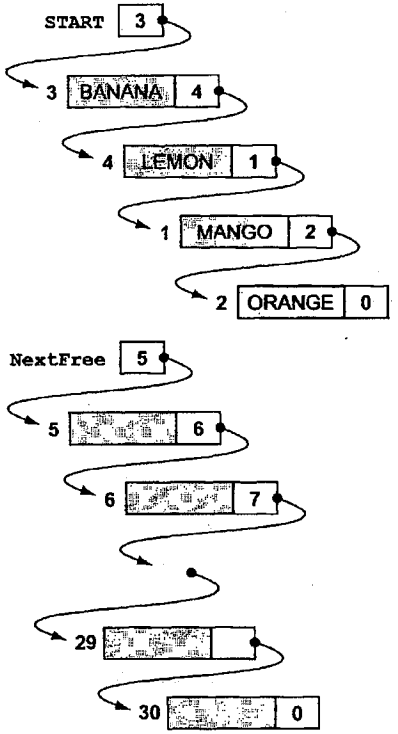
PROGRAMMING CHALLENGE 3: FRUITS LINKED LIST

A program is to be written to represent and implement a linked list of nodes. Each node contains a string data value and a pointer. The pointers link the data items in alphabetical order.

The unused nodes are linked as shown below. The first unused node is the position where the next new data item is to be stored.



The diagram shows the linked list with:

- the items MANGO, ORANGE, BANANA and LEMON (added in that order).
- the unused nodes linked together.

Each node is implemented as an instance of the class **ListNode**. The class **ListNode** has the following properties:

Class: **ListNode**		
Properties		
Identifier	**Data Type**	**Description**
DataValue	**STRING**	The node data
PointerValue	**INTEGER**	The node pointer

A linked list is implemented as an instance of the class **LinkedList**. The class **LinkedList** has the following properties and methods:

Class: **LinkedList**		
Properties		
Identifier	**Data Type**	**Description**
Node	**ARRAY [30] OF ListNode**	The linked list data structure - data values and pointers. The array index starts at 1. For testing purposes the dataset has a maximum of 30 items.
Start	**INTEGER**	Index position of the node at the start of the linked list
NextFree	**INTEGER**	Index position of the next unused node
Methods		
Identifier		**Description**
Initialise	**PROCEDURE**	Sets all the node data values to 'empty string'. Set pointers to indicate all nodes are unused and linked. Initialise values for **Start** and **NextFree**.
AddNode	**PROCEDURE**	Add a new data item to the linked list.
RemoveNode	**PROCEDURE**	Remove data item from the linked list.
Traversal	**PROCEDURE**	Display the data items in order.
ReverseTraversal	**PROCEDURE**	Display the data items in reverse order.
DisplayLinkedList	**PROCEDURE**	Display the current state of pointers and the array contents.
IsEmpty	**FUNCTION RETURNS BOOLEAN**	Tests for empty linked list.
IsFull	**FUNCTION RETURNS BOOLEAN**	Tests for no unused nodes.

Task 1

Write program code that repeatedly:

- displays a menu with the following choices:
  1. Add an item
  2. Traverse the linked list of used nodes and output the data values
  3. Output all pointers and data values
  X. Exit
- calls an appropriate procedure depending on the user's choice.

Evidence 1

Program code for Task 1. **[5]**

Task 2

Write program code for the classes **ListNode** and **LinkedList** including the **IsEmpty** method.
The code should follow the specification given.
Do not attempt to write the methods **AddNode, RemoveNode, Traversal, ReverseTraversal** or **IsFull** at this stage.

Evidence 2

Program code for the **ListNode** and **LinkedList** classes (Task 2). **[12]**

Task 3

Write code to create a **LinkedList** object in the main program.
Run the program and select menu choice 3 to confirm the initial values of the pointers and data values when the linked list is empty.

Evidence 3

Screenshot confirming all values after initialisation of the **LinkedList** object (Task 3). **[3]**

Task 4

Consider the **AddNode** method. The following algorithm will add a new data item to the linked list.
The algorithm uses the variables below:

Identifier	Data Type	Description
NewItem	**STRING**	New data item input by the user
Found	**BOOLEAN**	Flags to TRUE when the position at which to insert the new item has been found
Current	**INTEGER**	Current array index position during list traversal
Previous	**INTEGER**	Previous array index position during list traversal
Temp	**INTEGER**	Temporary storage of pointer value


```
PROCEDURE AddNode
    INPUT NewItem
    Node[NextFree].DataValue ← NewItem

    IF Start = 0
        THEN
            Start ← NextFree
            Temp ← Node[NextFree].PointerValue
            Node[NextFree].PointerValue ← 0
            NextFree ← Temp
        ELSE
            //traverse the list - starting at Start to find
            //the position at which to insert the new item
            Temp ← Node[NextFree].PointerValue

            IF NewItem < Node[Start].DataValue
                THEN
                    // new item will become the start of the list
                    Node[NextFree].PointerValue ← Start
                    Start ← NextFree
                    NextFree ← Temp
                ELSE
                    // the new item is not at the start of the list
                    Previous ← 0
                    Current ← Start
                    Found ← False
                    REPEAT
                        IF NewItem <= Node[Current].DataValue
                            THEN
                                Node[Previous].PointerValue ← NextFree
                                Node[NextFree].PointerValue ← Current
                                NextFree ← Temp
                                Found ← True
                            ELSE
                                // move on to the next node
                                Previous ← Current
                                Current ← Node[Current].PointerValue
                        ENDIF
                    UNTIL Found = True OR Current = 0
                    IF Current = 0
                        THEN
                            Node[Previous].PointerValue ← NextFree
                            Node[NextFree].PointerValue ← 0
                            NextFree ← Temp
                    ENDIF
            ENDIF
    ENDIF
ENDPROCEDURE
```

Write code to implement for the **LinkedList** class:

- the **AddNode** method
- the **IsFull** method.

The main program should check each time that the **LinkedList** object is not full before using the **AddNode** method.

Run the program as follows:

- Menu choice 1 four times, inputting the data values:
  MANGO, ORANGE, BANANA, LEMON in that order.
- Menu choice 3 to display.

Evidence 4
Program code for method **AddNode**. **[8]**

Evidence 5
Screenshot showing the pointers and the addition of the four nodes to the linked list. **[3]**

Task 5
Write program code to implement the **LinkedList** class method **Traversal** by calling the **TraversalInOrder** procedure given below.

```
PROCEDURE TraversalInOrder(Index)
    IF Index <> 0
        THEN
            OUTPUT Node[Index].DataValue
            // follow the pointer to the next data item in the linked list
            TraversalInOrder(Node[Index].PointerValue)
    ENDIF
ENDPROCEDURE
```

Evidence 6
Program code for procedures **Traversal** and **TraversalInOrder**. **[2]**

Task 6
Run the program as follows:

- Menu choice 1 four times, inputting the data values:
  MANGO, ORANGE, BANANA, LEMON in that order.
- Menu choice 2 to display.

Evidence 7
Screenshot showing the program execution to test the **Traversal** method. **[2]**

Task 7
Make a copy of the **TraversalInOrder** and **Traversal** procedures.
Paste to form two new procedures **TraversalInReverseOrder** and **ReverseTraversal**.

Make the necessary changes/additions to these procedures in order that the data items are output in reverse order by calling the new method **ReverseTraversal**.
Run the program code from a new menu choice 4.
Test the method using the four items given in Task 6.

Evidence 8
Program code for the new procedures. **[2]**

Evidence 9
Screenshot showing option 4 selected and the resulting output. **[1]**

Task 8
Write code to implement the **RemoveNode** method which removes a node from the linked list.

Evidence 10
Present the variables used in your **RemoveNode** algorithm in a table like this: **[3]**

Identifier	Data Type	Description
Item	**STRING**	Data item input by the user
...	...	...
...	...	...
...	...	...

Evidence 11
Your program code for **RemoveNode** method. **[9]**

Task 9
Add to the main program the code to check that the **LinkedList** object is not empty before using the **RemoveNode** method.
Run the program code from a new menu choice 5.

Continue to run the program from task 6 with the following menu choices:

- Menu choice 3 to display,
- Menu choice 5, inputting the data values: ORANGE, LEMON in that order,
- Menu choice 3 to display.

Evidence 12
Program code for the new procedures. **[2]**

Evidence 13
Screenshot showing option 5 selected and the resulting output. **[1]**