CS 61B Homework 3
Due 3pm Wednesday, February 9, 2000

This homework assignment is designed to give you practice working with arrays and linked lists. This is an individual assignment; you may not share code with other students.

Copy the Homework 3 directory by doing the following, starting from your home directory:

```
mkdir hw3
cd hw3
cp $master/hw/hw3/* .
```

Your task is to write two methods for removing successive duplicate items from lists, and one method for adding them. The smoosh() method operates on lists represented as arrays, and the squish() method and twin() method operate on singly-linked lists.

The Homework3 class includes test code for all three methods, as well as a skeleton for the smoosh() method. The SList class from Lab 3 is also present, and here includes skeletons for the squish() method and the twin() method. You can test all three methods by compiling and running Homework3.java.

Part I (5 points)

Fill in the smoosh() method in the Homework3 class so that it performs as indicated in the comment. Your solution should not use linked lists, nor should it use your squish() method.

```
  /**
   *  smoosh() takes an array of integers and returns an array containing the
   *  same numbers, but wherever the input array had two or more consecutive
   *  duplicate numbers, they are replaced in the output array by one copy of
   *  the number.  Hence, no two consecutive numbers in the output array are
   *  the same.
   *
   *  The output array is the same length as the input array.  Any unused
   *  elements at the end are set to -1.
   *
   *  For example, if the input array is [ 0 0 0 0 1 1 0 0 0 3 3 3 1 1 0 ], the
   *  output array is [ 0 1 0 3 1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 ].
   *
   *  @param ints the input array.
   *  @return the compressed array with no two consecutive duplicate numbers.
   **/

  private static int[] smoosh(int[] ints) {
    // Fill in your solution here.  (Ours is sixteen lines long.)
  }
```

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.

```
  /**
   *  squish() takes this list and, wherever two or more consecutive items are
   *  equal(), it removes duplicate nodes so that only one consecutive copy
   *  remains.  Hence, no two consecutive items in this list are equal() 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 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 ten 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.

```
  /**
   *  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 six 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. If you have any comments you would like to make to the grader, such as parts of your solution that you know are not correct, create a file called GRADER and put your comments there. In either GRADER or a comment in Homework3.java, include your name, login, and lab section. 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.