

```
1 from LinkedStack import LinkedStack
2
3 class Deque:
4
5     new *
6     def __init__(self):
7         """Create an empty deque."""
8         self._left_stack = LinkedStack()
9         self._right_stack = LinkedStack()
10
11     new *
12     def is_empty(self):
13         """Return True if the deque is empty."""
14         return self._left_stack.is_empty() and self._right_stack.is_empty()
15
16     new *
17     def __len__(self):
18         """Return the number of elements in the deque."""
19         return len(self._left_stack) + len(self._right_stack)
20
21     2 usages new *
22     def add_first(self, e):
23         """Add an element to the front of the deque."""
24         self._left_stack.push(e)
25
26     2 usages new *
27     def add_last(self, e):
28         """Add an element to the back of the deque."""
29         self._right_stack.push(e)
30
31     2 usages new *
32     def remove_first(self):
33         """Remove and return the element at the front of the deque."""
34         if self._left_stack.is_empty():
35             while not self._right_stack.is_empty():
36                 self._left_stack.push(self._right_stack.pop())
37         if self._left_stack.is_empty():
38             raise Exception('Deque is empty')
39         return self._left_stack.pop()
```

```
35     def remove_last(self):
36         """Remove and return the element at the back of the deque."""
37         if self._right_stack.is_empty():
38             while not self._left_stack.is_empty():
39                 self._right_stack.push(self._left_stack.pop())
40         if self._right_stack.is_empty():
41             raise Exception('Deque is empty')
42         return self._right_stack.pop()
43
44     new *
45     def first(self):
46         """Return but do not remove the element at the front of the deque."""
47         if self._left_stack.is_empty():
48             while not self._right_stack.is_empty():
49                 self._left_stack.push(self._right_stack.pop())
50         if self._left_stack.is_empty():
51             raise Exception('Deque is empty')
52         return self._left_stack.top()
53
54     new *
55     def last(self):
56         """Return but do not remove the element at the back of the deque."""
57         if self._right_stack.is_empty():
58             while not self._left_stack.is_empty():
59                 self._right_stack.push(self._left_stack.pop())
60         if self._right_stack.is_empty():
61             raise Exception('Deque is empty')
62         return self._right_stack.top()
63
64     1 usage new *
65     def test_deque():
66         deque = Deque()
67         print("Is deque empty?", deque.is_empty())
68         print("Length of deque:", len(deque))
69         deque.add_first(10)
70         deque.add_last(20)
71         deque.add_first(5)
72         deque.add_last(30)
73         print("After adding elements:")
74         print("First element:", deque.first())
75         print("Last element:", deque.last())
```

```

62 1 usage new *
63 def test_deque():
64     deque = Deque()
65     print("Is deque empty?", deque.is_empty())
66     print("Length of deque:", len(deque))
67     deque.add_first(10)
68     deque.add_last(20)
69     deque.add_first(5)
70     deque.add_last(30)
71     print("After adding elements:")
72     print("First element:", deque.first())
73     print("Last element:", deque.last())
74     print("Length of deque:", len(deque))
75
76     print("Removed first element:", deque.remove_first())
77     print("Removed last element:", deque.remove_last())
78
79     print("After removing elements:")
80     print("First element:", deque.first())
81     print("Last element:", deque.last())
82     print("Length of deque:", len(deque))
83
84     # Remove remaining elements
85     print("Removed first element:", deque.remove_first())
86     print("Removed last element:", deque.remove_last())
87
88     print("Is deque empty?", deque.is_empty())
89     print("Length of deque:", len(deque))
90
91 if __name__ == "__main__":
    test_deque()

```

```
1 from LinkedStack import LinkedStack
2 from LinkedQueue import LinkedQueue
3
4 1 usage new *
5 class Deque:
6     """Double-ended queue implementation using a stack and a queue."""
7
8     new *
9     def __init__(self):
10         """Create an empty deque."""
11         self._stack = LinkedStack()
12         self._queue = LinkedQueue()
13
14     new *
15     def is_empty(self):
16         """Return True if the deque is empty."""
17         return self._stack.is_empty() and self._queue.is_empty()
18
19     new *
20     def __len__(self):
21         """Return the number of elements in the deque."""
22         return len(self._stack) + len(self._queue)
23
24     2 usages new *
25     def add_first(self, e):
26         """Add an element to the front of the deque."""
27         self._stack.push(e)
28
29     2 usages new *
30     def add_last(self, e):
31         """Add an element to the back of the deque."""
32         self._queue.enqueue(e)
33
34     2 usages new *
35     def remove_first(self):
36         """Remove and return the element at the front of the deque."""
37         if self._stack.is_empty():
38             if self._queue.is_empty():
39                 raise Exception('Deque is empty')
40             while not self._queue.is_empty():
```

```
33         while not self._queue.is_empty():
34             self._stack.push(self._queue.dequeue())
35         return self._stack.pop()
36
37     2 usages new *
38     def remove_last(self):
39         """Remove and return the element at the back of the deque."""
40         if self._queue.is_empty():
41             if self._stack.is_empty():
42                 raise Exception('Deque is empty')
43             while not self._stack.is_empty():
44                 self._queue.enqueue(self._stack.pop())
45         return self._queue.dequeue()
46
47     new *
48     def first(self):
49         """Return but do not remove the element at the front of the deque."""
50         if self._stack.is_empty():
51             if self._queue.is_empty():
52                 raise Exception('Deque is empty')
53             while not self._queue.is_empty():
54                 self._stack.push(self._queue.dequeue())
55         return self._stack.top()
56
57     new *
58     def last(self):
59         """Return but do not remove the element at the back of the deque."""
60         if self._queue.is_empty():
61             if self._stack.is_empty():
62                 raise Exception('Deque is empty')
63             while not self._stack.is_empty():
64                 self._queue.enqueue(self._stack.pop())
65         return self._queue.first()
66
67     1 usage new *
68     def test_deque():
69         deque = Deque()
70
71         print(deque.is_empty())
72         print(len(deque))
```

Deque : is\_empty()

1 usage new \*

4 def test\_deque():

5 deque = Deque()

6  
7 print(deque.is\_empty())

8 print(len(deque))

9  
10 deque.add\_first(10)

11 deque.add\_last(20)

12 deque.add\_first(5)

13 deque.add\_last(30)

14  
15 print("After adding elements:")

16 print(deque.first())

17 print(deque.last())

18 print(len(deque))

19 print(deque.remove\_first())

20 print(deque.remove\_last())

21  
22 print("After removing elements:")

23 print(deque.first())

24 print(deque.last())

25 print(len(deque))

26  
27 print(deque.remove\_first())

28 print(deque.remove\_last())

29  
30 print(deque.is\_empty())

31 print(len(deque))

32  if \_\_name\_\_ == "\_\_main\_\_":

33 test\_deque()

```
Z:\DSALG01-1DB2\venv\Scripts\python.exe "Z:\DSALG01-1DB2\FINALS\Activities\Term Project #1(part 1).py"
Is deque empty? True
Length of deque: 0
After adding elements:
First element: 5
Last element: 30
Length of deque: 4
Removed first element: 5
Removed last element: 30
After removing elements:
First element: 10
Last element: 20
Length of deque: 2
Removed first element: 10
Removed last element: 20
Is deque empty? True
Length of deque: 0

Process finished with exit code 0
```

```
Z:\DSALG01-1DB2\venv\Scripts\python.exe "Z:\DSALG01-1DB2\FINALS\Activities\Term Project #1(part 2).p
True
0
After adding elements:
5
20
4
5
20
After removing elements:
10
30
2
10
30
True
0

Process finished with exit code 0
```