

```
1 function Node(data) {
2   this.data = data;
3   this.next = null;
4 }
5
6 class SinglyLinkedList {
7   constructor() {
8     this.head = null;
9     this.tail = null;
10    this.numberOfValues = 0;
11  }
12
13  add(data) {
14    const node = new Node(data);
15    if(!this.head) {
16      this.head = node;
17      this.tail = node;
18    } else {
19      this.tail.next = node;
20      this.tail = node;
21    }
22    this.numberOfValues++;
23  }
24
25  remove(data) {
26    let previous = this.head;
27    let current = this.head;
28    while(current) {
29      if(current.data === data) {
30        if(current === this.head) {
31          this.head = this.head.next;
32        }
33        if(current === this.tail) {
34          this.tail = previous;
35        }
36        previous.next = current.next;
37        this.numberOfValues--;
38      } else {
39        previous = current;
40      }
41      current = current.next;
42    }
43  }
44
45  insertAfter(data, toNodeData) {
46    let current = this.head;
47    while(current) {
48      if(current.data === toNodeData) {
49        const node = new Node(data);
50        if(current === this.tail) {
51          this.tail.next = node;
52          this.tail = node;
53        } else {
54          node.next = current.next;
55          current.next = node;
56        }
57        this.numberOfValues++;
58      }
59      current = current.next;
60    }
61  }
```

```
61 }
62
63 traverse(fn) {
64     let current = this.head;
65     while(current) {
66         if(fn) {
67             fn(current);
68         }
69         current = current.next;
70     }
71 }
72
73 length() {
74     return this.numberOfValues;
75 }
76
77 print() {
78     let string = '';
79     let current = this.head;
80     while(current) {
81         string += `${current.data} `;
82         current = current.next;
83     }
84     console.log(string.trim());
85 }
86 }
87
88 const singlyLinkedList = new SinglyLinkedList();
89 singlyLinkedList.print(); // => ''
90 singlyLinkedList.add(1);
91 singlyLinkedList.add(2);
92 singlyLinkedList.add(3);
93 singlyLinkedList.add(4);
94 singlyLinkedList.print(); // => 1 2 3 4
95 console.log('length is 4:', singlyLinkedList.length()); // => 4
96 singlyLinkedList.remove(3); // remove value
97 singlyLinkedList.print(); // => 1 2 4
98 singlyLinkedList.remove(9); // remove non existing value
99 singlyLinkedList.print(); // => 1 2 4
100 singlyLinkedList.remove(1); // remove head
101 singlyLinkedList.print(); // => 2 4
102 singlyLinkedList.remove(4); // remove tail
103 singlyLinkedList.print(); // => 2
104 console.log('length is 1:', singlyLinkedList.length()); // => 1
105 singlyLinkedList.add(6);
106 singlyLinkedList.print(); // => 2 6
107 singlyLinkedList.insertAfter(3, 2);
108 singlyLinkedList.print(); // => 2 3 6
109 singlyLinkedList.insertAfter(4, 3);
110 singlyLinkedList.print(); // => 2 3 4 6
111 singlyLinkedList.insertAfter(5, 9); // insertAfter a non existing node
112 singlyLinkedList.print(); // => 2 3 4 6
113 singlyLinkedList.insertAfter(5, 4);
114 singlyLinkedList.insertAfter(7, 6); // insertAfter the tail
115 singlyLinkedList.print(); // => 2 3 4 5 6 7
116 singlyLinkedList.add(8); // add node with normal method
117 singlyLinkedList.print(); // => 2 3 4 5 6 7 8
118 console.log('length is 7:', singlyLinkedList.length()); // => 7
119 singlyLinkedList.traverse(node => { node.data = node.data + 10; });
120 singlyLinkedList.print(); // => 12 13 14 15 16 17 18
```

```
121 singlyLinkedList.traverse(node => { console.log(node.data); }); // => 12 13 14 15 16
    17 18
122 console.log('length is 7:', singlyLinkedList.length()); // => 7
123
```