```
function Node(data) {
  this.data = data;
  this.previous = null;
  this.next = null;
}

function DoublyLinkedList() {
  this.head = null;
  this.tail = null;
  this.numberOfValues = 0;
}

DoublyLinkedList.prototype.add = function (data) {
  var node = new Node(data);
  if(!this.head) {
    this.head = node;
    this.tail = node;
  } else {
    node.previous = this.tail;
    this.tail.next = node;
    this.tail = node;
  }
  this.numberOfValues++;
};
DoublyLinkedList.prototype.remove = function(data) {
  var current = this.head;
  while(current) {
    if(current.data === data) {
      if(current === this.head && current === this.tail) {
        this.head = null;
        this.tail = null;
      } else if(current === this.head) {
        this.head = this.head.next;
        this.head.previous = null;
      } else if(current === this.tail) {
        this.tail = this.tail.previous;
        this.tail.next = null;
      } else {
        current.previous.next = current.next;
        current.next.previous = current.previous;
      }
      this.numberOfValues--;
    }
    current = current.next;
  }
};
DoublyLinkedList.prototype.insertAfter = function(data, toNodeData) {
  var current = this.head;
  while(current) {
    if(current.data === toNodeData) {
      var node = new Node(data);
      if(current === this.tail) {
        this.add(data);
      } else {
        current.next.previous = node;
        node.previous = current;
        node.next = current.next;
        current.next = node;
        this.numberOfValues++;
      }
```

```
    }
    current = current.next;
  }
};
DoublyLinkedList.prototype.traverse = function(fn) {
  var current = this.head;
  while(current) {
    if(fn) {
      fn(current);
    }
    current = current.next;
  }
};
DoublyLinkedList.prototype.traverseReverse = function(fn) {
  var current = this.tail;
  while(current) {
    if(fn) {
      fn(current);
    }
    current = current.previous;
  }
};
DoublyLinkedList.prototype.length = function() {
  return this.numberOfValues;
};
DoublyLinkedList.prototype.print = function() {
  var string = '';
  var current = this.head;
  while(current) {
    string += current.data + ' ';
    current = current.next;
  }
  console.log(string.trim());
};

var doublyLinkedList = new DoublyLinkedList();
doublyLinkedList.print(); // => ''
doublyLinkedList.add(1);
doublyLinkedList.add(2);
doublyLinkedList.add(3);
doublyLinkedList.add(4);
doublyLinkedList.print(); // => 1 2 3 4
console.log('length is 4:', doublyLinkedList.length()); // => 4
doublyLinkedList.remove(3); // remove value
doublyLinkedList.print(); // => 1 2 4
doublyLinkedList.remove(9); // remove non existing value
doublyLinkedList.print(); // => 1 2 4
doublyLinkedList.remove(1); // remove head
doublyLinkedList.print(); // => 2 4
doublyLinkedList.remove(4); // remove tail
doublyLinkedList.print(); // => 2
console.log('length is 1:', doublyLinkedList.length()); // => 1
doublyLinkedList.remove(2); // remove tail, the list should be empty
doublyLinkedList.print(); // => ''
console.log('length is 0:', doublyLinkedList.length()); // => 0
doublyLinkedList.add(2);
doublyLinkedList.add(6);
doublyLinkedList.print(); // => 2 6
doublyLinkedList.insertAfter(3, 2);
doublyLinkedList.print(); // => 2 3 6
```

```
doublyLinkedList.traverseReverse(function(node) { console.log(node.data); });
doublyLinkedList.insertAfter(4, 3);
doublyLinkedList.print(); // => 2 3 4 6
doublyLinkedList.insertAfter(5, 9); // insertAfter a non existing node
doublyLinkedList.print(); // => 2 3 4 6
doublyLinkedList.insertAfter(5, 4);
doublyLinkedList.insertAfter(7, 6); // insertAfter the tail
doublyLinkedList.print(); // => 2 3 4 5 6 7
doublyLinkedList.add(8); // add node with normal method
doublyLinkedList.print(); // => 2 3 4 5 6 7 8
console.log('length is 7:', doublyLinkedList.length()); // => 7
doublyLinkedList.traverse(function(node) { node.data = node.data + 10; });
doublyLinkedList.print(); // => 12 13 14 15 16 17 18
doublyLinkedList.traverse(function(node) { console.log(node.data); }); // => 12 13 14
15 16 17 18
console.log('length is 7:', doublyLinkedList.length()); // => 7
doublyLinkedList.traverseReverse(function(node) { console.log(node.data); }); // =>
18 17 16 15 14 13 12
doublyLinkedList.print(); // => 12 13 14 15 16 17 18
console.log('length is 7:', doublyLinkedList.length()); // => 7

```