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]