5 Complete the table by filling in the missing object-oriented programming (OOP) terms and descriptions.

OOP term	Description
.....	A method that accesses the value of a property.
.....	A method that changes the value of a property.
Object	....
Method	....

[4]

3 A program stores data about animals using Object-Oriented Programming (OOP).

The class `Animal` stores the data about animals.

Animal	
`Name : String`	stores the name of the animal
`Sound : String`	stores the sound the animal makes
`Size : Integer`	stores the size of the animal as an integer between 1 (smallest) and 10 (largest)
`Intelligence : Integer`	stores the intelligence of the animal as an integer between 1 (least) and 10 (most)
`Constructor()`	initialises all attributes to its parameter values
`Description()`	returns a string message that contains the data from the attributes

(a) (i) Write program code to declare the class `Animal` and its constructor.

Do **not** declare the other methods.

Use your programming language appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program as **Question3_J25**.

Copy and paste the program code into part **3(a)(i)** in the evidence document.

[4]

(ii) The method `Description()` creates and returns a string of the animal's data in the format:

```
"The animal's name is " <Name> ", it makes a " <Sound> ", its size
is " <Size> " and its intelligence level is " <Intelligence>
```

For example:

```
The animal's name is Teddy, it makes a Bark, its size is 4 and its
intelligence level is 6
```

Write program code for `Description()`

Save your program.

Copy and paste the program code into part **3(a)(ii)** in the evidence document.

[3]

(b) The class `Parrot` inherits from the class `Animal`

`Parrot` inherits the attributes from `Animal` and overrides the `Description()` method. The additional attributes and methods in the class `Parrot` are:

Parrot	
`WingSpan : Integer`	the width of the parrot's wings to the nearest cm
`NumberWords : Integer`	the number of words the parrot can speak
`Constructor()`	calls the parent constructor using its parameter values; initialises `WingSpan` and `NumberWords` to its parameter values
`ChangeNumberWords()`	takes an integer parameter and adds this to the number currently in `NumberWords`

(i) Write program code to declare the class `Parrot`, its constructor and the method `ChangeNumberWords()`

Use your programming language appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into part **3(b)(i)** in the evidence document.

[4]

(ii) The method `Description()` in the class `Parrot` creates and returns a string of the animal's data in the format:

```
"The animal's name is " <Name> ", it makes a " <Sound> ", its size
is " <Size> " and its intelligence level is " <Intelligence> ". It
has a wingspan of " <WingSpan> "cm and can say " <NumberWords> "
words."
```

For example:

```
The animal's name is Chewie, it makes a Squawk, its size is 1 and
its intelligence level is 10. It has a wingspan of 30cm and can
say 29 words.
```

Write program code for `Description()`

Save your program.

Copy and paste the program code into part **3(b)(ii)** in the evidence document.

[2]

(c) The class `Wolf` inherits from the class `Animal`

`Wolf` inherits the attributes from `Animal` and overrides the `Description()` method. The additional attributes and methods in the class `Wolf` are:

Wolf	
`TerritorySize : Integer`	stores the size of the area where the wolf lives, to the nearest square mile
`Constructor()`	calls the parent constructor using its parameter values; initialises `TerritorySize` to its parameter value
`SetTerritorySize()`	takes an integer parameter and adds this to the number currently in `TerritorySize`

(i) Write program code to declare the class `Wolf`, its constructor and the method `SetTerritorySize()`

Use your programming language appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into part **3(c)(i)** in the evidence document.

[4]

(ii) The method `Description()` in the class `Wolf` creates and returns a string of the animal's data in the format:

```
"The animal's name is " <Name> ", it makes a " <Sound> ", its size
is " <Size> " and its intelligence level is " <Intelligence> ".
Its territory is " <TerritorySize> " square miles."
```

For example:

```
The animal's name is Nighteyes, it makes a Howl, its size is 8 and
its intelligence level is 7. Its territory is 100 square miles.
```

Write program code for `Description()`

Save your program.

Copy and paste the program code into part **3(c)(ii)** in the evidence document.

[2]

(d) (i) The main program declares instances of the classes for **three** animals:

- A parrot with the name 'Chewie'; it makes a 'Squawk' sound. Its size is 1, intelligence is 10, wingspan is 30 cm and it can say 29 words.
- A wolf with the name 'Nighteyes'; it makes a 'Howl' sound. Its size is 8, intelligence is 7 and its territory is 100 square miles.
- An animal that is a horse with the name 'Copper'; it makes a 'Neigh' sound. Its size is 10 and its intelligence is 6.

Write program code for the main program.

Save your program.

Copy and paste the program code into part **3(d)(i)** in the evidence document.

[2]

(ii) The main program also needs to:

- decrease the territory for the wolf Nighteyes by 20 square miles
- increase the number of words the parrot Chewie can say by 2 words
- output the description for all **three** animals.

Write program code to extend the main program.

Save your program.

Copy and paste the program code into part **3(d)(ii)** in the evidence document.

[3]

(iii) Test your program.

Take a screenshot of the output(s).

Save your program.

Copy and paste the screenshot into part **3(d)(iii)** in the evidence document.

[2]

3 A program stores data in a linked list that is designed using Object-Oriented Programming (OOP).

The class `Node` stores data about the nodes.

Node	
`TheData : Integer`	stores the integer data
`NextNode : Node`	stores the next node in the linked list
`Constructor()`	initialises `TheData` to its parameter value, initialises `NextNode` to a null value
`GetData()`	returns `TheData`
`GetNextNode()`	returns `NextNode`
`SetNextNode()`	takes an object of type `Node` as a parameter and stores it in `NextNode`

(a) (i) Write program code to declare the class `Node` and its constructor.

Do **not** declare the other methods.

Use your programming language appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program as **Question3_J25**.

Copy and paste the program code into part **3(a)(i)** in the evidence document.

[4]

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

Save your program.

Copy and paste the program code into part **3(a)(ii)** in the evidence document.

[3]

(iii) The method `SetNextNode()` takes an object of type `Node` as a parameter. The method stores the parameter in the attribute `NextNode`

Write program code for `SetNextNode()`

Save your program.

Copy and paste the program code into part **3(a)(iii)** in the evidence document.

[2]

(b) The class `LinkedList` stores the linked list.

LinkedList	
`HeadNode : Node`	stores the first node in the linked list
`Constructor()`	initialises `HeadNode` to a null value
`InsertNode()`	creates a new node using its integer parameter. Sets this node as the new head node and updates `NextNode`
`RemoveNode()`	finds the first node that contains its integer parameter and removes this node from the linked list
`Traverse()`	concatenates and returns the integer data in each node in the linked list

(i) Write program code to declare the class `LinkedList` and its constructor.

Do **not** declare the other methods.

Use your programming language appropriate constructor.

If you are writing in Python, include attribute declarations using comments.

Save your program.

Copy and paste the program code into part **3(b)(i)** in the evidence document.

[2]

(ii) The method `InsertNode()`:

- takes an integer as a parameter
- creates a new node with the parameter as the integer data
- uses the new node's method `SetNextNode()` to store the current `HeadNode` as the next node
- replaces `HeadNode` with the current node.

Write program code for `InsertNode()`

Save your program.

Copy and paste the program code into part **3(b)(ii)** in the evidence document.

[4]

(iii) The method `Traverse()` concatenates the integer data from the nodes in the linked list, starting with the node stored in `HeadNode`. The method returns the final string with each integer data separated with a space.

Write program code for `Traverse()`

Save your program.

Copy and paste the program code into part **3(b)(iii)** in the evidence document.

[3]

(iv) The method `RemoveNode()` takes an integer parameter to search for and remove from the linked list.

The method first checks if the linked list is empty. If the linked list is empty the method returns `FALSE`

If the linked list is **not** empty, the integer data in `HeadNode` is compared to the parameter. If it matches the parameter, `HeadNode` is changed to store the next node.

If the parameter does **not** match, the nodes are followed until either:

- the node with matching integer data is found. This node is removed, the appropriate nodes updated and `TRUE` returned
  or
- none of the nodes contain matching integer data to the parameter. No nodes are removed and `FALSE` is returned.

Write program code for `RemoveNode()`

Save your program.

Copy and paste the program code into part **3(b)(iv)** in the evidence document.

[6]

(c) (i) The main program creates a new `LinkedList` object and uses the appropriate method to insert **five** nodes with the following integer data values in the order given:

```
10      20      30      40      50
```

The main program then:

- calls `Traverse()`
- removes the node containing the integer data `30` using `RemoveNode()`
- calls `Traverse()`

Write program code for the main program.

Save your program.

Copy and paste the program code into part **3(c)(i)** in the evidence document.

[3]

(ii) Test your program.

Take a screenshot of the output(s).

Save your program.

Copy and paste the screenshot into part **3(c)(ii)** in the evidence document.

[2]

10 Objects and classes form the basic structure of Object-Oriented Programming (OOP).

(a) Outline the structure of a class.

.....

.....

.....

.....

.....

..... [3]

(b) Give **three** differences between an object and a class.

1 .....

.....

.....

2 .....

.....

.....

3 .....

.....

.....

[3]

9 A veterinary surgery wants to create a class for individual pets.
Some of the attributes required in the class are listed in the table.

Attribute	Data type	Description
`PetID`	`STRING`	unique ID assigned at registration
`PetType`	`STRING`	type of pet assigned at registration
`OwnerTelephone`	`STRING`	telephone number of owner assigned at registration
`DateRegistered`	`DATE`	date of registration

(a) State **one** reason why the attributes would be declared as `PRIVATE`.

.....

..... [1]

(b) Complete the class diagram for `Pet`, to include:

- an attribute and data type for the name of the pet
- an attribute and data type for the name of the owner
- a method to create a `Pet` object and set attributes at the time of registration
- a method to assign a pet ID
- a method to assign the date of registration
- a method to return the pet name
- a method to return the owner's telephone number.

Pet
`PetID` : `STRING` `PetType` : `STRING` `OwnerTelephone` : `STRING` `DateRegistered` : `DATE` ..... : .... : .....
....

[5]