

## readme

CS 61B Homework 3  
Due noon Wednesday, February 12, 2014

This homework assignment is designed to give you practice working with arrays, linked lists, and nested loops. It will also give you practice for the similar but harder run-length encoding computations in Project 1. 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:

```
cp -r ~cs61b/hw/hw3 .
cd 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`. As usual, you are welcome to add test cases to the `main()` method or change `main()` as you please; we will not test `main()`. However, you cannot change the interface of the public methods and classes, because our autograder will use them. So you might want to keep a copy of the original `main()` test code around--if you accidentally change a prototype, the test code will catch it.

You may NOT use Java's built-in data structure libraries, like `java.util.Vector` or `java.util.LinkedList`, in this homework (or any future homework, except where otherwise specified). All data structure implementations should be your own or those taken from lectures/labs/homeworks.

#### 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 ints. On completion the array contains
 * the same numbers, but wherever the array had two or more consecutive
 * duplicate numbers, they are replaced by one copy of the number. Hence,
 * after smoosh() is done, no two consecutive numbers in the array are the
 * same.
 *
 * Any unused elements at the end of the array 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 ],
 * it reads [ 0 1 0 3 1 0 -1 -1 -1 -1 -1 -1 -1 -1 ] after smoosh()
 * completes.
 *
 * @param ints the input array.
 */
public static void smoosh(int[] ints) {
    // Fill in your solution here. (Ours is fourteen lines long, not counting
    // blank lines or lines already present in this file.)
}
```

#### 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
 * 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 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.

```
/**
 * 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.
 * Make new SListNodes, but 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.