EJERCICIOS

PROFESOR: Alejandro Lara Caballero
ALUMNO: Leonardo Aguilar Martínez
MATRICULA: 2203025005

ESTRUCTURA DE DATOS LINEALES

EJERCICIO 1:

Escribe una función dentro de la clase que obtenga el nésimo nodo en una Lista ligada . La función recibe de entrada un índice entero y regresa el nodo correspondiente a dicha posición.

```
/*----------------------Funciones agregadas-----------
//Inciso 1
Nodo* ListaEnlazadaSimple::regresoNodoN(int x){
    Nodo* nodoEncontrado = buscar(x);
    return nodoEncontrado;
}
```

EXPLICACIÓN:

La función recibe como parametro un entero X, dentro del procedimiento declaramos un nodo de tipo Nodo llamado nodoEncontrado, el cual será igualado a la función buscar, a esta le pasamos como parametro el entero X, esto nos da como resultado el nodo buscado.
Con esta función podemos reutilizar la función buscar para hacer más corto el código.

EJERCICIO 2:

Desarrolla una función dentro de la clase que cuente el número de ocurrencias de un elemento en una lista ligada. Por ejemplo si la lista es L1={1,2,3,4,5,4,12}, al recibir como entrada 4, debe regresar 2.

```
int ListaEnlazadaSimple::ocurrencias(int x){
    Nodo* aux = cabeza;
    int ocurrencia;
    while(aux!= nullptr){
        if(aux->elemento == x){
            ocurrencia++;
        }
    }
    return ocurrencia;
}
```

EXPLICACIÓN:

La función recibe un entero X el cual veremos cuantas veces se repite dentro de la lista. Es por eso que recorremos la lista con un nodo llamado aux, desde la cabeza hasta el final, si alguno de esos nodos tiene dentro el valor del entero X, sumaremos a un iterador llamado ocurrencia un 1, por lo tanto, al final, retornamos cuantas veces se repite ese número en la lista.

EJERCICIO 8:

Escribe una función dentro de la estructura que permita eliminar todas las repeticiones del elemento x en una lista simplemente ligada. NO USES el método eliminar. Bórralos conforme los vas encontrando.

```
void ListaEnlazadaSimple::eliminarOcurrencia(int x){
    Nodo* aux = cabeza;
    while(aux!=nullptr){
        int ocurrencia = 0;

        if(aux->elemento = x){
            ocurrencia=1;
        }
        if(ocurrencia == 1){
            delete aux;
        }
        aux=aux->sig;
    }
}
```

EXPLICACIÓN:

Recibe como parametro, un entero X que será el dato que buscaremos. Empezamos a recorrer la lista desde la cabeza, utilizaremos una variable ocurrencia para detectar si se repite más de una vez un dato, si es así, lo borraremos, todo se repite hasta el ultimo elemento de la lista.

EJERCICIO 6:

Realiza en el main una función que dadas dos listas: I y J, traslada a la lista I todos los enteros almacenados en la lista J , de modo que los elementos trasladados se agregan al final de la lista I en el orden en el que aparecían en la lista y esta última queda vacía

```
void enviarDatos(ListaEnlazadaSimple *Lista_I, ListaEnlazadaSimple *Lista_J ){
    Nodo *inicio = Lista_J->cabeza;
    while(inicio != nullptr){
        Lista_I->inserta_final(inicio->elemento);
        Lista_J->eliminar(inicio);
    }
    return Lista_I;
}
```

EXPLICACIÓN:

Recibe como parametro dos listas. Se inicia recorriendo desde la cabeza la lista J, por cada iteración, mandamos a llamar el metodo insertar final de la lista I y como parametro le colocamos el nodo iterado de la lista J, después, eliminamos de la lista J ese elemento.
Dicho de otra forma: por cada repetición, insertamos un nodo de la lista J en la lista I y borramos el nodo para dejar vacio J.