NAME: SAI DISHA .D
ROLL NO. 231057026

Implement a doubly linked lists.

I have done the implementation in python.

```
class Node:
    def __init__(self, value):
        self.previous = None
        self.data = value
        self.next = None


class DoublyLinkedList:
    def __init__(self):
        self.head = None

    def isEmpty(self):
        if self.head is None:
            return True
        return False

    def length(self):
        temp = self.head
        count = 0
        while temp is not None:
            temp = temp.next
            count += 1
        return count

    def search(self, value):
        temp = self.head
        isFound = False
        while temp is not None:
            if temp.data == value:
                isFound = True
                break
            temp = temp.next
        return isFound

    def insertAtBeginning(self, value):
        new_node = Node(value)
        if self.isEmpty():
            self.head = new_node
        else:
            new_node.next = self.head
            self.head.previous = new_node
            self.head = new_node

    def insertAtEnd(self, value):
        new_node = Node(value)
        if self.isEmpty():
            self.insertAtBeginning(value)
        else:
            temp = self.head
            while temp.next is not None:
                temp = temp.next
            temp.next = new_node
            new_node.previous = temp

    def insertAfterElement(self, value, element):
        temp = self.head
        while temp is not None:
            if temp.data == element:
                break
            temp = temp.next
        if temp is None:
            print("{} is not present in the linked list. {} cannot be inserted into the list.".format(element, value))
        else:
            new_node = Node(value)
            new_node.next = temp.next
            new_node.previous = temp
            temp.next.previous = new_node
            temp.next = new_node

    def insertAtPosition(self, value, position):
        temp = self.head
        count = 0
        while temp is not None:
            if count == position - 1:
                break
            count += 1
            temp = temp.next
        if position == 1:
            self.insertAtBeginning(value)
        elif temp is None:
            print("There are less than {}-1 elements in the linked list. Cannot insert at {} position.".format(position,
                                                                                                      position))
        elif temp.next is None:
            self.insertAtEnd(value)
        else:
            new_node = Node(value)
            new_node.next = temp.next
            new_node.previous = temp
            temp.next.previous = new_node
            temp.next = new_node

    def printLinkedList(self):
        temp = self.head
        while temp is not None:
            print(temp.data, sep=",")
            temp = temp.next

    def updateElement(self, old_value, new_value):
        temp = self.head
        isUpdated = False
        while temp is not None:
            if temp.data == old_value:
                temp.data = new_value
                isUpdated = True
            temp = temp.next
        if isUpdated:
            print("Value Updated in the linked list")
        else:
            print("Value not Updated in the linked list")

    def updateAtPosition(self, value, position):
        temp = self.head
        count = 0
        while temp is not None:
            if count == position:
                break
            count += 1
            temp = temp.next
        if temp is None:
            print("Less than {} elements in the linked list. Cannot update.".format(position))
        else:
            temp.data = value
            print("Value updated at position {}".format(position))

    def deleteFromBeginning(self):
        if self.isEmpty():
            print("Linked List is empty. Cannot delete elements.")
        elif self.head.next is None:
            self.head = None
        else:
            self.head = self.head.next
            self.head.previous = None

    def deleteFromLast(self):
        if self.isEmpty():
            print("Linked List is empty. Cannot delete elements.")
        elif self.head.next is None:
            self.head = None
        else:
            temp = self.head
            while temp.next is not None:
                temp = temp.next
            temp.previous.next = None
            temp.previous = None

    def delete(self, value):
        if self.isEmpty():
            print("Linked List is empty. Cannot delete elements.")
        elif self.head.next is None:
            if self.head.data == value:
                self.head = None
        else:
            temp = self.head
            while temp is not None:
                if temp.data == value:
                    break
                temp = temp.next
            if temp is None:
                print("Element not present in linked list. Cannot delete element.")
            elif temp.next is None:
                self.deleteFromLast()
            else:
                temp.next = temp.previous.next
                temp.next.previous = temp.previous
                temp.next = None
                temp.previous = None

    def deleteFromPosition(self, position):
        if self.isEmpty():
            print("Linked List is empty. Cannot delete elements.")
        elif position == 1:
            self.deleteFromBeginning()
        else:
            temp = self.head
            count = 1
            while temp is not None:
                if count == position:
                    break
                temp = temp.next
            if temp is None:
                print("There are less than {} elements in linked list. Cannot delete element.".format(position))
            elif temp.next is None:
                self.deleteFromLast()
                temp.previous.next = temp.next
                temp.next.previous = temp.previous
                temp.next = None
                temp.previous = None
```