

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

```

```
61     }
62     current = current.next;
63 }
64 };
65 DoublyLinkedList.prototype.traverse = function(fn) {
66     var current = this.head;
67     while(current) {
68         if(fn) {
69             fn(current);
70         }
71         current = current.next;
72     }
73 };
74 DoublyLinkedList.prototype.traverseReverse = function(fn) {
75     var current = this.tail;
76     while(current) {
77         if(fn) {
78             fn(current);
79         }
80         current = current.previous;
81     }
82 };
83 DoublyLinkedList.prototype.length = function() {
84     return this.numberOfValues;
85 };
86 DoublyLinkedList.prototype.print = function() {
87     var string = '';
88     var current = this.head;
89     while(current) {
90         string += current.data + ' ';
91         current = current.next;
92     }
93     console.log(string.trim());
94 };
95
96 var doublyLinkedList = new DoublyLinkedList();
97 doublyLinkedList.print(); // => ''
98 doublyLinkedList.add(1);
99 doublyLinkedList.add(2);
100 doublyLinkedList.add(3);
101 doublyLinkedList.add(4);
102 doublyLinkedList.print(); // => 1 2 3 4
103 console.log('length is 4:', doublyLinkedList.length()); // => 4
104 doublyLinkedList.remove(3); // remove value
105 doublyLinkedList.print(); // => 1 2 4
106 doublyLinkedList.remove(9); // remove non existing value
107 doublyLinkedList.print(); // => 1 2 4
108 doublyLinkedList.remove(1); // remove head
109 doublyLinkedList.print(); // => 2 4
110 doublyLinkedList.remove(4); // remove tail
111 doublyLinkedList.print(); // => 2
112 console.log('length is 1:', doublyLinkedList.length()); // => 1
113 doublyLinkedList.remove(2); // remove tail, the list should be empty
114 doublyLinkedList.print(); // => ''
115 console.log('length is 0:', doublyLinkedList.length()); // => 0
116 doublyLinkedList.add(2);
117 doublyLinkedList.add(6);
118 doublyLinkedList.print(); // => 2 6
119 doublyLinkedList.insertAfter(3, 2);
120 doublyLinkedList.print(); // => 2 3 6
```

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