UNIVERSIDAD AUTÓNOMA GABRIEL RENÉ MORENO

FACULTAD DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN Y TELECOMUNICACIONES

LAB-5. LISTAS DOBLEMENTE ENCADENADAS.

TERMINADO: 95%

Trabajo en parejas

Nombre	Registro	Grupo
Choque Calle Jorge	219074240	12
Gedalge Cayhuara Cristian Gabriel	219022062	14

Santa Cruz – Bolivia
2024

```
public class Lista {

    public Nodo prim;
    public Nodo ult;
    public int cantElem;

    public Lista() {
        prim = ult = null;
        this.cantElem = 0;
    }
//    1. L1.insertarIesimo(x, i) : Método que inserta el elemento x, en la posición i, de la lista L1.

    public void insertarIesimo(int x, int i) {
        int k = 0;
        Nodo p = prim, ap = null;
        while (k < i && p != null) {
            ap = p;
            p = p.prox;
            k = k + 1;

        }
        insertarNodo(ap, p, x);
    }
//2. L1.insertarPrim(x) : Método que insertar el elemento x, al inicio de la lista L1.

    public void insertarPrim(int x) {
        if (vacia()) {
            prim = ult = new Nodo(null, x, null);

        } else {
            prim = prim.ant = new Nodo(null, x, prim);

        }
        cantElem = cantElem + 1;
    }
//3. L1.insertarUlt(x) : Método que inserta el elemento x, al final de la lista L1.

    public void insertarUltm(int x) {
        if (vacia()) {
            prim = ult = new Nodo(null, x, null);
        } else {
            ult = ult.prox = new Nodo(ult, x, null);
```

```
        }
        cantElem = cantElem + 1;
    }
//4. L1.insertarLugarAsc(x) : Método que inserta el elemento x, en su lugar correspondiente en la Lista ordenadas de menor a mayor.

    public void insertarLugarAsc(int x) {
        Nodo p = prim, ap = null;
        while (p != null && p.elem < x) {
            ap = p;
            p = p.prox;

        }
        insertarNodo(ap, p, x);
    }
//5. L1.insertarLugarDes(x) : Método que inserta el elemento x, en su lugar correspondiente en la Lista ordenadas de mayor a menor.

    public void insertarLugarDes(int x) {
        Nodo p = prim, ap = null;
        while (p != null && p.elem > x) {
            ap = p;
            p = p.prox;

        }
        insertarNodo(ap, p, x);
    }
//6. L1.insertarIesimo(L2, i) : Método que insertar los elementos de la lista L2 en la lista L1, desde la posición i.

    public void insertarIesimo(Lista L2, int i) {

        Nodo p = L2.prim;
        while (p != null) {

            insertarIesimo(p.elem, i);
            p = p.prox;
            i++;

        }
    }
//7. L1.insertarPrim(L2) : Método que insertar los elementos de la lista L2 al principio de la lista L1.
//

    public void insertarPrim(Lista L2) {
        Nodo p = L2.prim, ap = null;
        int i = 0;
```

```
        while (p != null) {
            insertarIesimo(p.elem, i);
            p = p.prox;
            i++;
        }
    }
//8. L1.insertarUlt(L2) : Método que insertar los elementos de la lista L2 al
final de la lista L1.
//

    public void insertarUlt(Lista L2) {
        Nodo p = L2.prim, ap = null;
        int i = cantElem;
        while (p != null) {
            insertarIesimo(p.elem, i);
            p = p.prox;
            i++;
        }
    }

//9. L1.iguales() : Método Lógico que devuelve True, si todos los
elementos de la lista L1 son iguales.
//
    public boolean iguales() {
        Nodo p = prim;
        while (p.prox != null) {
            if (p.elem != p.prox.elem) {
                return false;
            }
            p = p.prox;
        }
        return true;
    }
//10. L1.diferentes() : Método Lógico que devuelve True, si todos los
elementos de la lista L1 son diferentes.
//

     public Boolean Diferentes() {
        Nodo p = this.prim;
        while (p.prox != null) {
            if (p.elem == p.prox.elem) {
                return false;
            }
        }

        return true;
    }
```

```
//11. L1.mayorElem() : Método que devuelve el mayor elemento de la
lista L1.
//
   public int mayorElem() {
      Nodo p = prim;
      int numero = p.elem;
      while (p != null) {
         if (p.elem > numero) {
            numero = p.elem;
         }
         p = p.prox;
      }
      return numero;
   }
//12. L1.ordenado()  : Método Lógico que devuelve True, si todos los
elementos de la lista L1 están ordenados en forma ascendente o
descendente.
//

   public int menorElem() {
      Nodo p = prim;
      int numero = p.elem;
      while (p != null) {
         if (p.elem < numero) {
            numero = p.elem;
         }
         p = p.prox;
      }
      return numero;
   }

//13. L1.indexOf(x) : Método que devuelve la posición (lugar) de la
primera ocurrencia del elemento x. Si x no se encuentra en la lista L1, el
método devuelve –
//



      public int indexOf(int x) {
      Nodo actual = prim;
      int index = 0;
      while (actual != null) {
         if (actual.elem == x) {
            return index;
         }
         actual = actual.prox;
         index++;
      }
```

```
        return -1;
    }
```

//14. L1.lastIndexOf(x) : Método que devuelve la posición (lugar) de la última ocurrencia del elemento x. Si x no se encuentra en la lista L1, el método devuelve –
//

```
    public int lastIndexOf(int x) {
    Nodo actual = prim;
    int lastIndex = -1;
    while (actual != null) {
        if (actual.elem == x) {
            lastIndex++;
        }
        actual = actual.prox;
    }
    return lastIndex;
    }
```

//15. L1.palindrome() : Método lógico que devuelve True, si la lista L1 contiene elementos que forma un palíndrome.

```
  @Override
    public String toString() {
```

```
        String s1 = "[";
        Nodo p = prim;
        while (p != null) {
            s1 = s1 + p.elem;
            if (p.prox != null) {
                s1 = s1 + "  ";
            }
            p = p.prox;
        }
        return s1 + "]";
    }

    public void insertarNodo(Nodo ap, Nodo p, int x) {
        if (ap == null) {
            insertarPrim(x);
        } else if (p == null) {
            insertarUltm(x);
        } else {
            ap.prox = p.ant = new Nodo(ap, x, p);
            this.cantElem++;
        }
    }

    public boolean vacia() {
        return this.cantElem == 0;
    }

    public void insertarLugar(int x) {
        Nodo p = prim, ap = null;
        while (p != null && x > p.elem) {
            ap = p;
            p = p.prox;

        }
        insertarNodo(ap, p, x);

    }

    public static void main(String[] args) {
        Lista L1 = new Lista();
        L1.insertarIesimo(4, 0);
        L1.insertarPrim(80);
        L1.insertarUltm(5);
        L1.insertarUltm(10);
//        L1.insertarUltm(17);
//        L1.insertarUltm(22);
     //   L1.insertarLugarAsc(6);
        Lista L2 = new Lista();
```

```
        L2.insertarIesimo(6, 0);
        L2.insertarUltm(5);

        System.out.println(L1);

    }
}
```