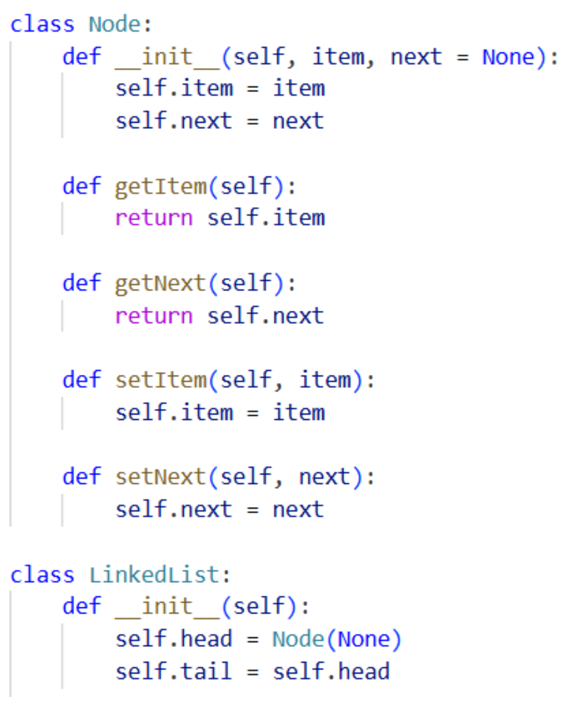
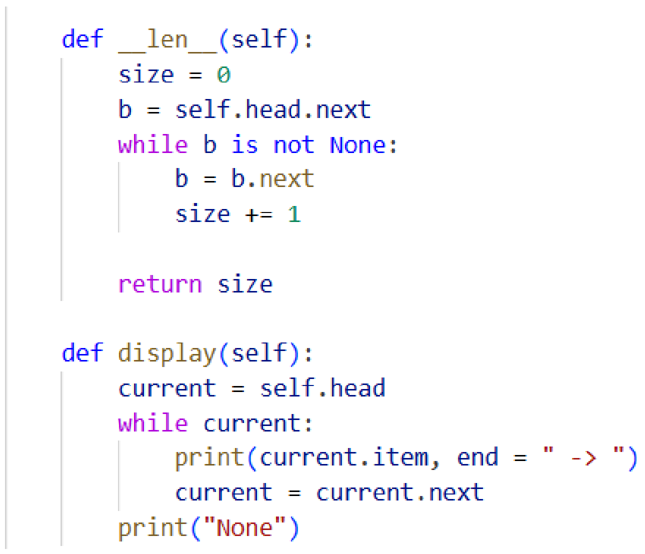
Lab Session 6

Ex A:

CODE



```python
class Node:
    def __init__(self, item, next = None):
        self.item = item
        self.next = next

    def getItem(self):
        return self.item

    def getNext(self):
        return self.next

    def setItem(self, item):
        self.item = item

    def setNext(self, next):
        self.next = next

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

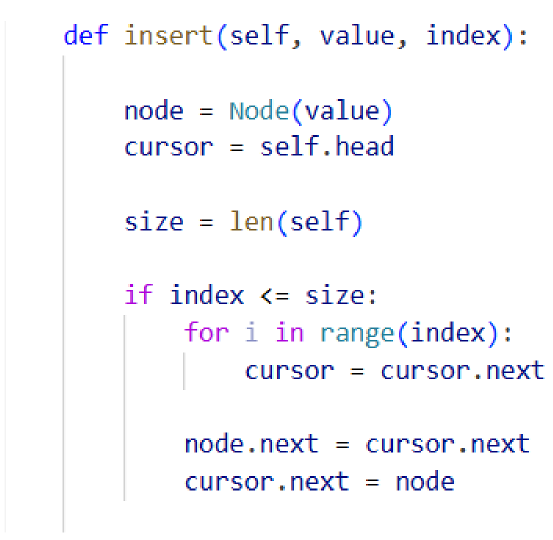


```python
def __len__(self):
    size = 0
    b = self.head.next
    while b is not None:
        b = b.next
        size += 1

    return size

def display(self):
    current = self.head
    while current:
        print(current.item, end = " -> ")
        current = current.next
    print("None")
```

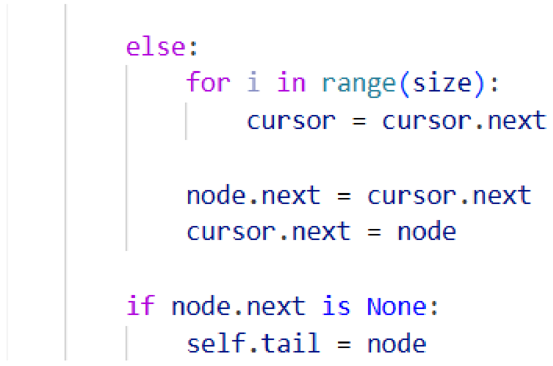
Ex B:

CODE

a) Insertion



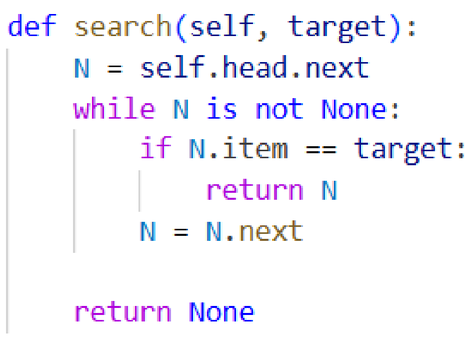
```python
def insert(self, value, index):

    node = Node(value)
    cursor = self.head

    size = len(self)

    if index <= size:
        for i in range(index):
            cursor = cursor.next

        node.next = cursor.next
        cursor.next = node
```



```python
    else:
        for i in range(size):
            cursor = cursor.next

        node.next = cursor.next
        cursor.next = node

    if node.next is None:
        self.tail = node
```

b) Searching

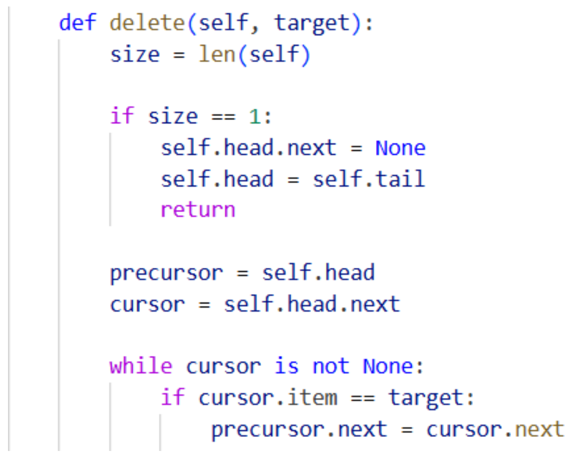
CODE



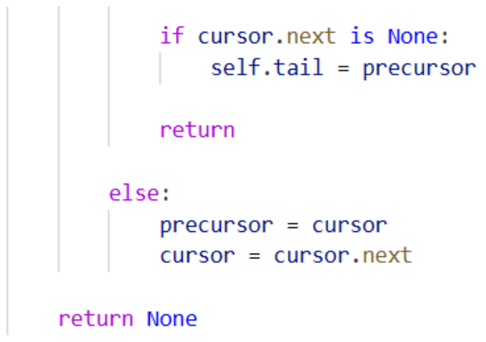
```python
def search(self, target):
    N = self.head.next
    while N is not None:
        if N.item == target:
            return N
        N = N.next

    return None
```

c) Deletion

CODE

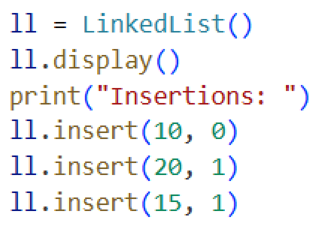


```python
def delete(self, target):
    size = len(self)

    if size == 1:
        self.head.next = None
        self.head = self.tail
        return

    precursor = self.head
    cursor = self.head.next

    while cursor is not None:
        if cursor.item == target:
            precursor.next = cursor.next
```

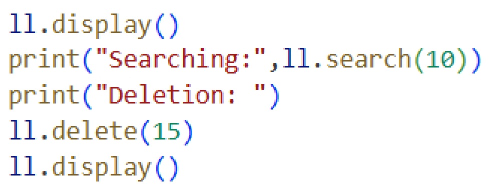


```python
            if cursor.next is None:
                self.tail = precursor

            return

        else:
            precursor = cursor
            cursor = cursor.next

    return None
```

TEST CASES

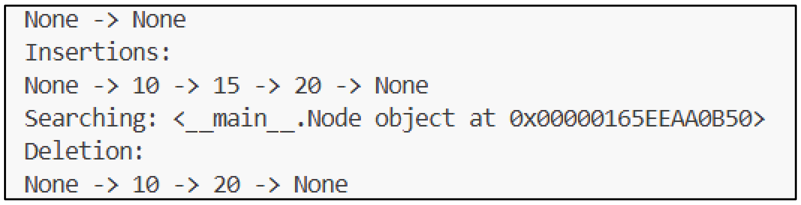


```python
ll = LinkedList()
ll.display()
print("Insertions: ")
ll.insert(10, 0)
ll.insert(20, 1)
ll.insert(15, 1)
```



```python
ll.display()
print("Searching:",ll.search(10))
print("Deletion: ")
ll.delete(15)
ll.display()
```

OUTPUT

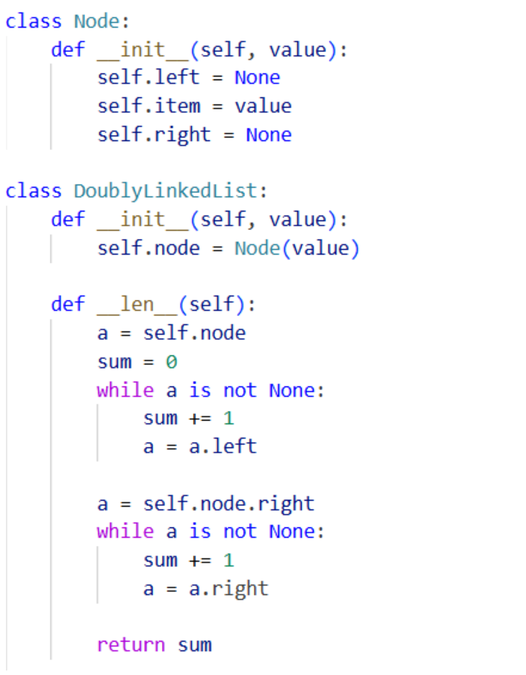


```
None -> None
Insertions:
None -> 10 -> 15 -> 20 -> None
Searching: <__main__.Node object at 0x00000165EEAA0B50>
Deletion:
None -> 10 -> 20 -> None
```

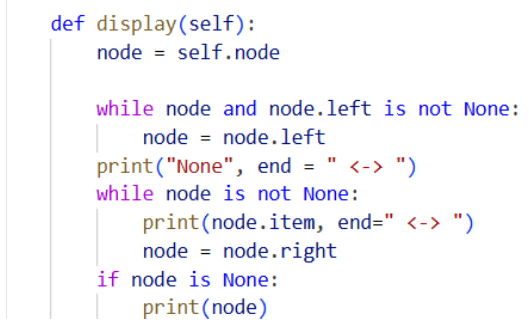
Ex C:

CODE

Initial Code:

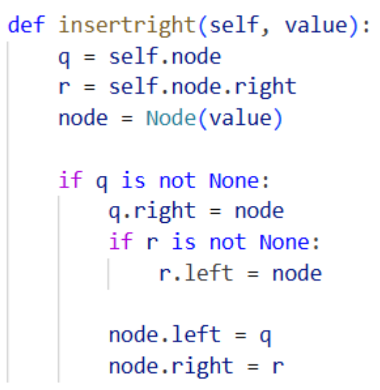


```python
class Node:
    def __init__(self, value):
        self.left = None
        self.item = value
        self.right = None

class DoublyLinkedList:
    def __init__(self, value):
        self.node = Node(value)

    def __len__(self):
        a = self.node
        sum = 0
        while a is not None:
            sum += 1
            a = a.left

        a = self.node.right
        while a is not None:
            sum += 1
            a = a.right

        return sum
```

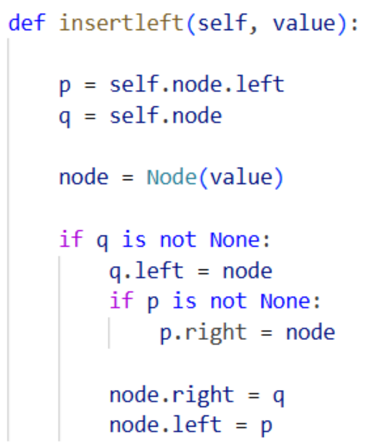


```python
    def display(self):
        node = self.node

        while node and node.left is not None:
            node = node.left
        print("None", end = " <-> ")
        while node is not None:
            print(node.item, end=" <-> ")
            node = node.right
        if node is None:
            print(node)
```

a) Insertion:

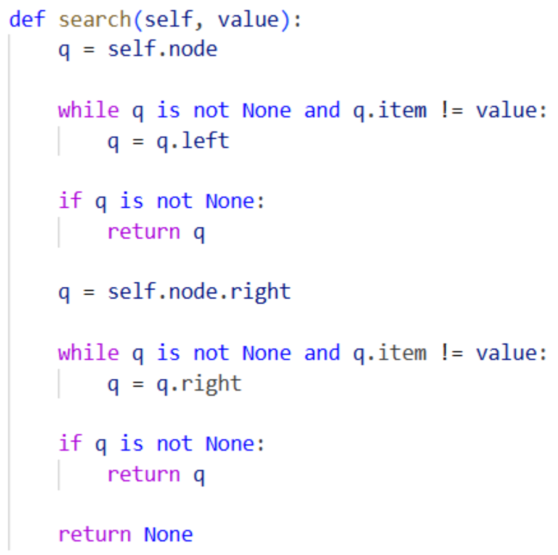


```python
def insertright(self, value):
    q = self.node
    r = self.node.right
    node = Node(value)

    if q is not None:
        q.right = node
        if r is not None:
            r.left = node

        node.left = q
        node.right = r
```



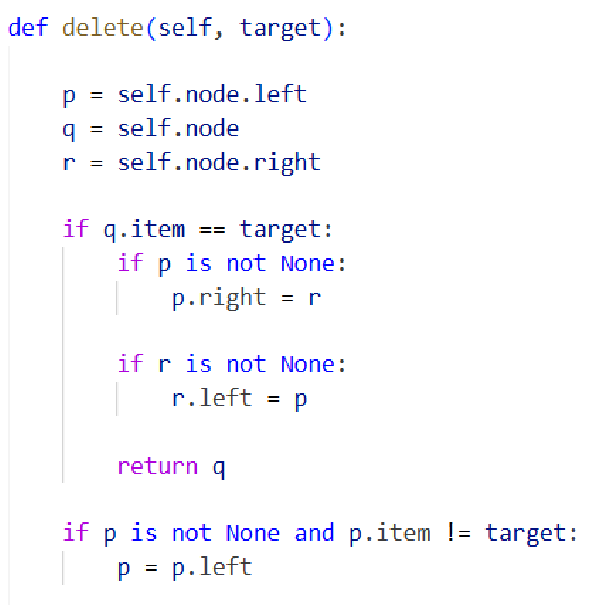
```python
def insertleft(self, value):

    p = self.node.left
    q = self.node

    node = Node(value)

    if q is not None:
        q.left = node
        if p is not None:
            p.right = node

        node.right = q
        node.left = p
```

b) Searching:

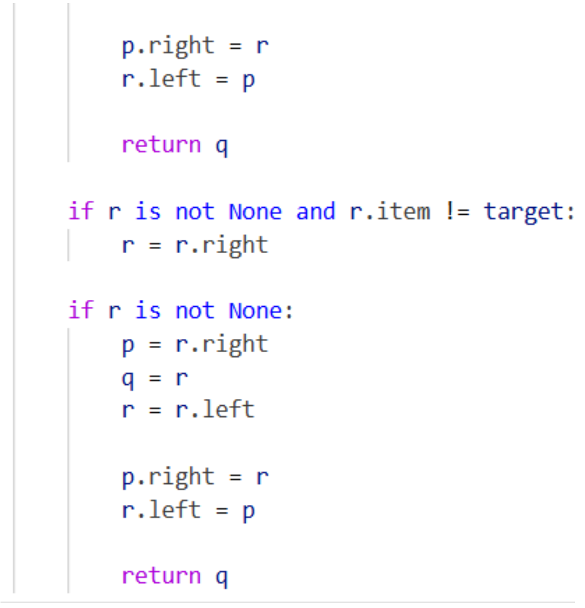


```
def search(self, value):
    q = self.node

    while q is not None and q.item != value:
        q = q.left

    if q is not None:
        return q

    q = self.node.right

    while q is not None and q.item != value:
        q = q.right

    if q is not None:
        return q

    return None
```

c) Deletion:

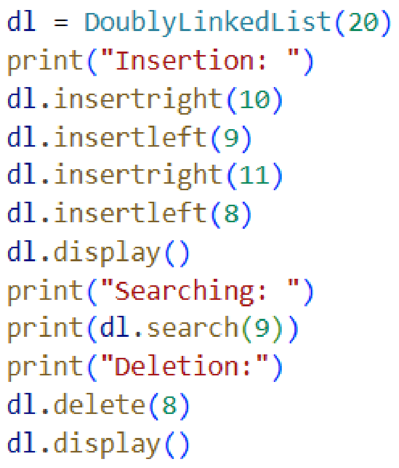


```
def delete(self, target):

    p = self.node.left
    q = self.node
    r = self.node.right

    if q.item == target:
        if p is not None:
            p.right = r

        if r is not None:
            r.left = p

        return q

    if p is not None and p.item != target:
        p = p.left
```



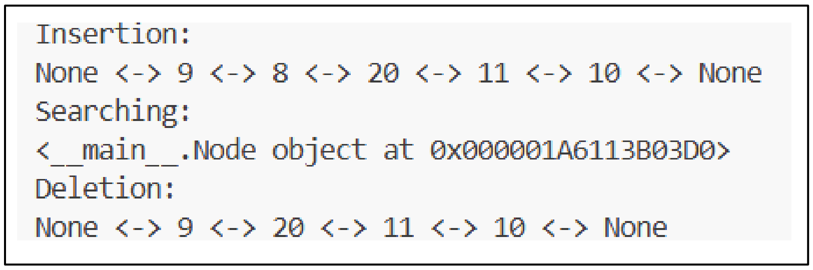
```
        p.right = r
        r.left = p

        return q

    if r is not None and r.item != target:
        r = r.right

    if r is not None:
        p = r.right
        q = r
        r = r.left

        p.right = r
        r.left = p

        return q
```

Test Case:



```python
dl = DoublyLinkedList(20)
print("Insertion: ")
dl.insertright(10)
dl.insertleft(9)
dl.insertright(11)
dl.insertleft(8)
dl.display()
print("Searching: ")
print(dl.search(9))
print("Deletion:")
dl.delete(8)
dl.display()
```

OUTPUT



```
Insertion:
None <-> 9 <-> 8 <-> 20 <-> 11 <-> 10 <-> None
Searching:
<__main__.Node object at 0x000001A6113B03D0>
Deletion:
None <-> 9 <-> 20 <-> 11 <-> 10 <-> None
```