Practical No 2

Aim : Write A Program To Implement Linked List

```
# Making The Class Node
class Node:
    # Initalizing The Class
    def __init__(self,data,nextNode=None):
        self.data = data
        self.nextNode = nextNode

    # Getting The Data
    def getData(self):
        return self.data

    # Setting The Data
    def setData(self,data):
        self.data = data

    # Getting The Next Node
    def getNextNode(self):
        return self.nextNode

    # Setting The Next Node
    def setNextNode(self,reference):
        self.nextNode = reference

# Making The Linked List Class
class LinkedList:
    # Initalizing The Class
    def __init__(self,head = None):
        self.head = head
        self.size = 0

    # Getting The Size Of The Linked List
    def getSize(self):
        return self.size

    # Adding A Node In A Linked List
    def addNode(self,data):
        newNode = Node(data,self.head)
        self.head = newNode
        self.size += 1
        return True

    # Adding A Node In A Certain Position In A Linked List
    def addNodeAtPos(self, data, pos):
        if pos < 0:
            print("Position cannot be negative")
            return False
```

```
    newNode = Node(data)
    if pos == 0:
        newNode.nextNode = self.head
        self.head = newNode
    else:
        currentNode = self.head
        currentPosition = 0
        while currentNode and currentPosition < pos - 1:
            currentNode = currentNode.nextNode
            currentPosition += 1

        if currentNode is None:
            print("Position is out of range")
            return False
        newNode.nextNode = currentNode.nextNode
        currentNode.nextNode = newNode
        self.size += 1
        return True

# Printing The Node
def printNode(self):
    curr = self.head
    while curr:
        print(curr.data,end=" -> ")
        curr = curr.getNextNode()
    print("None")

# Deleting A Node
def deleteNode(self, key):
    temp = self.head

    # Case 1: The head node itself holds the key
    if temp is not None and temp.data == key:
        self.head = temp.nextNode  # Change the head
        temp = None  # Free memory
        self.size -= 1
        return  # Exit after deletion

    # Case 2: Traverse the list to find the key
    prev = None
    while temp is not None:
        if temp.data == key:
            prev.nextNode = temp.nextNode  # Unlink the node
            temp = None  # Free memory
            self.size -= 1
            return  # Exit after deletion
        prev = temp
        temp = temp.nextNode

    # Case 3: Key not found
    print(f"Key {key} not found in the list.")

# Maximum In A List
```

```
    def maximum(self):
        curr = self.head
        maxElement = curr.data
        while curr:
            if maxElement < curr.data:
                maxElement = curr.data
            curr = curr.getNextNode()
        return maxElement
    # Minimum In A List
    def minimum(self):
        curr = self.head
        minElement = curr.data
        while curr:
            if minElement > curr.data:
                minElement = curr.data
            curr = curr.getNextNode()
        return minElement

# Testing The Code
# Creating The Linked List
mylist = LinkedList()

# Adding Nodes / Elements
mylist.addNode(10)
mylist.addNode(40)
mylist.addNode(4)
mylist.addNode(14)
mylist.addNode(14)

# Adding A Node At A Certain Position
print("The Size Of The List Is Before Adding A Node At A Certain Position: ",mylist.getSize())
mylist.addNodeAtPos(55,3)
print("The Size Of The List Is After Adding A Node At A Certain Position: ",mylist.getSize())

mylist.addNode(24)
mylist.addNode(34)

# Printing The Linked List
print("The Linked List Is : ")
mylist.printNode()

# Maximum And Minimum
print("The Largest Number  In A List Is : ",mylist.maximum())
print("The Smallest Number  In A List Is : ",mylist.minimum())
print("The Difference In Largest And Smallest In A List : ",mylist.maximum()-mylist.minimum())

# Deleting A Node
print("The Size Of The List Is Before Deleting A Node : ",mylist.getSize())
mylist.deleteNode(14)
print("The Size Of The List Is After Deleting A Node : ",mylist.getSize())
mylist.printNode()
```

```
The Size Of The List Is Before Adding A Node At A Certain Position:  5
The Size Of The List Is After Adding A Node At A Certain Position:  6
The Linked List Is :
34 -> 24 -> 14 -> 14 -> 4 -> 55 -> 40 -> 10 -> None
The Largest Number  In A List Is :  55
The Smallest Number  In A List Is :  4
The Difference In Largest And Smallest In A List :  51
The Size Of The List Is Before Deleting A Node :  8
The Size Of The List Is After Deleting A Node :  7
34 -> 24 -> 14 -> 4 -> 55 -> 40 -> 10 -> None
```

Company Linked List

```
# Making The Class Node
class Node:
    # Initalizing The Class
    def __init__(self,customer,salesman,nextNode=None):
        self.data = {
            "customer":customer,
            "salesman":salesman
            }
        self.nextNode = nextNode

    # Getting The Data
    def getData(self):
        return self.data

    # Setting The Data
    def setData(self,data):
        self.data = data

    # Getting The Next Node
    def getNextNode(self):
        return self.nextNode

    # Setting The Next Node
    def setNextNode(self,reference):
        self.nextNode = reference

# Making The Linked List Class
class LinkedList:
    # Initalizing The Class
    def __init__(self,head = None):
        self.head = head
        self.size = 0

    # Getting The Size Of The Linked List
    def getSize(self):
        return self.size

    # Adding A Node In A Linked List
    def addCustomer(self,customer,salesman):
```

```
    newNode = Node(customer,salesman,self.head)
    self.head = newNode
    self.size += 1
    return True

# Adding A Node In A Certain Position In A Linked List
def addCustomerAtPos(self,customer,salesman,pos):
    newNode = Node(customer,salesman,pos)
    currentNode = self.head
    currentPosition = 0
    while True:
        if currentPosition == pos:
            previousNode.nextNode = newNode
            newNode.nextNode = currentNode
            self.size += 1
            return True
        else:
            previousNode = currentNode
            currentNode = currentNode.nextNode
            currentPosition += 1

# Printint The Node
def printCustomer(self):
    curr = self.head
    while curr:
        print(curr.data)
        curr = curr.getNextNode()

# Deleting A Customer
def deleteCustomer(self,key):
    temp = self.head
    if (temp is not None):
        if (temp.data["customer"] == key):
            self.head == temp.nextNode
            temp = None
            return
    while (temp is not None):
        if temp.data["customer"] == key:
            break
        prev = temp
        temp = temp.nextNode
    if (temp == None):
        return
    prev.nextNode = temp.nextNode
    temp = None
    self.size -= 1

# Find Customer By Salesman
def customerBySalesman(self,key):
    curr = self.head
    data = {
            "salesman":key,
            "customers":[]
```

```
            }
        while curr:
            if curr.data.get("salesman") == key:
                data["customers"].append(curr.data.get("customer"))
            curr = curr.getNextNode()

        return data

    # Find Salesman By Customer
    def salesmanrByCustomer(self,key):
        curr = self.head
        data = {
                "customers":key,
                "salesman":""
            }
        while curr:
            if curr.data.get("customer") == key:
                data["salesman"]=curr.data.get("salesman")
            curr = curr.getNextNode()

        return data

# Running The Code
myList = LinkedList()

myList.addCustomer("Arshad","Huzaifa")
myList.addCustomer("Arshad","Huzaifa")
myList.addCustomer("Ashif","Huzaifa")
myList.addCustomer("Harsh","Huzaifa")

print("All Customer With Salesman")
myList.printCustomer()

print("Find Customer By Salesman")
print(myList.customerBySalesman("Huzaifa"))

print("Find Salesman By Customer")
print(myList.salesmanrByCustomer("Ashif"))
```

```
All Customer With Salesman
{'customer': 'Harsh', 'salesman': 'Huzaifa'}
{'customer': 'Ashif', 'salesman': 'Huzaifa'}
{'customer': 'Arshad', 'salesman': 'Huzaifa'}
{'customer': 'Arshad', 'salesman': 'Huzaifa'}
Find Customer By Salesman
{'salesman': 'Huzaifa', 'customers': ['Harsh', 'Ashif', 'Arshad', 'Arshad']}
Find Salesman By Customer
{'customers': 'Ashif', 'salesman': 'Huzaifa'}
```