```python
#!/usr/bin/env python3

# QUESTION 1 ~ Worked on by Jade Hudson and Sruthi Santhosh
class Node:
    def __init__(self, data = None, next=None):
        self.data = data
        self.next = next

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

    def insert(self, data):
        newNode = Node(data)
        if(self.head):
            current = self.head
            while(current.next):
                current = current.next
            current.next = newNode
        else:
            self.head = newNode

    def printLL(self):
        current = self.head
        while(current):
            print(current.data)
            current = current.next


LL = LinkedList()
i = 0
while i < 100 + 1:
    if i % 2 == 0:
        LL.insert(i)
    i = i + 1
LL.printLL()



# QUESTION 2 ~ Worked on by Jade Hudson and Sruthi Santhosh
class Node:
    def __init__(self, data=None, next=None):
        self.data = data
        self.next = next

    def find(self, data):
        current = self
        while current != None:
            if current.data == data:
                return Node(data)
            current = current.next
        return Node()


head = Node("Dublin")
another_node = Node("Galway")
head.next = another_node
a_third_node = Node("Cork")
another_node.next = a_third_node
result = head.find("Galway")
print(result.data) # → Galway



# QUESTION 3 ~ Worked on by Jade Hudson and Sruthi Santhosh
class Node:
    def __init__(self, data = None, next=None):
        self.data = data
        self.next = next

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

    def insert(self, data):
```

```
        newNode = Node(data)
        if(self.head):
            current = self.head
            while(current.next):
                current = current.next
            current.next = newNode
        else:
            self.head = newNode

    def printLL(self):
        current = self.head
        while(current):
            print(current.data)
            current = current.next


def reverseList(nodes):
    previous = None
    current = nodes.head
    following = current.next
    while current:
        current.next = previous
        previous = current
        current = following
        if following:
            following = following.next
    nodes.head = previous


LL = LinkedList()
i = 1
while i < 10 + 1:
    LL.insert(i)
    i = i + 1
LL.printLL()
reversed = reverseList(LL)
LL.printLL()



# QUESTION 4 ~ Worked on by Jade Hudson and Sruthi Santhosh
class Node:
    def __init__(self, data = None, next=None):
        self.data = data
        self.next = next

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

    def insert(self, data):
        newNode = Node(data)
        if(self.head):
            current = self.head
            while(current.next):
                current = current.next
            current.next = newNode
        else:
            self.head = newNode

    def Swapnodes(self):
        tmp = self.head
        if tmp is None:
            return
        while tmp is True and tmp.next is True:
            if tmp.data != tmp.next.data :
                tmp.data = tmp.next.data
                tmp.next.data = tmp.data
        tmp = tmp.next.next

    def printLL(self):
        current = self.head
        while(current):
            print(current.data)
            current = current.next
```

```
LL = LinkedList()
i = 1
while i < 10 + 1:
    LL.insert(i)
    i = i + 1
LL.printLL()
LL.Swapnodes()
LL.printLL()



# QUESTION 5 ~ Worked on by Jade Hudson and Sruthi Santhosh
class Node:
    def __init__(self, data = None, next=None):
        self.data = data
        self.next = next

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

    def insert(self, data):
        newNode = Node(data)
        if(self.head):
            current = self.head
            while(current.next):
                current = current.next
            current.next = newNode
        else:
            self.head = newNode

    def remove_n(self, n):
        if self.head is None:
            return None
        i = 0
        current = self.head
        while current.next and i < n:
            previous = current
            current = current.next
            i = i + 1
        if i == 0:
            self.head = self.head.next
        else:
            previous.next = current.next
            current = None

    def printLL(self):
        current = self.head
        while(current):
            print(current.data)
            current = current.next


LL = LinkedList()
i = 0
while i < 10:
    LL.insert(i)
    i = i + 1
LL.printLL()
n = 3
remove = LL.remove_n(n)
LL.printLL()
```