```
package examenagosto2021;

public class Solucion implements ILista {

    private Nodo raiz;

  public Solucion() {
     this.raiz = null;
  }

  public Nodo getRaiz() {
     return raiz;
  }

  public void setRaiz(Nodo raiz) {
     this.raiz = raiz;
  }

@Override
  public int mayor() {
     if (!vacia()) {
        int may = raiz.info;
        Nodo reco = raiz.sig;
        while (reco != null) {
           if (reco.info > may) {
```

```
                may = reco.info;
            }
            reco = reco.sig;
        }
        return may;
    } else {
        return Integer.MAX_VALUE;
    }
  }
@Override
  public int posMayor() {
    if (!vacia()) {
        int may = raiz.info;
        int x = 1;
        int pos = x;
        Nodo reco = raiz.sig;
        x=x+1;              // agregar esta linea
        while (reco != null) {
            if (reco.info > may) {
                may = reco.info;
                pos = x;
            }
            reco = reco.sig;
            x++;
        }
        return pos;
    } else {
```

```
            return Integer.MAX_VALUE;
        }
    }
@Override
    public boolean existe(int x) {
        Nodo reco = raiz;
        while (reco != null) {
            if (reco.info == x) {
                return true;
            }
            reco = reco.sig;
        }
        return false;
    }
@Override
    public boolean vacia() {
        if (raiz == null) {
            return true;
        } else {
            return false;
        }
    }
@Override
    public void imprimir() {
        Nodo reco = raiz;
        while (reco != null) {
            System.out.print(reco.info + "-");
```

```
            reco = reco.sig;
        }
        System.out.println();
    }

@Override
    public int cantidad() {
        int cant = 0;
        Nodo reco = raiz;
        while (reco != null) {
            reco = reco.sig;
            cant++;
        }
        return cant;
    }

    @Override
    public int[] convertilistaenarray(Lista l) {
            int dimensionvector;
            if (!l.vacia()){
              dimensionvector = l.cantidad();
            }else{dimensionvector=0;}

            int [] vec = new int[dimensionvector];
            Nodo aux = l.getRaiz();
            for (int i = 0; i < dimensionvector; i++) {
               vec[i] = aux.getInfo();
```

```
            aux = aux.getSig();


        }



    return vec;
  }

@Override
  public boolean ordenada() {
    if (cantidad() > 1) {
      Nodo reco1 = raiz;
      Nodo reco2 = raiz.sig;
      while (reco2 != null) {
        if (reco2.info < reco1.info) {
          return false;
        }
        reco2 = reco2.sig;
        reco1 = reco1.sig;
      }
    }
    return true;
  }
```

```
@Override
  public void insertar(int pos, int x) {
  if (pos <= cantidad() + 1) {
        Nodo nuevo = new Nodo(pos);
        nuevo.info = x;
        if (pos == 1) {
           nuevo.sig = raiz;
           raiz = nuevo;
        } else if (pos == cantidad() + 1) {
           Nodo reco = raiz;
           while (reco.sig != null) {
              reco = reco.sig;
           }
           reco.sig = nuevo;
           nuevo.sig = null;
        } else {
           Nodo reco = raiz;
           for (int f = 1; f <= pos - 2; f++) {
              reco = reco.sig;
           }
           Nodo siguiente = reco.sig;
           reco.sig = nuevo;
           nuevo.sig = siguiente;
        }
     }
  }
```

```
  @Override
  public Lista unirlistas(Lista l1, Lista l2) {
    Lista l3=new Lista();
     int i=1;


     Nodo aux1=l1.getRaiz();
     Nodo aux2=l2.getRaiz();
     while (aux1!=null){
        l3.insertar(i, aux1.getInfo());
        i=i+1;
        aux1=aux1.sig;
     }
     while (aux2!=null){
             l3.insertar(i, aux2.getInfo());
        i=i+1;
        aux2=aux2.sig;
     }
     return l3;
  }
@Override
  public int extraer(int pos) {
     if (pos <= cantidad()) {
        int informacion;
        if (pos == 1) {
           informacion = raiz.info;
           raiz = raiz.sig;
        } else {
```

```
            Nodo reco;
            reco = raiz;
            for (int f = 1; f <= pos - 2; f++) {
               reco = reco.sig;
            }
            Nodo prox = reco.sig;
            reco.sig = prox.sig;
            informacion = prox.info;
         }
         return informacion;
      } else {
         return Integer.MAX_VALUE;
      }
   }

@Override
   public void intercambiar(int pos1, int pos2) {
      if (pos1 <= cantidad() && pos2 <= cantidad()) {
         Nodo reco1 = raiz;
         for (int f = 1; f < pos1; f++) {
            reco1 = reco1.sig;
         }
         Nodo reco2 = raiz;
         for (int f = 1; f < pos2; f++) {
            reco2 = reco2.sig;
         }
         int aux = reco1.info;
```

```
            reco1.info = reco2.info;
            reco2.info = aux;
        }
    }
    @Override
    public void ordenarlista() {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
    }

}
```