1 Homework Assignment 1

1.1 The Assignment

Your assignment for Homework 1 is to write a simple program to read baskets into a list of baskets, sort the items in each basket, and then sort the baskets.

The driver program is given in the code on the website.

You may use this code as is EXCEPT that you must add a line with your name on it. That is, you could change the top documentation from

```
/*********************************************************************
 * Homework 01.
 * This assignment is to read baskets into a list, to sort the
 * baskets, and then to count the number of unique baskets.
 *
 * Copyright (C) 2012 by Duncan A. Buell.  All rights reserved.
 *
 * @author Duncan A. Buell
 * @version 1.00 2012-07-03
**/
```

to read

```
/*********************************************************************
 * Homework 01.
 * This assignment is to read baskets into a list, to sort the
 * baskets, and then to count the number of unique baskets.
 *
 * Copyright (C) 2012 by Duncan A. Buell.  All rights reserved.
 * Used with permission and modified by J. Random Student.
 *
 * @author Duncan A. Buell
 * @version 1.00 2012-07-03
**/
```

The "modified by" is an easy way to provide attribution to the original author, claim some mods of your own, but leave to a later discussion the question of how much modification took place.

In the zip file you have a driver program you can use pretty much as is, and you have a `FileUtils` program you can use entirely as given to you (For the most part, you can use my `FileUtils` program throughout this course.)

There is also a file for input called `zin` and a sample output file `zout`. The `zout` file is the result of running my code on this problem.

YOU DO NOT HAVE TO DUPLICATE EXACTLY MY OUTPUT IN THIS COURSE, BUT YOU DO HAVE TO DUPLICATE THE CONTENT OF THE OUTPUT AND YOU COULD BE GRADED OFF IF YOU DECIDE TO CHANGE THE CONTENT TO SOMETHING THAT I DON'T THINK IS CORRECT.

You have an interface for the "application" class that is to be called `MyListStructure`. You must implement the methods in the interface, and your `MyListStructure` class must state that it implements the interface.

The data in the input file can be read conveniently by looking more carefully at the `Scanner` class. The problem is that you are reading in data, that each line is a different basket, but the basket sizes vary. One way to get the data into the right places, assuming you are reading from a `Scanner` called `inFile`, is to use code like the following.

```
Scanner localScanner = null;
if(inFile.hasNext())
{
  localScanner = new Scanner(inFile.nextLine());
  while(localScanner.hasNext())
  {
    int n = localScanner.nextInt();
    list.add(n);
  }
}
```

This reads one line of input using the `nextLine` method. It then creates a new, local `Scanner` with exactly that one string as its input, and now that one string can be parsed with `nextInt` and `hasNext`.

The data comes in with each basket unsorted and the baskets themselves unsorted. You will need to sort each basket, to then sort the baskets, and then remove the duplicates.

CAVEAT: When you go to remove duplicates, read the list from the end to the beginning. If you process forward, but remove an entry, the length of the list gets shorter and you run the risk of having the loop subscripts out of sync. If you process from the end to the beginning, you are always removing at the end, but you never look at the end part of the list from which you are removing.