OXFORD

INTERNATIONAL
AQA EXAMINATIONS

Please write clearly in block capitals.

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

Surname _____

Forename(s) _____

Candidate signature _____

INTERNATIONAL GCSE

COMPUTER SCIENCE

PAPER 1 PROGRAMMING

Date of exam Session Time allowed: 2 hours

For Examiner's Use	
question	Mark
1	
2	
3	
4	
5	
6	
7	
8	
9	
TOTAL	

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 **Preliminary Material**.

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 may **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).

Secure all your printed Electronic Answer Document pages 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** this document at regular intervals.

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

0 1 . 1 The **Skeleton Program** makes use of integer type variables, for example `Row`.

Explain the difference between an integer variable and a real variable.

[2 marks]

0 1 . 2 The **Skeleton Program** is decomposed into subroutines.

Describe **three** advantages of decomposing a program into subroutines.

[3 marks]

0 1 . 3 Indefinite iteration is used in the subroutine `PlayGame`.

State the command that is used in `PlayGame` to achieved indefinite iteration.

[1 mark]

0 1 . 4 How many parameters does the subroutine `DisplayBoard` have?

[1 mark]

0 1 . 5 Describe **two** advantages of using parameters in subroutine definitions.

[2 marks]

0 1 . 6 State the name of a local variable that is used in the `DisplayBoard` subroutine.

[1 mark]

0 1 . 7 Explain how local variables differ from global variables.

[2 marks]

0 2 . 1 In the subroutine `PlaceSnakesAndLadders` index `72` of the `Board` array has the value `20` assigned to it.

Explain what effect this assignment has within the game.

[3 marks]

0 2 . 2 The subroutine `CalculateSquare` converts a pair of `Row` and `Column` values to a `Square` number.

Explain why the calculation performed is different for odd and even numbered rows.

[1 mark]

0 2 . 3 The board is displayed as a grid, but a one-dimensional array has been chosen to represent this instead of a two-dimensional array.

Explain why this is an appropriate choice for this game.

[2 marks]

0 2 . 4 In the subroutine `PlayGame` the `MOD` operation is used in the following assignment statement:

```
PlayerNum = (PlayerNum + 1) MOD 2
```

Note the `MOD` operation is written as `%` in Python and C#.

Explain what the purpose of this assignment statement is and what the `MOD` operation does in the assignment statement.

[2 marks]

Turn over for the next question

Section B (Short programming questions)

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

Include the evidence required for your answers to **Section B** in your Electronic Answer Document.

You **must save** this document at regular intervals.

The questions in this section require you to make changes to the **Skeleton Program**.

Note that all of the programming questions in sections B and C can be answered independently of each other. Therefore, 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.

0 3 The **Skeleton Program** is to be improved so that it displays some additional information to help the player play the game.

The message `"WELCOME TO SNAKES AND LADDERS!"` should be displayed at the start of a game.

Modify the subroutine `PlayGame` so that this message is displayed once at the start.

Test that your change has worked by running the **Skeleton Program**.

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

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

[2 marks]

0 3 . 2 SCREEN CAPTURE(S) to show that the message is displayed at the start of the game.

[1 mark]

0 4 For each turn, the message `"You have landed on square: X"`, where X is the number of the square that the player has landed on is to be displayed.

If the player has landed on a ladder or a snake, X should be the number of the square that the player is on after they have moved up or down the ladder/snake.

This message should be displayed after the existing message that tells the player what value the die has rolled and before the board is redisplayed to show the player's new position.

Modify the subroutine `MakeMove` so that the message is displayed when a turn is taken.

Test that your change has worked by running the **Skeleton Program** and taking one turn.

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

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

[4 marks]

0 4 . 2 SCREEN CAPTURE(S) to show that the message is displayed after one turn.

[1 mark]

0 5 If the player lands on a snake or a ladder the **Skeleton Program** currently displays the message `"You have landed on a snake or ladder"`.

The information provided in this message is to be changed so that the player is told whether they landed either on a snake or a ladder and, if so, between which two squares they will be moving.

For example:

`"You are climbing up a ladder from square X to square Y"`

or

`"You are sliding down a snake from square X to square Y"`

where X is the square that the player landed on when the die was rolled and Y is the square that the snake or ladder took them to.

Modify the subroutine `MakeMove` so that an appropriate message is displayed when a turn is taken if the player lands either on a snake or a ladder.

Test that your change has worked by running the **Skeleton Program** and taking one turn. When taking the turn, select option `J` and jump to square `58`.

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

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

[6 marks]

0 5 . 2 SCREEN CAPTURE(S) to show that a suitable message is displayed when the player lands on square `58`.

[2 marks]

0 6 The **Skeleton Program** lets the player play the game only once. After this, the player must run the program again to play a second game.

The **Skeleton Program** is to be changed so that after a game finishes, the message `"Do you want to play again (Y/N)?"` is displayed.

If the player types in a letter `Y` the game should automatically play again. If `N` is typed, then the program should exit.

Modify the subroutine `Main` so that the game can be played again, as described above.

To test that your change has worked:

- run the **Skeleton Program**
- select option `J` and jump to square `99`, which will cause the game to finish
- when asked if you want to play again, answer `Y` and check that the game starts again.

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

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

[7 marks]

0 6 . 2 SCREEN CAPTURE(S) to show that when the game ended, and you asked to play again, a new game started.

In your SCREEN CAPTURE(S) it must be possible to see that the player has entered `Y` and that the game has restarted.

[1 mark]

0 7 In the **Skeleton Program** the game finishes when a player lands on square 99 or goes past it.

The game is to be changed so that it finishes only when a player lands on square 99. If a player rolls a number on the die that would take them past square 99 they should not move on that turn.

For example:

- If a player is on square 94 and rolls a 5 they will land on square 99 and the game should finish.
- If a player is on square 95 and rolls a 5 the roll would take them past square 99 to square 100, so the player should stay on square 95 for that turn.

If the player cannot move because their roll would take them past square 99 then the message `"Turn missed as roll too high"` should be displayed.

Modify the subroutine `MakeMove` so that if a die roll would take the player past square 99 the player does not move and the required message is displayed.

To test that your change has worked:

- run the **Skeleton Program**
- select option `J` and jump to square `98`
- press the `Enter` key so that player 1 has a turn.
- press the `Enter` key so that player 0 has a second turn. If the number rolled is greater than 1 then the message should be displayed to say that the player has missed a turn. **If, when the computer rolls the die, the value rolled is 1 you will need to re-run the test so that a number higher than 1 is rolled to prove that the player does not move if their roll would take them past square 99.**

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

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

[4 marks]

0 7 . 2 SCREEN CAPTURE(S) to show that when the die roll takes the player past square 99 they stay on their current square and the required message is displayed.

[2 marks]

Section C (Longer programming questions)

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

Include the evidence required for your answers to **Section C** in your Electronic Answer Document.

You **must save** this document at regular intervals.

The questions in this section require you to make changes to the **Skeleton Program**.

0 8 A new feature is to be implemented so that if a player lands on the square that is already occupied by the other player then one of the players will be sent back to the start of the board.

The player to be sent back will be chosen at random with an equal chance of each player being chosen.

Suitable message(s) will need to be displayed so that the players understand what has happened.

For example, if Player 0 is on square 23 and Player 1 then lands on square 23 then one of the two players, selected at random, will be moved back to square 0 whilst the other will stay on square 23.

Create a new subroutine called `MoveBackIfSameSquare`. This subroutine should test if the two players are on the same square and, if they are, should send one of the players back to square 0. The positions of the players should be passed to the subroutine as parameters.

The new subroutine should be called from the `PlayGame` subroutine, after the `MakeMove` subroutine has been called.

If you are unable to implement this feature by creating a new subroutine with parameters, you will still be able to get most of the marks for it if you write all of the code to implement it in the `PlayGame` subroutine instead.

To test that your change has worked

- run the **Skeleton Program**
- select option `J` and jump to square `15`
- select option `J` and jump to square `15`

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

0 8 . 1 Your amended PROGRAM SOURCE CODE for the new subroutine `MoveBackIfSameSquare` and the amended subroutine `PlayGame`.

[12 marks]

0 8 . 2 SCREEN CAPTURE(S) that show the two players landing on square 15 and that after this one player is moved to square 0 whilst the other remains on square 15.

[2 marks]

0 9 The snakes and ladders are currently placed at fixed positions on the board.

The **Skeleton Program** should be changed so that the five ladders are placed at random positions on the board and are of random lengths.

The snakes should stay in their current fixed positions.

When placing the ladders:

- A ladder can start at any square from 1 to 89.
- A ladder should be of a random length between 10 and 40 squares.
- Both ends of a ladder must be on the board. For example, a ladder starting at square 87 could not be 20 squares long as this would lead to square 107, which is not on the board. The maximum length of a ladder starting at square 87 would be 12.
- A ladder should not start in a square that already contains a snake or a ladder, so before placing a new ladder you should check that its start square is empty and choose a different location for it if the square is not empty.

Modify the `PlaceSnakesAndLadders` subroutine so that the ladders are placed at random positions on the board.

Test that your changes have worked by running the program and checking that the list of snakes and ladders displayed at the start of the game fulfils as many of the requirements of the task as you can.

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

0 9 . 1 Your amended PROGRAM SOURCE CODE for the subroutine `PlaceSnakesAndLadders`.

[14 marks]

0 9 . 2 SCREEN CAPTURE(S) that show the list of ladder positions on the board.

[2 marks]

END OF QUESTIONS