


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

DSALGO1 Activity #1 (Final Term Laboratory)

Code

```
# 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



```

37         6 usages
38         def insert_last(self, e):
39             self._insert_between(e, self._trailer._prev, self._trailer)
40
41         def __str__(self):
42             elements = []
43             current = self._header._next
44             while current is not self._trailer:
45                 elements.append(current._element)
46                 current = current._next
47             return str(elements)
48
49         1 usage
50         class LinkedDeque(_DoublyLinkedBase):
51             5 usages (5 dynamic)
52             def first(self):
53                 if self.is_empty():
54                     raise Exception("Deque is empty!")
55                 return self._header._next._element
56
57         # Linked Queue Implementation
58         1 usage
59         class LinkedQueue:
60             class _Node:
61                 def __init__(self, element, next):
62                     self._element = element
63                     self._next = next
64
65             def __init__(self):
66                 self._head = None
67                 self._tail = None
68                 self._size = 0
69
70             4 usages
71             def is_empty(self):
72                 return self._size == 0
73
74             1 usage
75             def enqueue(self, e):
76                 newest = self._Node(e, next=None)
77                 if self.is_empty():

```

SCHOOL OF INFORMATION TECHNOLOGY

NAME: Mark Lloyd Yadao

SECTION: IDB2

DATE: 11/15/2024



```
72         self._head = newestest
73     else:
74         self._tail._next = newestest
75         self._tail = newestest
76         self._size += 1
77
78     3 usages
79     def dequeue(self):
80         if self.is_empty():
81             raise Exception('Queue is empty')
82         answer = self._head._element
83         self._head = self._head._next
84         self._size -= 1
85         if self.is_empty():
86             self._tail = None
87         return answer
88
89     # Linked Stack Implementation
90     1 usage
91     class _Node:
92         def __init__(self, element, next):
93             self._element = element
94             self._next = next
95
96     def __init__(self):
97         self._head = None
98         self._size = 0
99
100     1 usage
101     def is_empty(self):
102         return self._size == 0
103
104     1 usage
105     def push(self, e):
106         self._head = self._Node(e, self._head)
107         self._size += 1
108
109     2 usages
110     def pop(self):
111         if self.is_empty():
```

SCHOOL OF INFORMATION TECHNOLOGY

NAME: Mark Lloyd Yadao

SECTION:IDB2

DATE:11/15/2024



```

109         answer = self._head._element
110         self._head = self._head._next
111         self._size -= 1
112         return answer
113
114
115 1 usage
116 def activity_1():
117     D = LinkedDeque()
118     for i in range(1, 9):
119         D.insert_last(i)
120
121     print("Original Deque:", D)
122
123     Q = LinkedQueue()
124     while not D.is_empty():
125         Q.enqueue(D.delete_first())
126
127     for _ in range(3):
128         D.insert_last(Q.dequeue())
129
130     D.insert_last(Q.dequeue())
131     while not Q.is_empty():
132         D.insert_last(Q.dequeue())
133
134     print("Rearranged Deque using Queue:", D)
135
136
137     S = LinkedStack()
138     for _ in range(5):
139         S.push(D.delete_first())
140
141     D.insert_last(S.pop())
142     for _ in range(4):
143         D.insert_last(S.pop())
144
145     print("Rearranged Deque using Stack:", D)
146
147
148 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]

```