Practical No 2

Aim : Implementation Of 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
```

```
>>> myList = LinkedList()
>>> myList.addNode(12)
    True
>>> myList.addNode(120)
    True
>>> myList.addNode(120)
    True
>>> myList.addNode(100)
    True
>>> myList.addNode(90)
    True
```

 # Adding A Node In A Certain Position In A Linked List

```
def addNodeAtPos(self,data,pos):
    newNode = Node(data,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 printNode(self):
    curr = self.head
    while curr:
        print(curr.data)
        curr = curr.getNextNode()
```

 # Deleting A Node

```
def deleteNode(self,key):
    temp = self.head
    if (temp is not None):
        if (temp.data == key):
            self.head == temp.nextNode
            temp = None
            return
    while (temp is not None):
        if temp.data == key:
            break
        prev = temp
        temp = temp.nextNode
    if (temp == None):
        return
    prev.nextNode = temp.nextNode
    temp = None
    self.size -= 1
    return True
```

```
>>> myList.addNodeAtPos(23,5)
True
>>> myList.addNodeAtPos(3,6)
True
>>> myList.printNode()
90
900
90
100
120
23
3
120
12
```

```
>>> myList.deleteNode(100)
>>> myList.printNode()
90
900
90
120
23
3
120
12
```

Write a program with explanation for printing the following in a given linked list:
a. maximum
b. minimum
c. maximum – minimum

```
# 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
```

```
>>> print("The Maximum Number In A List : ",myList.maximum())
The Maximum Number In A List :  900
>>> print("The Minimum Number In A List : ",myList.minimum())
The Minimum Number In A List :  3
>>> print("The Maximum - Minimum : ",myList.maximum() - myList.minimum())
The Maximum - Minimum :  897
```