1. One possible solution appears below.
   public boolean hasTwoConsecutive() {
      if (front == null || front.next == null) {
         return false;
      }
      ListNode current = front;
      while (current.next != null) {
         if (current.data + 1 == current.next.data) {
            return true;
         }
         current = current.next;
      }
      return false;
   }

2. One possible solution appears below.
   public boolean isSorted() {
      if (front != null) {
         ListNode current = front;
         while (current.next != null) {
            if (current.data > current.next.data) {
               return false;
            }
            current = current.next;
         }
      }
      return true;
   }

3. One possible solution appears below.
   public void stutter() {
      ListNode current = front;
      while (current != null) {
         current.next = new ListNode(current.data, current.next);
         current = current.next.next;
      }
   }

4. One possible solution appears below.
   public void reverse3() {
      if (front != null && front.next != null &&
          front.next.next != null) {
         ListNode temp = front;
         front = front.next.next;
         ListNode temp2 = front.next;
         front.next = temp.next;
         temp.next.next = temp;
         temp.next = temp2;
         while (temp.next != null && temp.next.next != null &&
               temp.next.next.next != null) {
            temp2 = temp.next;
            temp.next = temp.next.next.next;
            temp = temp2;
            temp2 = temp.next.next.next;
            temp.next.next.next = temp.next;
            temp.next.next = temp;
            temp.next = temp2;
         }
      }
   }
5. One possible solution appears below.
   ```java
   public void removeAll(int value) {
       while (front != null && front.data == value) {
           front = front.next;
       }
       if (front != null) {
           ListNode current = front;
           while (current.next != null)
               if (current.next.data == value) {
                   current.next = current.next.next;
               } else {
                   current = current.next;
               }
       }
   }
   ```

6. One possible solution appears below.
   ```java
   public LinkedIntList removeEvens() {
       LinkedIntList result = new LinkedIntList();
       if (front != null) {
           result.front = front;
           front = front.next;
           ListNode current = front;
           ListNode resultLast = result.front;
           while (current != null && current.next != null) {
               resultLast.next = current.next;
               resultLast = current.next;
               current.next = current.next.next;
               current = current.next;
           }
           resultLast.next = null;
       }
       return result;
   }
   ```

7. One possible solution appears below.
   ```java
   public void doubleList() {
       if (front != null) {
           ListNode half2 = new ListNode(front.data);
           ListNode back = half2;
           ListNode current = front;
           while (current.next != null) {
               current = current.next;
               back.next = new ListNode(current.data);
               back = back.next;
           }
           current.next = half2;
       }
   }
   ```

8. One possible solution appears below.
   ```java
   public void reverse() {
       ListNode current = front;
       ListNode previous = null;
       while (current != null) {
           ListNode nextNode = current.next;
           current.next = previous;
           previous = current;
           current = nextNode;
       }
       front = previous;
   }
   ```