Name : zhenzhen su

Instructor: Dr Reid

Class: CSC 143

#3 Problem statement: write a method isSorted that returns true if list is in non-decreasing order, returns false if otherwise

Code:

```
//return true if list is in increasing order
//false if otherwise
public boolean isSorted(){
    ListNode current = front;
    if(current == null){
        return true;
    }else{
        //loop through each node to check order
        while(current.next != null && current.data <= current.next.data){
            current = current.next;
        }
        //check if reaches the end of list
        if(current.next == null){
            return true;
        }else{
            return false;
        }
    }
}

public static void main(String[] args) {

    LinkedIntList list = new LinkedIntList();
    list.add(1);
    list.add(2);
    list.add(3);
    LinkedIntList list1 = new LinkedIntList();
    list.add(1);
    list.add(0);
    list.add(3);
    LinkedIntList list2 = new LinkedIntList();
    System.out.println("Is the list [1,0,3] sorted? " + list.isSorted());
    System.out.println("Is the list [1,2,3] sorted? " + list1.isSorted());
    System.out.println("Is the list [] sorted? " + list2.isSorted());

}
```

```
Console Output:
Is the list [1,0,3] sorted? false
Is the list [1,2,3] sorted? true
Is the list [] sorted? True
```

#5 problem statement: write a method countDuplicates that returns # of duplicates in a list

```
Code:
```

```java
//countDuplicates return the number of duplicates in a sorted list
public int countDuplicates(){
    int num = 0;
    ListNode current = front;
    //loop through list to find duplicates
    while(current != null && current.next != null){
        //check neighboring nodes
        if(current.data == current.next.data){
            num ++;
        }
        current = current.next;
    }
    return num;
}

public static void main(String[] args) {
       LinkedIntList list3 = new LinkedIntList();
       list3.add(1);
       list3.add(2);
       list3.add(2);
       list3.add(3);
       list3.add(4);
       list3.add(4);
       System.out.println("the number of duplicates are " + list3.countDuplicates());
}
```

Console output:

```
the number of duplicates are 2
```

#14 problem statement: removeAll method that removes all occurrences of a particular value

```java
//removes all occurrences of a given value
public void removeAll(int val){
    ListNode current = front;
    if(current != null){
        //check if there are occurrences at the beginning
        //remove all occurrences at the beginning
        while(current != null && current.next != null && current.data == val){
            front = front.next;
            current = front;
        }

        //check occurrence when there are one node left
        if(current != null && current.data == val){
            front = null;
        }
        //check from the second node to the second to last node
        while(current.next != null && current.next.next != null){
            //check if next node's value is the value to be removed
            if(current.next.data == val){
                //remove the next node
                current.next = current.next.next;
            }else{
                current = current.next;
            }
        }
        //check the last node if it is the value
        if(current.next != null && current.next.data == val){
            current.next = null;
        }
    }
}


//check removeAll
public static void main(String[] args) {

      LinkedIntList list4 = new LinkedIntList();
      list4.add(1);
      list4.add(2);
      list4.add(3);
      list4.add(1);
      list4.add(0);
      list4.add(3);
      System.out.println("The list is after removeAll " + list4.removeAll(3));
}

Console output:
The list is after removeAll [1, 2, 1, 0]
```

#16 problem statement: `removeEvens removes values in even-numbered indexes from a list and return a list that contains the removed values in their original order`

```
Code:
//removes the values in even-numbered indexes from a list
//return a new list of those removed values
public LinkedIntList removeEvens(){
    //special case when it is empty list
    if(front == null){
        return this;
    }

    //create a Linkedlist of removed evens
    LinkedIntList newList = new LinkedIntList();
    newList.front = this.front;
    ListNode list2 = newList.front;

    //modifies the front node of list1
    front = front.next;
    ListNode list1 = front;

    while(list2.next != null && list2.next.next != null
            && list1.next != null && list1.next.next != null ){
        //connect even-numbered nodes for list2
        list2.next = list2.next.next;
        list2 = list2.next;
        //connect odd numbered nodes for lsit1
        list1.next = list1.next.next;
        list1 = list1.next;
    }
    //handle the last nodes for list2
    if(list2.next != null && list2.next.next == null){
        list2.next = null;
    }else if(list2.next != null && list2.next.next != null){
        list2.next = list2.next.next;
    }
    //handle last few nodes for list1
    if(list1.next != null){
        list1.next = null;
    }

    return newList ;
}
public static void main(String[] args) {

       //check removeEvens
       LinkedIntList list5 = new LinkedIntList();
       list5.add(1);
       list5.add(6);
       list5.add(11);
       list5.add(33);
       list5.add(1);
       list5.add(2);
       LinkedIntList list6 = new LinkedIntList();
       list6.add(1);
       list6.add(6);
       list6.add(11);
       list6.add(33);
       list6.add(1);
       System.out.println("The list is after removeEvens is " + list5.removeEvens());
       System.out.println("The original list is after removeEvens is " + list5);

       System.out.println("The list is after removeEvens is " + list6.removeEvens());
       System.out.println("The original list is after removeEvens is " + list6);

}
```

Console output:

```
The list is after removeEvens is [1, 11, 1]

The original list is after removeEvens is [6, 33, 2]

The list is after removeEvens is [1, 11, 1]

The original list is after removeEvens is [6, 33]
```

#21 problem statement: reverse the items in the linkedList

Code:

```
//reverse the list items
public void reverse(){
    ListNode current = front;
    ListNode start = front;

    //case when it is empty list
    if(front != null && front.next != null){

        //insert the last node to the front
        while(current.next.next != null){
            current = current.next;
        }
        //set insertion node to be last node
        ListNode insertion = current.next;
        //insertion node point to the starting node of the list
        insertion.next = start;
        //set the current node to null
        current.next = null;

        //front node points to current node
        front = insertion;
        //current point to start
        current = start;


        //reverse the rest of node
        while(start.next != null){

            //move current to point at second to last node
            while(current.next.next != null){
                current = current.next;
            }
            //point last node to the starting node of the list
            current.next.next = start;

            //insertion node points to current node
            insertion.next = current.next;

            //current node points to null
            current.next = null;

            //update the insertion node to newly inserted node
            insertion = insertion.next;

            //current point to start
            current = start;

        }

    }
}


public static void main(String[] args) {
        LinkedIntList list6 = new LinkedIntList();
               list6.add(1);
               list6.add(6);
               list6.add(11);
               list6.add(33);
       System.out.println("The list [1,6,11,33,1] after reverse is " +
       list6.reverse());
}
```

Console output:

```
The list [1,6,11,33,1] after reverse is [1, 33, 11, 6, 1]
```