**Doubly Linked List**

class Node {

    constructor(value) {

        this.value = value;

        this.next = null;

        this.prev = null;

    }

}

class DoublyLinkedList {

    constructor(value) {

        const newNode = new Node(value);

        this.head = newNode;

        this.tail = newNode;

        this.length = 1;

    }

    printList() {

        let temp = this.head;

        while (temp !== null) {

            console.log(temp.value);

            temp = temp.next;

        }

    }

    getHead() {

        if (this.head === null) {

            console.log("Head: null");

        } else {

            console.log("Head: " + this.head.value);

        }

    }

    getTail() {

        if (this.tail === null) {

            console.log("Tail: null");

        } else {

            console.log("Tail: " + this.tail.value);

        }

    }

    getLength() {

        console.log("Length: " + this.length);

    }

    makeEmpty() {

        this.head = null;

        this.tail = null;

        this.length = 0;

    }

    push(value) {

        const newNode = new Node(value);

        if (this.length === 0) {

            this.head = newNode;

            this.tail = newNode;

        } else {

            this.tail.next = newNode;

            newNode.prev = this.tail;

            this.tail = newNode;

        }

        this.length++;

        return this;

    }

    pop() {

        if (this.length === 0) return undefined;

        let temp = this.tail;

        if (this.length === 1) {

            this.head = null;

            this.tail = null;

        } else {

            this.tail = this.tail.prev;

            this.tail.next = null;

            temp.prev = null;

        }

        this.length--;

        return temp;

    }

    unshift(value) {

        const newNode = new Node(value);

        if (this.length === 0) {

            this.head = newNode;

            this.tail = newNode;

        } else {

            newNode.next = this.head;

            this.head.prev = newNode;

            this.head = newNode;

        }

        this.length++;

        return this;

    }

    shift() {

        if (this.length === 0) return undefined;

        let temp = this.head;

        if (this.length === 1) {

            this.head = null;

            this.tail = null;

        } else {

            this.head = this.head.next;

            this.head.prev = null;

            temp.next = null;

        }

        this.length--;

        return temp;

    }

    get(index) {

        if (index < 0 || index >= this.length) return undefined;

        let temp = this.head;

        if (index < this.length / 2) {

            for (let i = 0; i < index; i++) {

                temp = temp.next;

            }

        } else {

            temp = this.tail;

            for (let i = this.length - 1; i > index; i--) {

                temp = temp.prev;

            }

        }

        return temp;

    }

    set(index, value) {

        let temp = this.get(index);

        if (temp) {

            temp.value = value;

            return true;

        }

        return false;

    }

    insert(index, value) {

        if (index < 0 || index > this.length) return false;

        if (index === this.length) return this.push(value);

        if (index === 0) return this.unshift(value);

        const newNode = new Node(value);

        const before = this.get(index - 1);

        const after = before.next;

        before.next = newNode;

        newNode.prev = before;

        newNode.next = after;

        after.prev = newNode;

        this.length++;

        return true;

    }

    remove(index) {

        if (index === 0) return this.shift();

        if (index === this.length - 1) return this.pop();

        if (index < 0 || index >= this.length) return undefined;

        const temp = this.get(index);

        temp.prev.next = temp.next;

        temp.next.prev = temp.prev;

        temp.next = null;

        temp.prev = null;

        this.length--;

        return temp;

    }

}