

0 9 . 1 Your PROGRAM SOURCE CODE for the new subroutine `CollectNext`.

[9 marks]

0 9 . 2 Your PROGRAM SOURCE CODE for the subroutine `PlayGame`.

[5 marks]

0 9 . 3 SCREEN CAPTURE(S) to show the result of carrying out the 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