Part II (3 points)  
-------------------
Fill in the `squish()` method in the `SList` class so that it performs as indicated in the comment. Your solution should not use arrays, nor should it use your `smoosh()` method. Do not change the prototype of the `SList` constructor or the `insertEnd` method; our test software will call them.

```java
/**
 * squish() takes this list and, wherever two or more consecutive items are
 * equals(), it removes duplicate nodes so that only one consecutive copy
 * remains. Hence, no two consecutive items in this list are equals() upon
 * completion of the procedure.
 * 
 * After squish() executes, the list may well be shorter than when squish()
 * began. No extra items are added to make up for those removed.
 * 
 * For example, if the input list is [0 0 0 0 1 1 0 0 0 3 3 1 1 0], the
 * output list is [0 1 0 3 1 0].
 * 
 * IMPORTANT: Be sure you use the equals() method, and not the "=="
 * operator, to compare items.
 **/

public void squish() {
    // Fill in your solution here. (Ours is eleven lines long.)
}
```

Part III (2 points)  
-------------------
Fill in the `twin()` method in the `SList` class so that it performs as indicated in the comment. Your solution should not use arrays.

```java
/**
 * twin() takes this list and doubles its length by replacing each node
 * with two consecutive nodes referencing the same item.
 * 
 * For example, if the input list is [3 7 4 2 2], the
 * output list is [3 3 7 7 4 4 2 2 2 2].
 * 
 * IMPORTANT: Do not try to make new copies of the items themselves.
 * Just copy the references to the items.
 **/

public void twin() {
    // Fill in your solution here. (Ours is seven lines long.)
}
```

** Submitting your solution **

Change (cd) to your `hw3` directory, which should contain `Homework3.java`, `SList.java`, `SListNode.java`, `TestHelper.java`, and any other files needed to run your methods. Make sure your homework compiles and runs on the _lab_ machines just before you submit.

From your `hw3` directory, type *submit hw3*. After submitting, if you realize your solution is flawed, you may fix it and submit again. You may submit as often as you like. Only the last version you submit before the deadline will be graded.