CS61B, Fall 2008 **HW #2** **P. N. Hilfinger**

Due: Wed., 17 September 2008 (at midnight)

We have skeleton files for your solutions. Assuming you have set up `svnwork` and `staff` directories as indicated in the first lab, you can use this procedure on the Unix machines:

```
# Fetch any new stuff into your working copy of staff
% svn update ~/staff
% cd ~/svnwork
% svn copy ~/staff/hw2 hw2
% svn commit -m "Get initial HW2 skeleton"
```

Now you have a `trunk/hw2` directory in the repository, with a copy in your Subversion working directory (`~/svnwork`). (These files are also in `~cs61b/code/hw2`.) Modify the files there, commit them (frequently as you are working), and then hand them in with

```
% cd ~/svnwork
% svn commit -m "Make sure HW1 work is committed"
% svn update
% svn copy hw1 svn+ssh://cs61b-ta@quasar.cs.berkeley.edu/cs61b-yu/tags/hw1-1 \
>     -m 'Submit HW1'
```

Place your answers to all these problems in the files named `Arrays.java` and `Lists.java`. Don't allow the behavior of the functions in `Arrays.java` or `Lists.java` depend on anything in `Utils.java`, or on anything you might add to `IntList.java` or `IntList2.java`, since we will replace all these files with the originals before testing. You can add any other files you want, however (be sure to commit them, of course!).

1. Complete the following Java function so that it performs as indicated in its comment. The files `IntList.java` and `IntList2.java` in the template contain the declarations of the classes `IntList` and `IntList2`. Put your answers in a file called `Lists.java`, for which we've also provided a template. You may find the functions in the file `Utils.java` to be useful for testing.

```
/** The list of lists formed by breaking up L into "natural runs":
 *  that is, maximal ascending sublists, in the same order as
 *  the original.  For example, if L is (1, 3, 7, 5, 4, 6, 9, 10),
 *  then result is the three-item list ((1, 3, 7), (5), (4, 6, 9, 10)).
 *  Destructive: creates no new IntList items, and may modify the
 *  original list pointed to by L. */
static IntList2 naturalRuns (IntList L) {
    /* *Fill in here* */
}
```

2. Complete the following Java functions so that they perform as indicated in their comments. Remember that some arrays can have zero elements. Put your answers to this problem (all parts) in a file named `Arrays.java`, for which we've provided a template. You may find the contents of the file `Utils.java` useful in testing your answers.

a.
```
/** A new array consisting of the elements of A followed by the
 *  the elements of B. */
static int[] catenate(int[] A, int[] B) {
    /* *Fill in here* */
}
```

b.
```
/** The array formed by removing LEN items from A,
 *  beginning with item #START (counts from 0). */
static int[] remove (int[] A, int start, int len) {
    /* *Fill in here* */
}
```

Continued...

c.
```
/** The array of arrays formed by breaking up A into
 *  maximal ascending lists, without reordering.
 *  For example, if A is {1, 3, 7, 5, 4, 6, 9, 10}, then
 *  returns the three-element array
 *  {{1, 3, 7}, {5}, {4, 6, 9, 10}}. */
static int[][] naturalRuns (int[] A) {
    /* *Fill in here* */
}
```