

.TermProject1A

Code:

```
1 from LinkedStack import LinkedStack
2
3 1 usage
4 class DequeWithTwoStacks:
5     def __init__(self):
6         self.stack1 = LinkedStack()
7         self.stack2 = LinkedStack()
8
9     3 usages (2 dynamic)
10    def is_empty(self):
11        return self.stack1.is_empty() and self.stack2.is_empty()
12
13    def __len__(self):
14        return len(self.stack1) + len(self.stack2)
15
16    2 usages
17    def add_first(self, e):
18        self.stack1.push(e)
19
20    2 usages
21    def add_last(self, e):
22        self.stack2.push(e)
23
24    1 usage
25    def remove_first(self):
26        if self.stack1.is_empty():
27            while not self.stack2.is_empty():
28                self.stack1.push(self.stack2.pop())
29        if self.stack1.is_empty():
30            raise Exception("Deque is empty!")
31        return self.stack1.pop()
32
33    1 usage
34    def remove_last(self):
35        if self.stack2.is_empty():
36            while not self.stack1.is_empty():
37                self.stack2.push(self.stack1.pop())
38        if self.stack2.is_empty():
39            raise Exception("Deque is empty!")
40        return self.stack2.pop()
```

```

36
1 usage
37 def first(self):
38     if self.stack1.is_empty():
39         while not self.stack2.is_empty():
40             self.stack1.push(self.stack2.pop())
41     if self.stack1.is_empty():
42         raise Exception("Deque is empty!")
43     return self.stack1.top()
44
1 usage
45 def last(self):
46     if self.stack2.is_empty():
47         while not self.stack1.is_empty():
48             self.stack2.push(self.stack1.pop())
49     if self.stack2.is_empty():
50         raise Exception("Deque is empty!")
51     return self.stack2.top()
52
1 usage
53 def print_deque(self):
54     # Gather elements from stack1 and stack2
55     stack1_elements = []
56     while not self.stack1.is_empty():
57         stack1_elements.append(self.stack1.pop())
58     stack2_elements = []
59     while not self.stack2.is_empty():
60         stack2_elements.append(self.stack2.pop())
61
62     # Refill stacks with elements
63     for element in reversed(stack2_elements):
64         self.stack2.push(element)
65     for element in reversed(stack1_elements):
66         self.stack1.push(element)
67
68     # The final deque is stack1 elements followed by stack2 elements
69     return stack1_elements + stack2_elements

```

```

70
71 try:
72     deque = DequeWithTwoStacks()
73     deque.add_first(10)
74     deque.add_last(20)
75     deque.add_first(5)
76     deque.add_last(30)
77
78     print("First element:", deque.first())
79     print("Last element:", deque.last())
80
81     print("Removed first:", deque.remove_first())
82     print("Removed last:", deque.remove_last())
83
84     print("Length:", len(deque))
85
86     print("Is empty?", deque.is_empty())
87
88     print("Deque list:", deque.print_deque())
89
90 except Exception as e:
91     print(f"Error: {e}")
92

```

Output:

```

Run  TeamProject#1A x
C:\Program Files\Python312\python.exe" Z:\FinalGrading\Trees\TeamProject#1A.py
First element: 5
Last element: 30
Removed first: 5
Removed last: 30
Length: 2
Is empty? False
Deque list: [10, 20]

Process finished with exit code 0

```

.TermProject1B

Code:

```
1 usage
1  class LinkedQueue:
2      class _Node:
3          __slots__ = '_element', '_next'
4
5          def __init__(self, element, next=None):
6              self._element = element
7              self._next = next
8
9          def __init__(self):
10             self._head = None
11             self._tail = None
12             self._size = 0
13
14         def __len__(self):
15             return self._size
16
17     9 usages (2 dynamic)
18     def is_empty(self):
19         return self._size == 0
20
21     1 usage
22     def first(self):
23         if self.is_empty():
24             raise Exception('Queue is empty')
25         return self._head._element
26
27     1 usage
28     def enqueue(self, e):
29         newest = self._Node(e, next=None)
30         if self.is_empty():
31             self._head = newest
32         else:
33             self._tail._next = newest
34             self._tail = newest
35             self._size += 1
36
37     1 usage
38     def dequeue(self):
39         if self.is_empty():
40             raise Exception('Queue is empty')
```

```

37         answer = self._head._element
38         self._head = self._head._next
39         self._size -= 1
40         if self.is_empty():
41             self._tail = None
42         return answer
43
44
45     1 usage
46     class LinkedStack:
47     class _Node:
48         __slots__ = '_element', '_next'
49
50         def __init__(self, element, next=None):
51             self._element = element
52             self._next = next
53
54         def __init__(self):
55             self._head = None
56             self._size = 0
57
58         def __len__(self):
59             return self._size
60
61     8 usages (2 dynamic)
62     def is_empty(self):
63         return self._size == 0
64
65     2 usages
66     def push(self, e):
67         self._head = self._Node(e, self._head)
68         self._size += 1
69
70     1 usage
71     def top(self):
72         if self.is_empty():
73             raise Exception('Stack is empty')
74         return self._head._element

```

```

72     def pop(self):
73         if self.is_empty():
74             raise Exception('Stack is empty')
75         answer = self._head._element
76         self._head = self._head._next
77         self._size -= 1
78         return answer
79
80
81     1 usage
82     class DequeWithStackAndQueue:
83         def __init__(self):
84             self.stack = LinkedStack()
85             self.queue = LinkedQueue()
86
87         3 usages (2 dynamic)
88         def is_empty(self):
89             return self.stack.is_empty() and self.queue.is_empty()
90
91         def __len__(self):
92             return len(self.stack) + len(self.queue)
93
94         4 usages
95         def add_first(self, e):
96             self.stack.push(e)
97
98         4 usages
99         def add_last(self, e):
100             self.queue.enqueue(e)
101
102         1 usage
103         def remove_first(self):
104             if self.stack.is_empty():
105                 raise Exception('Deque is empty')
106             return self.stack.pop()
107
108         1 usage
109         def remove_last(self):
110             if self.queue.is_empty():
111                 raise Exception('Deque is empty')
112             return self.queue.dequeue()

```

```

108 1 usage
109  def first(self):
110     if self.stack.is_empty():
111         raise Exception('Deque is empty')
112     return self.stack.top()

113 1 usage
114  def last(self):
115     if self.queue.is_empty():
116         raise Exception('Deque is empty')
117     return self.queue.first()

118 2 usages
119  def print_deque(self):
120     stack_elements = []
121     while not self.stack.is_empty():
122         stack_elements.append(self.stack.pop())
123     queue_elements = []
124     current = self.queue._head
125     while current:
126         queue_elements.append(current._element)
127         current = current._next
128     for element in reversed(stack_elements):
129         self.stack.push(element)
130
131     return stack_elements + queue_elements
132
133  try:
134     print("Test started")
135     deque = DequeWithStackAndQueue()
136
137     deque.add_first(10)
138     deque.add_last(20)
139     deque.add_first(5)
140     deque.add_last(30)
141     deque.add_first(2)
142     deque.add_last(40)
143     deque.add_first(1)
144     deque.add_last(50)
145

```

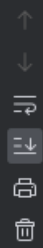
LinkedQueue

```
145
146     print("First element:", deque.first())
147     print("Last element:", deque.last())
148
149     print("Deque list:", deque.print_deque())
150
151     print("Removed first:", deque.remove_first())
152     print("Removed last:", deque.remove_last())
153
154     print("Length:", len(deque))
155
156     print("Is empty?", deque.is_empty())
157     print("Deque list:", deque.print_deque())
158 except Exception as e:
159     print(f"Error: {e}")
160
```

Output:

Run

TeamProject#1B x



"C:\Program Files\Python312\python.exe" Z:\FinalGrading\Trees\TeamProject#1B.py

Test started

First element: 1

Last element: 20

Deque list: [1, 2, 5, 10, 20, 30, 40, 50]

Removed first: 1

Removed last: 20

Length: 6

Is empty? False

Deque list: [2, 5, 10, 30, 40, 50]

Process finished with exit code 0