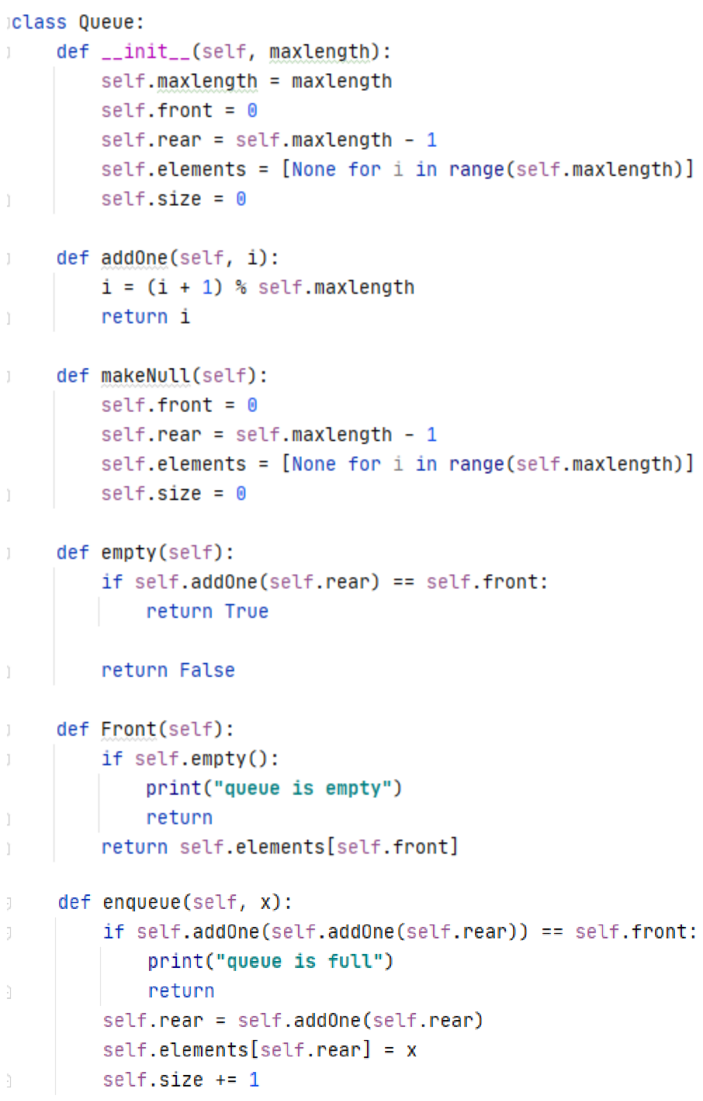
USMAN RASHEED SIDDIQUI CS-24038
NED UNIVERSITY OF ENGINEERING AND TECHNOLOGY DATA STRUCTURES AND ALGORITM

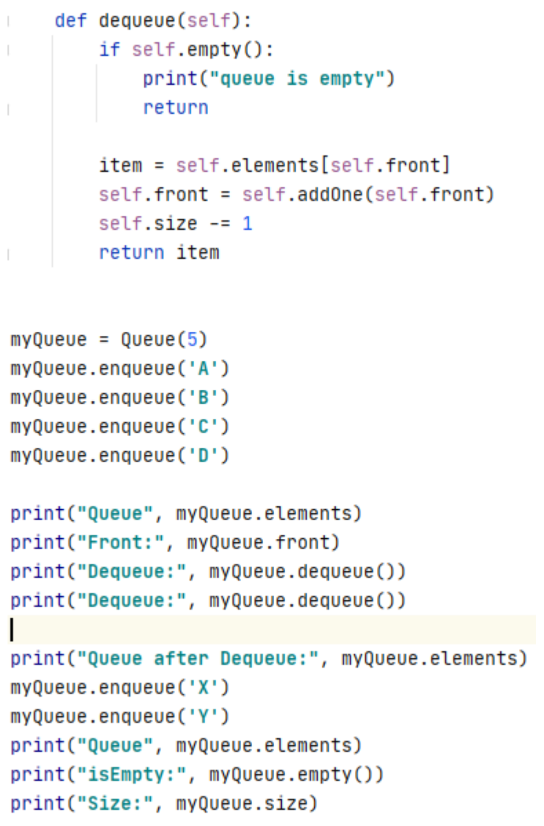
Lab Session 10

Ex A:

CODE

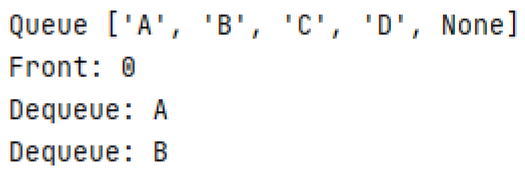


```
class Queue:
    def __init__(self, maxlength):
        self.maxlength = maxlength
        self.front = 0
        self.rear = self.maxlength - 1
        self.elements = [None for i in range(self.maxlength)]
        self.size = 0

    def addOne(self, i):
        i = (i + 1) % self.maxlength
        return i

    def makeNull(self):
        self.front = 0
        self.rear = self.maxlength - 1
        self.elements = [None for i in range(self.maxlength)]
        self.size = 0

    def empty(self):
        if self.addOne(self.rear) == self.front:
            return True

        return False

    def Front(self):
        if self.empty():
            print("queue is empty")
            return
        return self.elements[self.front]

    def enqueue(self, x):
        if self.addOne(self.addOne(self.rear)) == self.front:
            print("queue is full")
            return
        self.rear = self.addOne(self.rear)
        self.elements[self.rear] = x
        self.size += 1
```

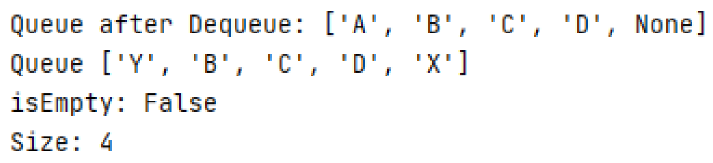


```
    def dequeue(self):
        if self.empty():
            print("queue is empty")
            return

        item = self.elements[self.front]
        self.front = self.addOne(self.front)
        self.size -= 1
        return item


myQueue = Queue(5)
myQueue.enqueue('A')
myQueue.enqueue('B')
myQueue.enqueue('C')
myQueue.enqueue('D')

print("Queue", myQueue.elements)
print("Front:", myQueue.front)
print("Dequeue:", myQueue.dequeue())
print("Dequeue:", myQueue.dequeue())

print("Queue after Dequeue:", myQueue.elements)
myQueue.enqueue('X')
myQueue.enqueue('Y')
print("Queue", myQueue.elements)
print("isEmpty:", myQueue.empty())
print("Size:", myQueue.size)
```

OUTPUT



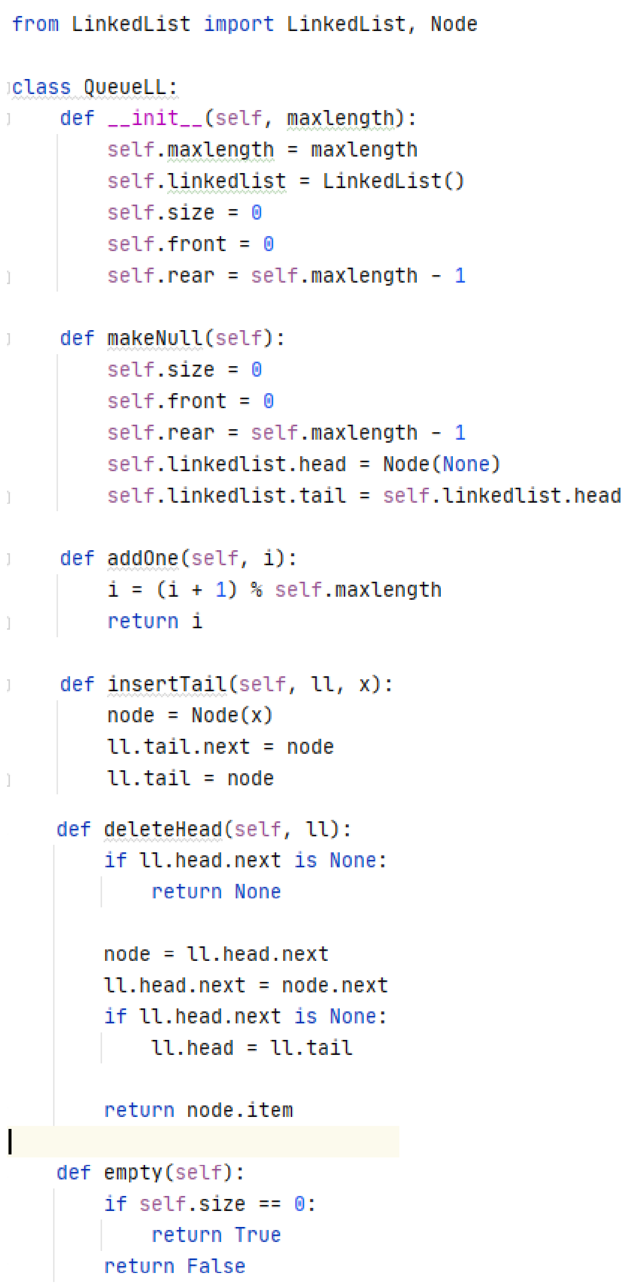
```
Queue ['A', 'B', 'C', 'D', None]
Front: 0
Dequeue: A
Dequeue: B
```



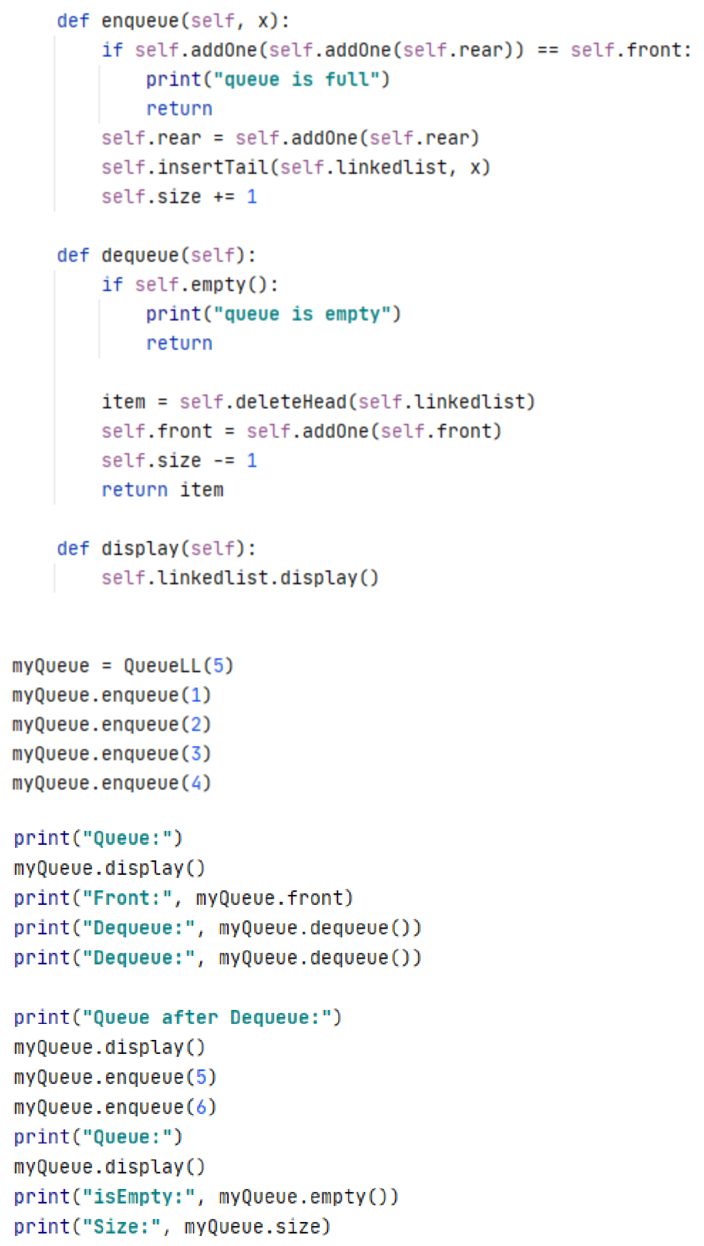
```
Queue after Dequeue: ['A', 'B', 'C', 'D', None]
Queue ['Y', 'B', 'C', 'D', 'X']
isEmpty: False
Size: 4
```

Ex D:

CODE

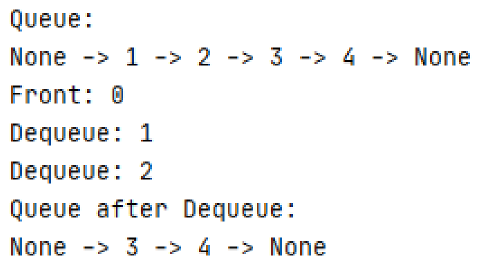


```python
from LinkedList import LinkedList, Node

class QueueLL:
    def __init__(self, maxlength):
        self.maxlength = maxlength
        self.linkedlist = LinkedList()
        self.size = 0
        self.front = 0
        self.rear = self.maxlength - 1

    def makeNull(self):
        self.size = 0
        self.front = 0
        self.rear = self.maxlength - 1
        self.linkedlist.head = Node(None)
        self.linkedlist.tail = self.linkedlist.head

    def addOne(self, i):
        i = (i + 1) % self.maxlength
        return i

    def insertTail(self, ll, x):
        node = Node(x)
        ll.tail.next = node
        ll.tail = node

    def deleteHead(self, ll):
        if ll.head.next is None:
            return None

        node = ll.head.next
        ll.head.next = node.next
        if ll.head.next is None:
            ll.head = ll.tail

        return node.item

    def empty(self):
        if self.size == 0:
            return True
        return False
```

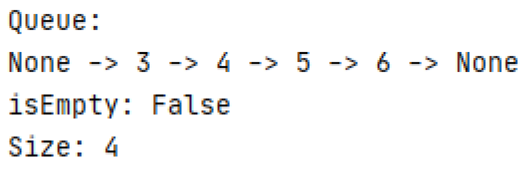


```python
    def enqueue(self, x):
        if self.addOne(self.addOne(self.rear)) == self.front:
            print("queue is full")
            return
        self.rear = self.addOne(self.rear)
        self.insertTail(self.linkedlist, x)
        self.size += 1

    def dequeue(self):
        if self.empty():
            print("queue is empty")
            return

        item = self.deleteHead(self.linkedlist)
        self.front = self.addOne(self.front)
        self.size -= 1
        return item

    def display(self):
        self.linkedlist.display()


myQueue = QueueLL(5)
myQueue.enqueue(1)
myQueue.enqueue(2)
myQueue.enqueue(3)
myQueue.enqueue(4)

print("Queue:")
myQueue.display()
print("Front:", myQueue.front)
print("Dequeue:", myQueue.dequeue())
print("Dequeue:", myQueue.dequeue())

print("Queue after Dequeue:")
myQueue.display()
myQueue.enqueue(5)
myQueue.enqueue(6)
print("Queue:")
myQueue.display()
print("isEmpty:", myQueue.empty())
print("Size:", myQueue.size)
```

OUTPUT



```
Queue:
None -> 1 -> 2 -> 3 -> 4 -> None
Front: 0
Dequeue: 1
Dequeue: 2
Queue after Dequeue:
None -> 3 -> 4 -> None
```



```
Queue:
None -> 3 -> 4 -> 5 -> 6 -> None
isEmpty: False
Size: 4
```