```
package ll_exercises;

import java.util.NoSuchElementException;
import linkedlists.*;
import java.util.Iterator;


public class ListManipulatorImpl<E extends Comparable<E>>
                        implements ListManipulatorIF<E> {

  // SingleLinkedListIF<E> l;

   @Override
   public int size(SingleLinkedListIF l) {
      //l= new SingleLinkedListImpl();
      Iterator<E> iter= l.iterator();
      //int i=0;
      int size=0;
      while(iter.hasNext()){
         size++;
         iter.next();
      }
      return size;
      //throw new UnsupportedOperationException("Not supported yet.");
   }

   @Override
   public boolean sameSame(SingleLinkedListIF l1, SingleLinkedListIF l2) {
      Iterator<E> iter1= l1.iterator();
      Iterator<E> iter2=l2.iterator();
      int size1=0;
      int size2=0;
//       while (iter1.hasNext()) {
//          size1++;
//          iter1.next();
//       }
//       while(iter2.hasNext()){
//         size2++;
//         iter2.next();
//       }
      return (iter1.next()==iter2.next());


   }

   @Override
```

```
  public boolean sublist(SingleLinkedListIF<E> l1, SingleLinkedListIF<E> l2) {
    SingleLinkedListIterator<E> iter1= (SingleLinkedListIterator<E>)l1.iterator();
    SingleLinkedListIterator<E> iter2=(SingleLinkedListIterator<E>)l2.iterator();
    SingleLinkedListIterator<E> iter3=(SingleLinkedListIterator<E>)l1.iterator();
    SingleLinkedListIterator<E> iter4=(SingleLinkedListIterator<E>)l2.iterator();

    int size1=this.size(l1);
    int size2=this.size(l2);
    int i=0;
    int k=0;

    //Object el=l1.removeFirst();
    //Object checkEl=l2.removeFirst();
    //E stuff1 = iter1.next();
    while(k<=size2 && iter3.hasNext()){

      while((iter1.hasNext()&& iter2.hasNext())){
        System.out.println("O");
        k++;
        int counter=0;
        E stuff1 = iter1.next();
        E stuff2 = iter2.next();

        if(stuff1.equals(stuff2)){
          while((iter3.hasNext()&& iter4.hasNext()) &&
iter3.next().equals(iter4.next())== true ){
            System.out.println("X");
            k++;
            i++;
            //iter3.next();
            //iter4.next();
          }
        }
        else {
          while(iter2.hasNext() && iter2.next()!=stuff1)
            counter++;
          if (counter>size1){
            return false;

          }
          else{
            for(int j=0;j<counter;j++){
              iter4.next();
              System.out.println("EEE");
            }
            iter4.next();
```

```
                while((iter3.hasNext()&& iter4.hasNext()) &&
(iter3.next().equals(iter4.next())==true)){
                    System.out.println("X");
                    k++;
                    i++;

                }

            }

        }

      }
    }
    System.out.println(k);

    System.out.println(i);
    return(i==size1);

    //throw new UnsupportedOperationException("Not supported yet.");
  }


  @Override
  public void feed(SingleLinkedListIF l1, SingleLinkedListIF l2) throws
NoSuchElementException {
    if (l1.isEmpty())
      throw new NoSuchElementException();
    l2.insertFirst(l1.removeFirst());
    l2.display();
    //throw new UnsupportedOperationException("Not supported yet.");
  }

  @Override
  public void superFeed(SingleLinkedListIF l1, SingleLinkedListIF l2, int n) throws
NoSuchElementException {
    if(l1.isEmpty())
      throw new NoSuchElementException();
    SingleLinkedListIF<E> temp= new SingleLinkedListImpl();

    for(int i=0; i<n; i++){
      temp.insertFirst((E) l1.removeFirst());
  }
    for(int k=0;k<n;k++)
      l2.insertFirst(temp.removeFirst());
    l2.display();
```

```
    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public SingleLinkedListIF diff(SingleLinkedListIF l1, SingleLinkedListIF l2) {
    SingleLinkedListIF<E> temp= new SingleLinkedListImpl();
    int size=this.size(l1);
    int sizeTemp=this.size(temp);
    int i=0;
    while(i<size){
        Object el = l1.removeFirst();
        i++;
        if(l2.find(el)==-1){
            sizeTemp++;
            temp.insertFirst((E) el);
        }

    }
    int k=0;
    while(k<sizeTemp){
        l1.insertFirst(temp.removeFirst());
        k++;
    }
    l1.display();
    return l1;
    //throw new UnsupportedOperationException("Not supported yet.");
}

@Override
public int delDiff(SingleLinkedListIF l1, SingleLinkedListIF l2) {
    int counter=0;
    int size=this.size(l1);
    int i=0;
    while(i<size){
        Object el=l1.removeFirst();
        i++;
        while(l2.find(el)!=-1){
            counter++;
            l2.delete(el);
        }
    }
    System.out.println(counter);
    return counter;
    //throw new UnsupportedOperationException("Not supported yet.");
}
```

```
}
```