This assignment gives you some practice with structures. A tar file has been provided in hw05.tar.gz. Download and un-tar this file with the command "tar -xvzf hw05.tar.gz" to create the hw05 sub-directory that starts with a Makefile, as well as some .c and .h files, and some test input data files.

For this assignment, you need to implement an arrayList structure very similar to the Java "ArrayList" class, as documented in Java Docs ArrayList. The main difference between Java ArrayLists and this homework is that in Java, you can create an ArrayList of any generic type <T>. For this assignment, we will restrict ourselves to arrayLists of integers. Secondly, the Java class has object oriented methods. For this assignment, we will use simple C functions that take an explicit argument of type "arrayList". I have defined the "arrayList" type as a pointer to a structure in arrayList.h. The arrayList.h file also contains function declarations for all the required "method" style functions that deal with arrayLists. You should not modify the arrayList.h file. You will need to implement the arrayList methods by writing the function definitions for all the "method" style functions in file arrayLists.c. The java documentation for these methods also applies to the C functions I am asking you to write. For instance, the arrayListIndexOf function should return a -1 if it cannot find it's argument in the arrayList, just like the Java "indexOf" method. It is acceptable to include invocations of "assert" in the C arrayList functions instead of throwing exceptions. The "arrayListToString" method should behave like the Java toString method on ArrayLists… namely return a string that starts with a left square bracket, contains the ASCII representation of each element in the arrayList separated by commas, followed by a right square bracket. The result of the arrayListToString method should be written into the space provided by your caller in the "buffer" argument, and you should return the buffer pointer when you are finished.

Note that because we have not yet talked a lot about dynamic memory, I have provided the arrayListCreate function to create a new arrayList, an arrayListEnlarge function that doubles the number of integers that an arrayList can hold, and an arrayListFree function to give back the dynamic memory allocated.

I have also provided a "tryList.c" function which uses arrayLists. The tryList.c main function reads a list of numbers from stdin into an arrayList, prints out the arrayList along with some info, and then creates a couple of new arrayLists with running averages derived from the input lists. You should not have to modify this code.

When the arrayList.c functions are defined correctly, the output should look like the following:

```
~/cs220/hw05>make test
gcc -Wall -g -o tryList tryList.c arrayList.c
./tryList <test1.txt
Read input data: [ 10, 20, 40, 10, 30, 60, 40, 50]
The number 40 first appears in the input at index 2
Running averages (3s): [ 23, 23, 26, 33, 43, 50]
Running averages (4s): [ 57, 62, 72, 72, 82]
Running averages (4s) does contain the number 72
```

When you are done testing, run "make submit" to create a file called "hw05_<*userid*>.tar.gz", where <*userid*> is your gmail userid. Upload this file on myCourses under Content, Homework Submissions, Homework 05 Submission. This assignment is due at 11:59 PM on Sunday, February 26, 2017. You may submit as many times as you wish; but only the latest submission will be graded.

This assignment is worth 10 points. Your grade will be calculated as follows:

- If you submit late without an extension, there will be a two-point deduction for every 24 hours you are late. Extensions are available only in special circumstances, and can only be given by the instructor or a TA.
- There will be a three-point deduction for submissions that do not follow the required format.
- There will be a six-point deduction if there are compiler errors when compiling your code.
- There will be a three-point deduction if your code does not run to completion on any of the test cases used to grade your code.
- There will be a two-point deduction for each class of compiler warning message issued when compiling your code.
- There will be a two-point deduction if any of the arrayLists created are not freed.
- There will be a three-point deduction if tryList does not print the correct results for the test cases in the Makefile test, and a three-point deduction for an unpublished test case on tryList.