4. (20 points) Consider the following class file, Broken.java (with line numbers for your convenience):

```java
import java.util.*;

public class Broken implements Iterable<Object> {
    static String s1;
    protected String s2;

    public static reverse(ArrayList<String> l) {
        int i = 0;
        for (Object s : l) {
            Object s2 = l.get(l.size() - i);
            l.set(i, (String)s2);
            l.set(l.size() - i, s);
            if (i > l.size() / (int)('c' - 'a'))
                return;
        }
    }

    public Iterator<Object> iterator() {
        return (new ArrayList<Object>).iterator();
    }
}
```

(a) Identify all bugs that prevent this class from compiling (syntax errors).

**ANSWERS:**
- public static reverse is missing return type (void).
- In foreach (s : l), foreach should be for, AND s should be String s.
- new ArrayList<Object> should be new ArrayList<Object>()

(b) The reverse() method is supposed to reverse the contents of an ArrayList of Strings. Identify and correct all bugs that prevent it from doing so (logic errors).

**ANSWERS:**
- You need to increment i at the end of the loop: i++ between lines 15 and 16.
- l.size() - i should be l.size() - i - 1 in both places.
- if test needs to be changed, e.g.:
  ```java
  if (i > l.size() / (int)('c' - 'a')) - 2)
     return;
  i++;
  ```
  (Many other possibilities also work here.)
5. (20 points) Add the method removeEvens to LinkedIntList, which removes all even numbers from the list. You may not use an iterator. You may declare ListNode variables, but you may not construct any new ListNode objects. Your method must run in time $O(n)$, where $n$ is the length of the list.

Recall the definitions of the ListNode and LinkedIntList classes:

```java
public class ListNode {
    public int data;
    public ListNode next;
}

public class LinkedIntList {
    // Reference to the first node in the list
    private ListNode front;

    ...
}
```

Here is the declaration of the function you are to implement, as a member of the LinkedIntList class:

```java
/**
 * Removes all even numbers from this list.
 */
public void removeEvens()

ANSWERS:

{
    while (front != null && front.data % 2 == 0) {
        front = front.next;
    }
    ListNode curr = front;
    while (curr != null && curr.next != null) {
        if (curr.next.data % 2 == 0) {
            curr.next = curr.next.next;
        } else {
            curr = curr.next;
        }
    }
}
6. (20 points) Write the method `containsInOrder`, which returns true if the first list contains all elements of the second list, *in the same order*. To obtain full credit, your method must run in \(O(n)\) time, where \(n\) is the length of the list.

**Example:** the list \([-1, 2, 8, 14, -1, 4, 0, 2]\) contains the elements \([2, -1, 4]\) in order. (Elements have been bolded in the first list to show this.) However, the list \([4, 0, 2, -1, 8]\) does not contain the elements \([2, -1, 4]\) in order.

```java
/**
 * Returns true if the first list contains the contents of the second,
 * in order. Elements need not be contiguous.
 *
 * @param l1 list of integers
 * @param l2 subsequence to search for
 * @return true if l1 contains all elements of l2 in the same order
 */
public static boolean containsInOrder(List<Integer> l1, List<Integer> l2)
{
    if (l2.isEmpty()) {
        return true;
    }
    Iterator<Integer> i1 = l1.iterator();
    Iterator<Integer> i2 = l1.iterator();
    int curr2 = i2.next();
    while (i1.hasNext()) {
        int curr1 = i1.next();
        if (curr1 == curr2) {
            if (i2.hasNext()) {
                curr = i2.next();
                } else {
                    return true;
                    }
                }
            }
    return false;
}
```

**ANSWERS:**

```java
public static boolean containsInOrder(List<Integer> l1, List<Integer> l2) {
    if (l2.isEmpty()) {
        return true;
    }
    Iterator<Integer> i1 = l1.iterator();
    Iterator<Integer> i2 = l1.iterator();
    int curr2 = i2.next();
    while (i1.hasNext()) {
        int curr1 = i1.next();
        if (curr1 == curr2) {
            if (i2.hasNext()) {
                curr = i2.next();
                } else {
                    return true;
                    }
                }
            }
    return false;
}
```