```java
// Implementor of ADT
public class LinkedList<T> {
    private Node<T> head;
    private Node<T> current;

    public void findPrevious() {
        Node<T> tmp = head;
        while(tmp.next != current)
            tmp = tmp.next;
        current = tmp;
    }

    public void findLast() {
        while(current.next != null)
            current = current.next;
    }

    public void display() {
        Node<T> tmp = head;
        while(tmp != null) {
            System.out.println(tmp.data);
            tmp = tmp.next;
        }
    }

    public boolean find(T x) {
        Node<T> tmp = current;
        current = head;
        while(current != null) {
            if(current.data.equals(x))
                return true;
            current = current.next;
        }
        current = tmp;
        return false;
    }

    public void swap(int i, int j) {
        Node<T> tmpI = null;
        Node<T> tmpJ = null;
        int count = 0;

        Node<T> tmp = head;
        while(tmp != null) {
            if(count == i)
                tmpI = tmp;
            if(count == j)
                tmpJ = tmp;
            count++;
            tmp = tmp.next;
        }

        if(tmpI != null && tmpJ != null) {
            T x = tmpI.data;
            tmpI.data = tmpJ.data;
            tmpJ.data = x;
        }
    }
}

public class ArrayList<T> {
    private T nodes[];
    private int size;
    private int maxsize;
    private int current;

    public void findPrevious() {
        current--;
    }

    public void findLast() {
        current = size - 1;
    }
    public void display() {
        for(int i = 0; i < size; i++)
            System.out.println(nodes[i]);
    }
}

// User of ADT
class MyClass {
    public static<T> void display(List<T> l) {
        if(!l.empty()) {
            l.findFirst();
            while(!l.last()) {
                System.out.println(l.retrieve());
                l.findNext();
            }
            System.out.println(l.retrieve());
        }
    }

    public static<T> void removeEqual(List<T> l, T x) {
        if(!l.empty()) {
            l.findFirst();
            while(!l.last()) {
                if(l.retrieve().equals(x))
                    l.remove();
                else
                    l.findNext();
            }
            if(l.retrieve().equals(x))
                l.remove();
        }
    }

    public static<T> boolean find(List<T> l, T x) {
        if(!l.empty()) {
            l.findFirst();
            while(!l.last()) {
                if(l.retrieve().equals(x))
                    return true;
                l.findNext();
            }
            if(l.retrieve().equals(x))
                return true;
        }
        return false;
    }

    public static<T> List<T> merge(List<T> l1, List<T> l2) {
        List<T> l3 = new List<T>();

        if(!l1.empty()) {
            l1.findFirst();
            while(!l1.last()) {
                l3.insert(l1.retrieve());
                l1.findNext();
            }
            l3.insert(l1.retrieve());
        }

        if(!l2.empty()) {
            l2.findFirst();
            l3.findFirst();
            while(!l2.last()) {
                l3.insert(l2.retrieve());
                l2.findNext();
                if(!l3.last())
                    l3.findNext();
            }
            l3.insert(l2.retrieve());
        }

        return l3;
    }

    public static<T> void insertFirst(List<T> l, T x) {
        if(l.empty())
            l.insert(x);
        else {
            l.findFirst();
            T e = l.retrieve();
            l.update(x);
            l.insert(e);
        }
    }

    public static<T> void insertLast(List<T> l, T x) {
        if(l.empty()) {
            l.insert(x);
        }
        else {
            while(!l.last()) {
                l.findNext();
            }
            l.insert(x);
        }
    }
}
```