SCHOOL OF INFORMATION TECHNOLOGY	
NAME: Mark Lloyd Yadao	
SECTION:IDB2	DATE:11/15/2024

DSALGO1 Activity #1 (Final Term Laboratory)

Code

```python
# Linked Deque Implementation
1 usage
class _DoublyLinkedBase:
    class _Node:
        def __init__(self, element, prev, next):
            self._element = element
            self._prev = prev
            self._next = next

    def __init__(self):
        self._header = self._Node( element: None,  prev: None,  next: None)
        self._trailer = self._Node( element: None, self._header,  next: None)
        self._header._next = self._trailer
        self._size = 0

    3 usages
    def is_empty(self):
        return self._size == 0

    1 usage
    def _insert_between(self, e, predecessor, successor):
        new_node = self._Node(e, predecessor, successor)
        predecessor._next = new_node
        successor._prev = new_node
        self._size += 1

    2 usages
    def delete_first(self):
        if self.is_empty():
            raise Exception("Deque is empty!")
        return self._delete_node(self._header._next)

    1 usage
    def _delete_node(self, node):
        predecessor = node._prev
        successor = node._next
        predecessor._next = successor
        successor._prev = predecessor
        self._size -= 1
        return node._element
```

SCHOOL OF INFORMATION TECHNOLOGY	
NAME: Mark Lloyd Yadao	
SECTION:IDB2	DATE:11/15/2024

```python
    6 usages
    def insert_last(self, e):
        self._insert_between(e, self._trailer._prev, self._trailer)

    def __str__(self):
        elements = []
        current = self._header._next
        while current is not self._trailer:
            elements.append(current._element)
            current = current._next
        return str(elements)

1 usage
class LinkedDeque(_DoublyLinkedBase):
    5 usages (5 dynamic)
    def first(self):
        if self.is_empty():
            raise Exception("Deque is empty!")
        return self._header._next._element

# Linked Queue Implementation
1 usage
class LinkedQueue:
    class _Node:
        def __init__(self, element, next):
            self._element = element
            self._next = next

    def __init__(self):
        self._head = None
        self._tail = None
        self._size = 0

    4 usages
    def is_empty(self):
        return self._size == 0

    1 usage
    def enqueue(self, e):
        newest = self._Node(e, next: None)
        if self.is_empty():
```

SCHOOL OF INFORMATION TECHNOLOGY	
NAME: Mark Lloyd Yadao	
SECTION:IDB2	DATE:11/15/2024

```python
            self._head = newest
        else:
            self._tail._next = newest
        self._tail = newest
        self._size += 1

    3 usages
    def dequeue(self):
        if self.is_empty():
            raise Exception('Queue is empty')
        answer = self._head._element
        self._head = self._head._next
        self._size -= 1
        if self.is_empty():
            self._tail = None
        return answer

# Linked Stack Implementation
1 usage
class LinkedStack:
    class _Node:
        def __init__(self, element, next):
            self._element = element
            self._next = next

    def __init__(self):
        self._head = None
        self._size = 0

    1 usage
    def is_empty(self):
        return self._size == 0

    1 usage
    def push(self, e):
        self._head = self._Node(e, self._head)
        self._size += 1

    2 usages
    def pop(self):
        if self.is_empty():
```

SCHOOL OF INFORMATION TECHNOLOGY	
NAME: Mark Lloyd Yadao	
SECTION:IDB2	DATE:11/15/2024

```python
        answer = self._head._element
        self._head = self._head._next
        self._size -= 1
        return answer


1 usage
def activity_1():
    D = LinkedDeque()
    for i in range(1, 9):
        D.insert_last(i)

    print("Original Deque:", D)


    Q = LinkedQueue()
    while not D.is_empty():
        Q.enqueue(D.delete_first())

    for _ in range(3):
        D.insert_last(Q.dequeue())

    D.insert_last(Q.dequeue())
    while not Q.is_empty():
        D.insert_last(Q.dequeue())

    print("Rearranged Deque using Queue:", D)


    S = LinkedStack()
    for _ in range(5):
        S.push(D.delete_first())

    D.insert_last(S.pop())
    for _ in range(4):
        D.insert_last(S.pop())

    print("Rearranged Deque using Stack:", D)


activity_1()
```

Output

```
Original Deque: [1, 2, 3, 4, 5, 6, 7, 8]
Rearranged Deque using Queue: [1, 2, 3, 4, 5, 6, 7, 8]
Rearranged Deque using Stack: [6, 7, 8, 5, 4, 3, 2, 1]
```