QUESTION 1.

1 A declarative language is used to represent the following facts and rules about a

```
01  feature(dog, drinks_milk).

02  feature(dog, has_lungs).

03  feature(horse, has_lungs).

04  feature(tuna, lives_in_water).

05  feature(tuna, has_gills).

06  feature(crab, lives_in_water).

07  mammal(drinks_milk).

08  mammal(has_lungs).

09  fish(lives_in_water).

10  fish(has_gills).

11  is_a_mammal(X) IF (feature(X, Y) AND mammal(Y)) AND (feature(X, Z)
    AND mammal(Z)).
```

These clauses are explained in the following table.

Clause	Explanation
01	A dog has the feature, drinks milk
07	A mammal drinks milk
11	X is a mammal, if: • X has the feature Y and a mammal has a feature Y, **and** • X has the feature Z and a mammal has the feature Z

(a) More facts are to be included.

(i) A bird has wings, and a bird lays eggs.

Write the additional clauses to record these facts.

12 ....

13 ....
[2]

(ii) An eagle has all the features of a bird.

Write the additional clauses to record this fact.

14 ....

15 ....
[2]

(b) **(i)** Using the variable `B`, the goal

```
feature(B, drinks_milk)
```

returns

```
B = dog
```

Write the result returned by the goal

```
feature(B, lives_in_water)
```

`B =` .... [2]

(ii) Write a goal, using the variable `C`, to find the feature(s) of tuna.

.... [2]

(c) An animal is a bird if it lays eggs **and** it has wings.

Complete the following rule.

`is_a_bird(X) IF` ....

.... [3]

(d) Declarative programming and object-oriented programming are two examples of programming paradigms.

(i) Define the term **programming paradigm**.

....

.... [1]

(ii) Give **two** examples of programming paradigms, other than declarative and object-oriented programming.

1 ....

2 ....

[2]

QUESTION 2.

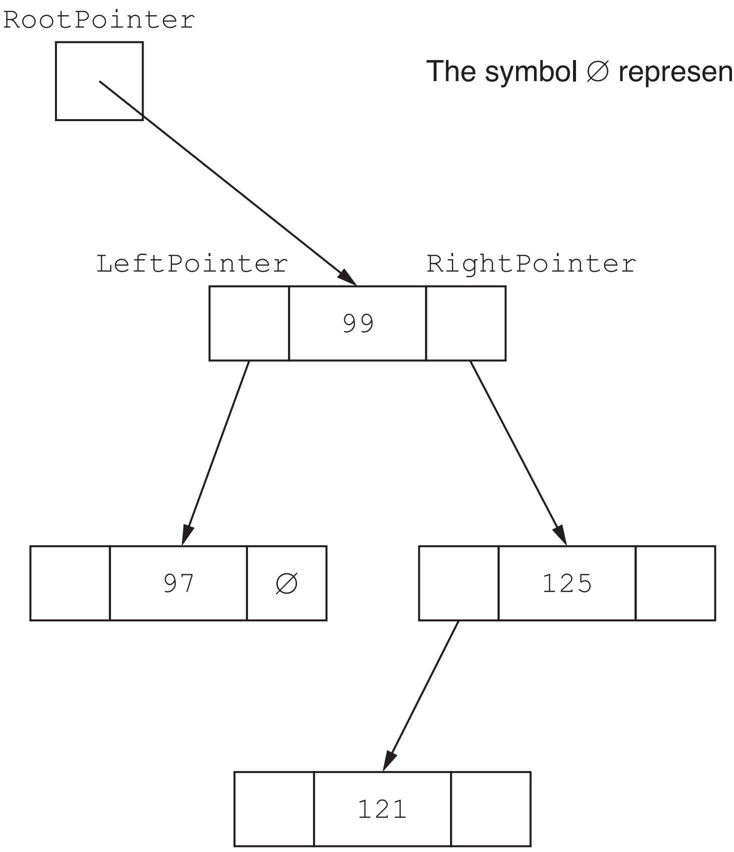
2 A computer games club wants to run a competition. The club needs a system to achieved in the competition.

A selection of score data is as follows:

99, 125, 121, 97, 109, 95, 135, 149

(a) A linked list of nodes will be used to store the data. Each node consists of the data, a left pointer and a right pointer. The linked list will be organised as a binary tree.

(i) Complete the binary tree to show how the score data above will be organised.



The symbol ∅ represents a null pointer.

[5]

(ii) The following diagram shows a 2D array that stores the nodes of the bi list.

Add the correct pointer values to complete the diagram, using your answ **part (a)(i)**.

RootPointer

0

FreePointer

Index	LeftPointer	Data	RightPointer
0		99	
1		125	
2		121	
3		97	
4		109	
5		95	
6		135	
7		149	
8			

[6]

(b) The club also considers storing the data in the order in which it receives linked list in a 1D array of records.

The following pseudocode algorithm searches for an element in the linked list.

Complete the **six** missing sections in the algorithm.

```
FUNCTION FindElement(Item : INTEGER) RETURNS ..........................................

   .......................................... ← RootPointer

   WHILE CurrentPointer .......................................... NullPointer

      IF List[CurrentPointer].Data <> ..........................................

         THEN

            CurrentPointer ← List[..........................................].Pointer

         ELSE

            RETURN CurrentPointer

      ENDIF

   ENDWHILE

   CurrentPointer ← NullPointer

   .......................................... CurrentPointer

ENDFUNCTION
```

[6]

(c) The games club is looking at two programming paradigms: imperative and programming paradigms.

Describe what is meant by the **imperative programming paradigm** and the **object-** **programming paradigm**.

(i) Imperative ....

....

....

....

....

.... [3]

(ii) Object-oriented ....

....

....

....

....

.... [3]

(d) Players complete one game to place them into a category for the competition. wants to implement a program to place players into the correct category. The has decided to use object-oriented programming (OOP).

The highest score that can be achieved in the game is 150. Any score less than 50 w qualify for the competition. Players will be placed in a category based on their score.

The following diagram shows the design for the class `Player`. This includes the properties and methods.

```
                              Player
--------------------------------------------------------------------------
Score    : INTEGER // initialised to 0
Category : STRING  // "Beginner", "Intermediate",
                   // "Advanced" or "Not Qualified", initialised
                   // to "Not Qualified"
PlayerID : STRING  // initialised with the parameter InputPlayerID
--------------------------------------------------------------------------
Create()          // method to create and initialise an object using
                  // language-appropriate constructor
SetScore()        // checks that the Score parameter has a valid value
                  // if so, assigns it to Score
SetCategory()     // sets Category based on player's Score
SetPlayerID()     // allows a player to change their PlayerID
                  // validates the new PlayerID
GetScore()        // returns Score
GetCategory()     // returns Category
GetPlayerID()     // returns PlayerID
```

(i) The constructor receives the parameter `InputPlayerID` to create

Other properties are initialised as instructed in the class diagram.

Write **program code** for the `Create()` constructor method.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

.... [5]

(ii) Write **program code** for the following **three** get methods.

Programming language ....

`GetScore()`

Program code

....

....

....

....

`GetCategory()`

Program code

....

....

....

....

`GetPlayerID()`

Program code

....

....

....

....

[4]

(iii) The method `SetPlayerID()` asks the user to input the new player ID a value.

It checks that the length of the `PlayerID` is less than or equal to 15 charac greater than or equal to 4 characters. If the input is valid, it sets this as the `Play` otherwise it loops until the player inputs a valid `PlayerID`.

Use suitable input and output messages.

Write **program code** for `SetPlayerID()`.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

.... [4]

(iv) The method `SetScore()` checks that its `INTEGER` parameter `ScoreI` it is valid, it is then set as `Score`. A valid `ScoreInput` is greater than or e less than or equal to 150.

If the `ScoreInput` is valid, the method sets `Score` and returns `TRUE`.

If the `ScoreInput` is not valid, the method does not set `Score`, displays an erro message, and it returns `FALSE`.

Write **program code** for `SetScore(ScoreInput : INTEGER)`.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [5]

(v) Write **program code** for the method `SetCategory()`. Use the properti in the original class definition.

Players will be placed in one of the following categories.

Category	Criteria
Advanced	Score is greater than 120
Intermediate	Score is greater than 80 and less than or equal to 120
Beginner	Score is greater than or equal to 50 and less than or equal to 80
Not Qualified	Score is less than 50

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [4]

(vi) Joanne has played the first game to place her in a category for the comp

The procedure `CreatePlayer()` performs the following tasks.

- allows the player ID and score to be input with suitable prompts
- creates an instance of `Player` with the identifier `JoannePlayer`
- sets the score for the object
- sets the category for the object
- outputs the category for the object

Write **program code** for the `CreatePlayer()` procedure.

Programming language ....

Program code

....

....

....

....

....

....

....

....

....

....

....

....

....

....

.... [8]

(e) The programmer wants to test that the correct category is set for a player's s

As stated in **part (d)(v)**, players will be placed in one of the following categories.

Category	**Criteria**
Advanced	Score is greater than 120
Intermediate	Score is greater than 80 and less than or equal to 120
Beginner	Score is greater than or equal to 50 and less than or equal to 80
Not Qualified	Score is less than 50

Complete the table to provide test data for each category.

Category	**Type of test data**	**Example test data**
Beginner	Normal	
	Abnormal	
	Boundary	
Intermediate	Normal	
	Abnormal	
	Boundary	
Advanced	Normal	
	Abnormal	
	Boundary	

[3]

(f) In **part (b)**, the club stored scores in a 1D array. This allows the club to sort t

The following is a sorting algorithm in pseudocode.

```
NumberOfScores ← 5
FOR Item ← 1 TO NumberOfScores – 1
    InsertScore ← ArrayData[Item]
    Index ← Item – 1
    WHILE (ArrayData[Index] > InsertScore) AND (Index >= 0)
        ArrayData[Index + 1] ← ArrayData[Index]
        Index ← Index – 1
    ENDWHILE
    ArrayData[Index + 1] ← InsertScore
ENDFOR
```

(i) Give the name of this algorithm.

.... [1]

(ii) State the name of **one** other sorting algorithm.

.... [1]

(iii) Complete a dry run of the algorithm using the following trace table.

Item	NumberOfScores	InsertScore	Index	ArrayData				
				0	1	2	3	
				99	125	121	109	115

[7]