Práctica de Laboratorio N° 04:
Eliminación en Listas Doblemente Enlazadas

I. OBJETIVOS DE LA PRÁCTICA:

- Implementar funciones de eliminación en una lista doblemente enlazada, considerando diferentes posiciones relativas al nodo objetivo: al inicio, al final, con información específica, anterior o posterior a un valor dado.
- Comprender el manejo correcto de punteros dobles (ligaizq y ligader) durante las operaciones de eliminación, garantizando la integridad estructural de la lista en cada caso.
- Consolidar los conocimientos adquiridos mediante la resolución de ejercicios prácticos, asegurando que las listas se mantengan correctamente enlazadas después de cada operación.

II. DESARROLLO DE LA PRÁCTICA:

1. Implementación de funciones de eliminación en una lista doblemente enlazada en C++:
   En esta práctica se continuará con el desarrollo de la lista doblemente enlazada iniciado en la Práctica N.° 03, donde se implementaron las funciones de recorrido e inserción. Ahora, se deben añadir al mismo archivo las funciones necesarias para eliminar nodos en diferentes posiciones de la lista, según los siguientes casos:
   - Eliminar el primer nodo
   - Eliminar el último nodo
   - Eliminar el nodo con información X
   - Eliminar el nodo anterior al nodo con información X
   - Eliminar el nodo posterior al nodo con información X

```
160  //Elimina el primer nodo de una lista doblemente enlazada
161  void elimina_inicio(Nodo *&p, Nodo *&f) {
162      Nodo *q = p;
163
164      if (q->ligader!=NULL) {
165          p = q->ligader;
166          p->ligaizq = NULL;
167      } else {
168          p = NULL;
169          f = NULL;
170      }
171      delete(q);
172  }
173
174  //Elimina el último nodo de una lista doblemente enlazada
175  void elimina_ultimo(Nodo *&p, Nodo *&f) {
176      Nodo *q = f;
177
178      if (q->ligaizq!=NULL) {
179          f = q->ligaizq;
180          f->ligader = NULL;
181      } else {
182          p = NULL;
183          f = NULL;
184      }
185      delete(q);
186  }
187
```

```
188 //Elimina un nodo con información X
189 void elimina_x(Nodo *&p, Nodo *&f, int x) {
190     Nodo *q = p;
191 
192     while (q->ligader != NULL && q->info != x){
193         q=q->ligader;
194     }
195 
196     if (q->info == x) {
197         if (q == p && q == f) { //La lista tiene un solo nodo
198             p = NULL;
199             f = NULL;
200         } else {
201             if (q == p) {   //El nodo con información x es el primero de la lista
202                 p = q->ligader;
203                 p->ligaizq = NULL;
204             } else {
205                 if (q == f) {   //El nodo con información x es el último de la lista
206                     f = q->ligaizq;
207                     f->ligader = NULL;
208                 } else {    //Es nodo con información x es un nodo intermedio
209                     Nodo *t = NULL;
210                     Nodo *r = NULL;
211                     t = q->ligaizq;
212                     r = q->ligader;
213                     t->ligader = r;
214                     r->ligaizq = t;
215                 }
216             }
217         }
218         delete(q);
219     } else {
220         cout << "El elemento con información " << x << " no se encuentra en la lista." << endl;
221         return;
222     }
223 
224 }
```

```
226 //Elimina el nodo anterior al nodo con información X
227 void elimina_antes_x(Nodo *&p, Nodo *&f, int x) {
228     Nodo *q = p;
229 
230     while (q->ligader != NULL && q->info != x){
231         q=q->ligader;
232     }
233 
234     if (q->info == x) {
235         if (p == q) {   //El nodo con información x es el primero de la lista
236             cout << "No existe nodo anterior al primero." << endl;
237             return;
238         } else {
239             Nodo *t = NULL;
240             t = q->ligaizq;
241             if (p == t) {   //El nodo a eliminar es el primer nodo de la lista
242                 p = q;
243                 p->ligaizq = NULL;
244             } else {
245                 Nodo *r = NULL;
246                 r = t->ligaizq;
247                 q->ligaizq = r;
248                 r->ligader = q;
249             }
250             delete(t);
251         }
252 
253     } else {
254         cout << "El elemento con información " << x << " no se encuentra en la lista." << endl;
255         return;
256     }
257 }
258 
```

2. Implementa una función que elimine el nodo posterior al nodo con información X:

   void elimina_despues_X(Nodo *&p, Nodo *&f, int x)

3. Implementa una función que mueva el menor elemento de una lista a la primera posición:

   void MoverMenor(Nodo *&p, Nodo *&f)

Ejemplo:
Lista original:
[20] <-> [15] <-> [8] <-> [30] <-> [12]
Después de ejecutar MoverMenor(), la lista debe quedar:
[8] <-> [20] <-> [15] <-> [30] <-> [12]
En este caso, el valor más bajo es 8 y ha sido movido al inicio de la lista, manteniendo el orden de los demás nodos.

4. Implementa una función que mueva el mayor elemento de una lista a la última posición:

   void MoverMayor(Nodo *&p, Nodo *&f)

   Ejemplo:
   Lista original:
   [10] <-> [25] <-> [18] <-> [40] <-> [22]
   Después de ejecutar MoverMayor(), la lista debe quedar:
   [10] <-> [25] <-> [18] <-> [22] <-> [40]
   En este caso, el valor más alto es 40 y ha sido movido al final de la lista, manteniendo el orden de los demás nodos.

5. Desarrolle un método que elimine los elementos duplicados en una lista doblemente enlazada ordenada:
   void eliminarRepetidos()

   Ejemplo:

   Lista original:
   [10] <-> [10] <-> [15] <-> [20] <-> [20] <-> [25]

   Después de ejecutar eliminarRepetidos():
   [10] <-> [15] <-> [20] <-> [25]
   Se eliminaron los duplicados de 10 y 20

6. Desarrolle un método que elimine todos los nodos que contengan un valor específico en una lista doblemente enlazada
   void eliminarCoincidentes(int x)

   Ejemplo 1:

   Lista original:
   5 <-> 3 <-> 7 <-> 3 <-> 9
   Valor a eliminar: 3
   Lista resultante:
   5 <-> 7 <-> 9

   Ejemplo 2:

   Lista original:
   2 <-> 2 <-> 2 <-> 4 <-> 5
   Valor a eliminar: 2
   Lista resultante:
   4 <-> 5

NOTA: Los ejercicios 4, 5 y 6 deben ser entregados en pseudocódigo escrito a mano el día martes 6 de mayo, al inicio de la clase.