class Nodo {

constructor(valor) {

this.valor = valor;

this.siguiente = null;

this.anterior = null;

}

}

class ListaCircularDobleEnlazada {

constructor() {

this.cabeza = null;

this.longitud = 0;

}

**// Método para agregar un elemento a la cabeza de la lista**

agregarALaCabeza(valor) {

const nuevoNodo = new Nodo(valor);

if (!this.cabeza) {

nuevoNodo.siguiente = nuevoNodo;

nuevoNodo.anterior = nuevoNodo;

this.cabeza = nuevoNodo;

} else {

nuevoNodo.siguiente = this.cabeza;

nuevoNodo.anterior = this.cabeza.anterior;

this.cabeza.anterior.siguiente = nuevoNodo;

this.cabeza.anterior = nuevoNodo;

this.cabeza = nuevoNodo;

}

this.longitud++;

}

**// Método para agregar un elemento a la cola de la lista**

agregarALaCola(valor) {

const nuevoNodo = new Nodo(valor);

if (!this.cabeza) {

nuevoNodo.siguiente = nuevoNodo;

nuevoNodo.anterior = nuevoNodo;

this.cabeza = nuevoNodo;

} else {

nuevoNodo.siguiente = this.cabeza;

nuevoNodo.anterior = this.cabeza.anterior;

this.cabeza.anterior.siguiente = nuevoNodo;

this.cabeza.anterior = nuevoNodo;

}

this.longitud++;

}

**// Método para agregar un elemento en una posición específica de la lista**

agregarPorPosicion(valor, posicion) {

if (posicion < 0 || posicion > this.longitud) {

console.error("Posición inválida");

return;

}

if (posicion === 0) {

this.agregarALaCabeza(valor);

return;

}

if (posicion === this.longitud) {

this.agregarALaCola(valor);

return;

}

const nuevoNodo = new Nodo(valor);

let nodoActual = this.cabeza;

for (let i = 0; i < posicion - 1; i++) {

nodoActual = nodoActual.siguiente;

}

nuevoNodo.siguiente = nodoActual.siguiente;

nuevoNodo.anterior = nodoActual;

nodoActual.siguiente.anterior = nuevoNodo;

nodoActual.siguiente = nuevoNodo;

this.longitud++;

}

}

**// Método para agregar una lista de elementos a la lista**

agregarPorContenido(lista) {

if (!lista || lista.length === 0) {

return;

}

let nodoActual = null;

lista.forEach(valor => {

const nuevoNodo = new Nodo(valor);

if (!this.cabeza) {

nuevoNodo.siguiente = nuevoNodo;

nuevoNodo.anterior = nuevoNodo;

this.cabeza = nuevoNodo;

nodoActual = nuevoNodo;

} else {

nuevoNodo.siguiente = nodoActual.siguiente;

nuevoNodo.anterior = nodoActual;

nodoActual.siguiente.anterior = nuevoNodo;

nodoActual.siguiente = nuevoNodo;

nodoActual = nuevoNodo;

}

this.longitud++;

});

}

}

**// Método para calcular el promedio de los valores en la lista**

calcularPromedio() {

let suma = 0;

let nodoActual = this.cabeza;

do {

suma += nodoActual.valor;

nodoActual = nodoActual.siguiente;

} while (nodoActual !== this.cabeza);

return suma / this.longitud;

}

**// Método para pasar el último nodo a la cabeza y la cabeza al último**

intercambiarCabezaYCola() {

if (this.longitud < 2) {

return;

}

const antiguaCabeza = this.cabeza;

const nuevaCabeza = this.cabeza.anterior;

this.cabeza = antiguaCabeza.anterior;

antiguaCabeza.anterior = nuevaCabeza.anterior;

nuevaCabeza.anterior = antiguaCabeza;

nuevaCabeza.siguiente = antiguaCabeza.siguiente;

antiguaCabeza.siguiente = nuevaCabeza;

}

}

**Eliminar por contenido**

eliminarPorContenido(valor) {

if (!this.cabeza) {

return;

}

let nodoActual = this.cabeza;

do {

if (nodoActual.valor === valor) {

if (this.longitud === 1) {

this.cabeza = null;

} else {

nodoActual.anterior.siguiente = nodoActual.siguiente;

nodoActual.siguiente.anterior = nodoActual.anterior;

if (nodoActual === this.cabeza) {

this.cabeza = nodoActual.siguiente;

}

}

this.longitud--;

return;

}

nodoActual = nodoActual.siguiente;

} while (nodoActual !== this.cabeza);

}

**// Suponiendo que tenemos una lista circular doblemente enlazada llamada lista**

function insertarDespuesDeValor(lista, valorBuscado, nuevoValor) {

if (!lista.cabeza) {

return;

}

let nodoActual = lista.cabeza;

do {

if (nodoActual.valor === valorBuscado) {

const nuevoNodo = new Nodo(nuevoValor);

nuevoNodo.siguiente = nodoActual.siguiente;

nuevoNodo.anterior = nodoActual;

nodoActual.siguiente.anterior = nuevoNodo;

nodoActual.siguiente = nuevoNodo;

lista.longitud++;

return;

}

nodoActual = nodoActual.siguiente;

} while (nodoActual !== lista.cabeza);

}

// Ejemplo de uso:

insertarDespuesDeValor(lista, 3, 10); // Inserta el valor 10 después del primer nodo con valor 3}

**encontrarValorMaximo() {**

if (!this.cabeza) {

return null;

}

let maximo = this.cabeza.valor;

let nodoActual = this.cabeza.siguiente;

while (nodoActual !== this.cabeza) {

if (nodoActual.valor > maximo) {

maximo = nodoActual.valor;

}

nodoActual = nodoActual.siguiente;

}

return maximo;

}

- encontrarValorMinimo() {

if (!this.cabeza) {

return null;

}

let minimo = this.cabeza.valor;

let nodoActual = this.cabeza.siguiente;

while (nodoActual !== this.cabeza) {

if (nodoActual.valor < minimo) {

minimo = nodoActual.valor;

}

nodoActual = nodoActual.siguiente;

}

return minimo;

}

}

**eliminarNodosRepetidos() {**

if (!this.cabeza) {

return;

}

const valoresVistos = new Set();

let nodoActual = this.cabeza;

do {

if (valoresVistos.has(nodoActual.valor)) {

nodoActual.anterior.siguiente = nodoActual.siguiente;

nodoActual.siguiente.anterior = nodoActual.anterior;

if (nodoActual === this.cabeza) {

this.cabeza = nodoActual.siguiente;

}

this.longitud--;

} else {

valoresVistos.add(nodoActual.valor);

}

nodoActual = nodoActual.siguiente;

} while (nodoActual !== this.cabeza);

}

}

eliminarNodosPares() {

if (!this.cabeza) {

return;

}

let indice = 0;

let nodoActual = this.cabeza;

do {

if (indice % 2 === 0) {

nodoActual.anterior.siguiente = nodoActual.siguiente;

nodoActual.siguiente.anterior = nodoActual.anterior;

if (nodoActual === this.cabeza) {

this.cabeza = nodoActual.siguiente;

}

this.longitud--;

}

indice++;

nodoActual = nodoActual.siguiente;

} while (nodoActual !== this.cabeza);

}

**eliminarNodosImpares() {**

if (!this.cabeza) {

return;

}

let indice = 0;

let nodoActual = this.cabeza;

do {

if (indice % 2 !== 0) {

nodoActual.anterior.siguiente = nodoActual.siguiente;

nodoActual.siguiente.anterior = nodoActual.anterior;

if (nodoActual === this.cabeza) {

this.cabeza = nodoActual.siguiente;

}

this.longitud--;

}

indice++;

nodoActual = nodoActual.siguiente;

} while (nodoActual !== this.cabeza);

}

}

**encontrarNodoMedio() {**

if (!this.cabeza) {

return null;

}

let punteroLento = this.cabeza;

let punteroRapido = this.cabeza;

while (punteroRapido.siguiente !== this.cabeza && punteroRapido.siguiente.siguiente !== this.cabeza) {

punteroLento = punteroLento.siguiente;

punteroRapido = punteroRapido.siguiente.siguiente;

}

return punteroLento;

}

}

**duplicarNodos() {**

if (!this.cabeza) {

return;

}

let nodoActual = this.cabeza;

do {

const nuevoNodo = new Nodo(nodoActual.valor); // Crear un nuevo nodo con el mismo valor

nuevoNodo.siguiente = nodoActual.siguiente;

nuevoNodo.anterior = nodoActual;

nodoActual.siguiente.anterior = nuevoNodo;

nodoActual.siguiente = nuevoNodo;

nodoActual = nuevoNodo.siguiente;

this.longitud++;

} while (nodoActual !== this.cabeza);