# CS 051G Homework Lablet# 8
## LIst Practice

**Objective:** To gain experience using 1 dimensional lists.

—

**The Scenario.** Because of the Cesar Chavez day holiday, there will be no quiz or lab this Friday. However, to get you some practice with arrays, we are asking you to do this "lablet", consisting of writing five simple methods.

For this program, you are allowed to work in pairs with another student in the class. If you do this, please turn in a single copy of your program, with both students' names on the class. You may work with any student in the class, even if you have worked with them in the past.

**Question 1: Average** Write a method that takes an array of doubles and computes the average of all of them, i.e., the sum of the numbers divided by the number of entries in the array.

```
method average (aList: ListNumber)  Number
```

**Question 2: The Same Elements** Write a method that takes two arrays of ints and determines if they represent the same lists, i.e., that they have the same length and have equal numbers in corresponding positions. That is for every slot i in the lists, the values in that slot are the same for the arrays.

```
method theSame(fstList: ListNumber, sndList: ListNumber)  Boolean
```

If `firstList` $= [1, 3, 5, 7]$, `secondList` $= [1, 3, 5, 7, 9]$, `thirdList` $= [1, 3, 7, 5]$ then `theSame(firstList, secondList)` will return false because they have different lengths and `theSame(firstList, thirdList)` will also return false, this time because the elements in slot 3 in the arrays are different.

**Question 3: Truncate at Flag** Write a method that takes a list of doubles as well as a double called flag, and returns a new list composed of all the elements of the list up to, but not including the first occurrence of flag. If flag does not occur in the list, just return a new copy of the list.

```
method truncate (aList: ListNumber) at (flag: Number)  ListNumber
```

If `myList` is composed of $[1, 3, 5, 8, -2, -999, 15, 6]$ then `truncateAtFlag(myList,-999)` will return the new array $[1, 3, 5, 8, -2]$, while `truncateAtFlag(myList1,-99)` will return the array $[1, 3, 5, 8, -2, -999, 15, 6]$.

**Question 4: Filter Evens and Square** Write a method that takes an array of ints and returns a new array composed of the squares of the even ints held in the original array.

```
method filterEvensNSquare (aList: ListNumber)  List Number
```

If `myList` is composed of $[1, 3, 5, 8, -2, -999, 15, 6]$ then `filterEvens(myList)` will return the new array $[64, 4, 36]$.

**Question 5: RainFall** Write a method called rainfall that consumes a list of numbers representing daily rainfall amounts as entered by a user. The list may contain the number -999 indicating the end of the data of interest. Produce the average of the non-negative values in the list up to the first -999 (if it shows up). There may be negative numbers other than -999 in the list.

```
method rainfall(aList: ListNumber)  Number
```

If `myList` is composed of $[1, 3, 5, 8, -2, -999, 15, 6]$ then `rainfall(myList)` will return 3 (= 15/5).

## 0.1 How to Proceed

We will provide you with a starter file at
`http://www.cs.pomona.edu/classes/cs051G/labs/listplay/ListPlayStart.grace`
that contains the headers of all of the methods you are to write. It also contains code to test your methods that will be executed when your program starts. It prints out the answers from your methods along with what the correct answer should be. For example, the first line printed will initially result in

```
Average: 0.0 should be 4
```

This tell you that the version of the `average` method I provided you with returns the value 0, but it should have returned 4. Obviously you should fix the method so that it will compute the correct average of 4.

Please do not change the names or parameters of any of the methods as we are likely to test them on our own data and our calls will depend on the names all being consistent with those provided to you.

## 0.2 Submitting Your Work

When your work is complete you should deposit in the appropriate dropoff folder a single folder containing the file with your program. Make sure the folder name is of the form `lablet8_lastnamefirstname`. Also make sure that your name is included in the comment at the top of the Grace source file.

Before turning in your work, be sure to double check both its logical organization and your style of presentation. Make your code as clear as possible and include appropriate comments describing major sections of code and declarations.

This lab is due *Monday night* at 11 p.m. as usual. The grading guidelines are simple: 2 points for each of the 5 parts.