IntListB

List implementation with a sentinel node

IntListB

```
import java.io.*;
import java.util.*;

public class IntListB {
//Implements a singly linked list of ints with a sentinel node

  private class Node {
    private int data;
    private Node next;
    private Node(int d, Node n) {
      data = d;
      next = n;
    }
  }

  private Node head;
```

IntListB

```
public IntListB() {
        head = new Node(Integer.MIN_VALUE, null);
        //the empty list contains a sentinel node
}

public boolean empty() {
        return head.next == null;
}
public void insertFirst(int d) {
        head.next = new Node(d, head.next);
}
public void insertLast(int d)  {
       Completed in class
}
```

IntListB

```
public int minI() {
//PRE: !empty()
//returns the smallest int in the list
//the implementation must be iterative

       int min = head.next.data;
       Node temp = head.next.next;
       while (temp != null) {
           if (min >  temp.data)
               min = temp.data;
           temp = temp.next;
       }
       return min;
}
```

IntListB

```
public int minR() {
//PRE: !empty()
//returns the smallest int in the list
//the implementation must be recursive
      return minR(head.next);

}

 public int minR(Node h) {
      //PRE: h != null
      //returns the minimum int in the list beginning at h
      //the implementation must be recursive
      if (h.next == null) return h.data;
      int min = minR(h.next);
      return h.data < min ? h.data : min;
 }
```

IntListB

```
    public String toString() {
        if (head.next == null) return "[]";
        String s = "[" + head.next.data;
        Node temp = head.next.next;
        while (temp != null) {
            s = s + ',' + temp.data;
            temp = temp.next;
        }
        return s + "]";
    }
}
```

Exercise

- Implement a remove function