SECTION B

You are advised to spend no more than **15 minutes** on this section.

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

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

Question 8 The algorithm, represented using pseudo-code in **Figure 5**, and the variable table, **Table 4**, describe a simple two-player game.

Player One chooses a whole number greater than or equal to 1 and then Player Two tries to guess the number chosen by Player One. Player Two gets up to three attempts to guess the number. Player Two wins the game if they correctly guess the number, otherwise Player One wins the game.

Note that in **Figure 5**, the symbol ≠ means 'is not equal to'.

Figure 5

```
NumberToGuess ← 0
WHILE NumberToGuess < 1
  OUTPUT "Player One enter a number: "
  INPUT NumberToGuess
ENDWHILE
Guess ← 0
GuessCount ← 0
WHILE Guess ≠ NumberToGuess AND GuessCount < 3
  OUTPUT "Player Two have a guess: "
  INPUT Guess
  GuessCount ← GuessCount + 1
ENDWHILE
IF Guess = NumberToGuess
  THEN OUTPUT "Player Two wins"
  ELSE OUTPUT "Player One wins"
ENDIF
```

Table 4

Identifier	Data type	Purpose
NumberToGuess	Integer	Stores the number entered by Player One
GuessCount	Integer	Stores the number of guesses that Player Two has made so far
Guess	Integer	Stores the most recent guess made by Player Two

What you need to do

Write a program for the above algorithm.

Test the program by conducting **Test 1** and **Test 2**.

Test 1
Test that your program works correctly by conducting the following test:

- Player One enters the number `0`
- Player One enters the number `5`
- Player Two enters a guess of `5`

Test 2
Test that your program works correctly by conducting the following test:

- Player One enters the number `6`
- Player Two enters guesses of `1`, `3`, `7`

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

2 1 Your PROGRAM SOURCE CODE.
[12 marks]

2 2 SCREEN CAPTURE(S) showing the result of **Test 1**.
[2 marks]

2 3 SCREEN CAPTURE(S) showing the result of **Test 2**.
[1 mark]

Turn over for SECTION C

SECTION D

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

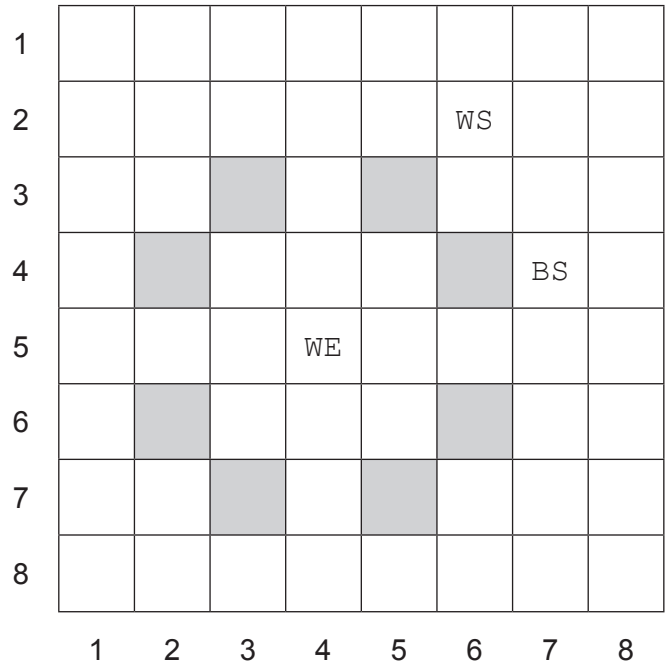
Type your answers to **Section D** in your Electronic Answer Document.
You **must save** this document at regular intervals.

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

Question 10 This question refers to the subroutine `CheckEtluMoveIsLegal`.

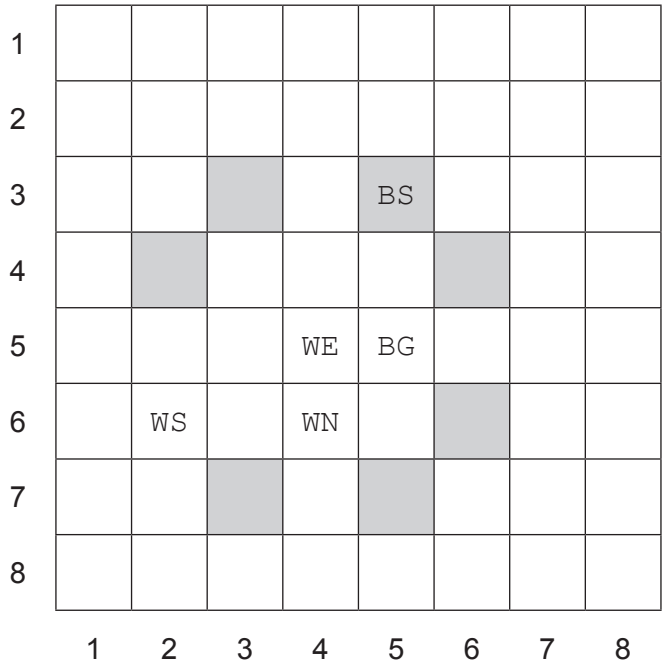
The rules describing how an etlu can move are to be changed so that an etlu can move as described in **Figure 6** and **Figure 7**.

Figure 6



White's etlu can move two squares in a vertical direction plus one in a horizontal direction or two squares in a horizontal direction plus one in a vertical direction. The shaded squares show where the etlu can move to.

Figure 7



White's etlu **cannot** move into the same square as another white piece but can move into the same square as a black piece (the black piece will be captured). An etlu can jump over other pieces of either colour. The shaded squares show where the etlu can move to.

Adapt the subroutine `CheckEtluMoveIsLegal` so that an etlu can move as described in **Figure 6** and **Figure 7**.

Test that the changes you have made work:

- run the **Skeleton Program**
- enter `N` when asked if you want to play the sample game
- enter a start square of `28` and a finish square of `36`
- enter a start square of `12` and a finish square of `13`
- enter a start square of `36` and a finish square of `16`
- enter a start square of `36` and a finish square of `15`.

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

3 4 Your PROGRAM SOURCE CODE for the amended subroutine `CheckEtluMoveIsLegal`.

[2 marks]

3 5 SCREEN CAPTURE(S) showing the results of each of the moves in the required test.

[2 marks]

Question 11 This question refers to the subroutine `CheckMoveIsLegal`.

When the user has entered the start square and the finish square for their move a number of checks are made to see if their intended move is legal.

Add a validation check to the subroutine `CheckMoveIsLegal` so that a move is only accepted as being legal if the **start square** refers to a square that is on the board.

Test that the changes you have made work:

- run the **Skeleton Program**
- enter `N` when asked if you want to play the sample game
- enter a start square of `89` and a finish square of `44`
- enter a start square of `98` and a finish square of `44`
- enter a start square of `87` and a finish square of `86`.

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

3 6 Your PROGRAM SOURCE CODE for the amended subroutine `CheckMoveIsLegal`.

[4 marks]

3 7 SCREEN CAPTURE(S) showing the requested test. You must make sure that evidence for all parts of the requested test by the user is provided in the SCREEN CAPTURE(S).

[2 marks]

There are no questions printed on this page

Question 12 starts on the next page

Question 12 This question refers to the subroutine `MakeMove`.

When a redum is promoted the player will be offered a choice of which piece they would like the redum to become (it will not automatically become a marzaz pani).

Task 1
Adapt the program source code for the `MakeMove` subroutine so that if a redum of either colour is promoted the message "`Enter a type of piece: `" is displayed.

The value entered by the user should then be stored in a variable called `PieceType`. The user will enter a single letter to indicate their choice of piece, eg they will enter an `E` if they want to replace the redum with an etlu.

Instead of becoming a marzaz pani the redum should become the type of piece chosen by the player.

There is no need to check that the player has chosen a valid type of piece.

Test 1
Test that your adapted program source code works:

- run the **Skeleton Program**
- enter `Y` when asked if you want to play the sample game
- enter a start square of `12` and a finish square of `11`
- when asked to choose a piece enter the character `G`.

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

3	8	Your PROGRAM SOURCE CODE for the amended subroutine `MakeMove`. **[4 marks]**

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

Task 2
Add a validation check to the subroutine `MakeMove` so that it repeatedly attempts to get the type of piece chosen by the player until a `G`, `E`, `N` or `M` is entered.

Each time an invalid value is entered the message "`Not a valid type of piece, please try again.`" should be displayed.

You **only** need to adapt the program source code so that the type of piece chosen by the player with the **White** pieces is validated in the way described.

Test 2
Test that the changes you have made work:

- run the **Skeleton Program**
- enter `Y` when asked if you want to play the sample game
- enter a start square of `12` and a finish square of `11`
- when asked to choose a piece enter the character `Z`
- when asked to choose a piece enter the character `N`.

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

4 0	Your PROGRAM SOURCE CODE for the amended subroutine `MakeMove`. **[5 marks]**
4 1	SCREEN CAPTURE(S) showing the results of **Test 2**. **[2 marks]**

Turn over for the next question

Question 13 This question will further extend the functionality of the **Skeleton Program**.

In the future the game will be adapted so that a human player can choose to play against the computer. The computer will act as an artificial intelligence (AI) opponent. The AI will need to be able to select which of the available legal moves it should play and to do this it will use a fitness function.

A fitness function evaluates how "good" the position resulting from a move is. The AI will apply the fitness function to the positions resulting from each of the possible moves to obtain a score for each possible position. It will then choose to play the move that leads to the position with the best score according to the fitness function.

In this question you will create a subroutine that can evaluate how 'good' a position is. You do **not** need to create an AI opponent.

Task 1
Create a subroutine called `GetFitness` which takes one parameter (the board) and creates an **Overall Score** for a position in a game of CAPTURE THE SARRUM using the scoring method described in **Figure 8**. It should return this **Overall Score** to the calling routine. You may choose whether to make the new subroutine a function or a procedure.

You are likely to get some marks for this task even if your subroutine is only partially working.

Figure 8

The **Overall Score** for a position is obtained by:

- calculating a score for the player with the white pieces (**Score One**)
- calculating a score for the player with the black pieces (**Score Two**)
- subtracting **Score Two** from **Score One** to get the **Overall Score**

Each of the different pieces has a numeric value associated with it indicating how much the piece is worth:

- a redum has a value of 1
- a marzaz pani has a value of 2
- a nabu has a value of 2
- an etlu has a value of 3
- a gisgigir has a value of 6
- a sarrum has a value of 200

Score One is obtained by adding together the values of all the white pieces on the board.

Score Two is obtained by adding together the values of all the black pieces on the board.

Task 2
Adapt the MAIN PROGRAM BLOCK so that the **Overall Score** returned by a call to the `GetFitness` subroutine is displayed **before** asking the player to enter their move.

Task 3
Test your program works by conducting the following test:

- run the **Skeleton Program**
- enter Y when asked if you want to play the sample game.

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

4 2	Explain why it is important that the value of the sarrum is greater than 6. **[1 mark]**
4 3	Your PROGRAM SOURCE CODE for the subroutine `GetFitness`. **[13 marks]**
4 4	Your PROGRAM SOURCE CODE for the amended MAIN PROGRAM BLOCK. **[2 marks]**
4 5	SCREEN CAPTURE(S) showing the requested test. **[1 mark]**

END OF QUESTIONS

There are no questions printed on this page