

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

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