

```
1 function Node(data) {
2   this.data = data;
3   this.previous = null;
4   this.next = null;
5 }
6
7 class DoublyLinkedList {
8   constructor() {
9     this.head = null;
10    this.tail = null;
11    this.numberOfValues = 0;
12  }
13
14  add(data) {
15    const node = new Node(data);
16    if(!this.head) {
17      this.head = node;
18      this.tail = node;
19    } else {
20      node.previous = this.tail;
21      this.tail.next = node;
22      this.tail = node;
23    }
24    this.numberOfValues++;
25  }
26
27  remove(data) {
28    let current = this.head;
29    while(current) {
30      if(current.data === data) {
31        if(current === this.head && current === this.tail) {
32          this.head = null;
33          this.tail = null;
34        } else if(current === this.head) {
35          this.head = this.head.next;
36          this.head.previous = null;
37        } else if(current === this.tail) {
38          this.tail = this.tail.previous;
39          this.tail.next = null;
40        } else {
41          current.previous.next = current.next;
42          current.next.previous = current.previous;
43        }
44        this.numberOfValues--;
45      }
46      current = current.next;
47    }
48  }
49
50  insertAfter(data, toNodeData) {
51    let current = this.head;
52    while(current) {
53      if(current.data === toNodeData) {
54        const node = new Node(data);
55        if(current === this.tail) {
56          this.add(data);
57        } else {
58          current.next.previous = node;
59          node.previous = current;
60          node.next = current.next;
```

```
61         current.next = node;
62         this.numberOfValues++;
63     }
64 }
65     current = current.next;
66 }
67 }
68
69 traverse(fn) {
70     let current = this.head;
71     while(current) {
72         if(fn) {
73             fn(current);
74         }
75         current = current.next;
76     }
77 }
78
79 traverseReverse(fn) {
80     let current = this.tail;
81     while(current) {
82         if(fn) {
83             fn(current);
84         }
85         current = current.previous;
86     }
87 }
88
89 length() {
90     return this.numberOfValues;
91 }
92
93 print() {
94     let string = '';
95     let current = this.head;
96     while(current) {
97         string += `${current.data} `;
98         current = current.next;
99     }
100     console.log(string.trim());
101 }
102 }
103
104 const doublyLinkedList = new DoublyLinkedList();
105 doublyLinkedList.print(); // => ''
106 doublyLinkedList.add(1);
107 doublyLinkedList.add(2);
108 doublyLinkedList.add(3);
109 doublyLinkedList.add(4);
110 doublyLinkedList.print(); // => 1 2 3 4
111 console.log('length is 4:', doublyLinkedList.length()); // => 4
112 doublyLinkedList.remove(3); // remove value
113 doublyLinkedList.print(); // => 1 2 4
114 doublyLinkedList.remove(9); // remove non existing value
115 doublyLinkedList.print(); // => 1 2 4
116 doublyLinkedList.remove(1); // remove head
117 doublyLinkedList.print(); // => 2 4
118 doublyLinkedList.remove(4); // remove tail
119 doublyLinkedList.print(); // => 2
120 console.log('length is 1:', doublyLinkedList.length()); // => 1
```

```
121 doublyLinkedList.remove(2); // remove tail, the list should be empty
122 doublyLinkedList.print(); // => ''
123 console.log('length is 0:', doublyLinkedList.length()); // => 0
124 doublyLinkedList.add(2);
125 doublyLinkedList.add(6);
126 doublyLinkedList.print(); // => 2 6
127 doublyLinkedList.insertAfter(3, 2);
128 doublyLinkedList.print(); // => 2 3 6
129 doublyLinkedList.traverseReverse(node => { console.log(node.data); });
130 doublyLinkedList.insertAfter(4, 3);
131 doublyLinkedList.print(); // => 2 3 4 6
132 doublyLinkedList.insertAfter(5, 9); // insertAfter a non existing node
133 doublyLinkedList.print(); // => 2 3 4 6
134 doublyLinkedList.insertAfter(5, 4);
135 doublyLinkedList.insertAfter(7, 6); // insertAfter the tail
136 doublyLinkedList.print(); // => 2 3 4 5 6 7
137 doublyLinkedList.add(8); // add node with normal method
138 doublyLinkedList.print(); // => 2 3 4 5 6 7 8
139 console.log('length is 7:', doublyLinkedList.length()); // => 7
140 doublyLinkedList.traverse(node => { node.data = node.data + 10; });
141 doublyLinkedList.print(); // => 12 13 14 15 16 17 18
142 doublyLinkedList.traverse(node => { console.log(node.data); }); // => 12 13 14 15 16
143 console.log('length is 7:', doublyLinkedList.length()); // => 7
144 doublyLinkedList.traverseReverse(node => { console.log(node.data); }); // => 18 17 16
145 doublyLinkedList.print(); // => 12 13 14 15 16 17 18
146 console.log('length is 7:', doublyLinkedList.length()); // => 7
147
```