CSCI 221: Computer Programming II **Due Saturday March 18, 11:59pm**

Programming Assignment 4 *Leniency* 24 hours later

Collaboration Policy: Work alone. Write your own code. You may not use any code found on the internet or anywhere else in this assignment. You may not use any code provided by anyone other than the instructor, including tutors in CSL.

Task: Create an object that will allow one to store and manipulate an array of strings, without having to write many of the low-level array-manipulation routines. A `StringsList` object stores 0 or more String objects, with empty locations, if any at the end of the list. Those routines include:

public Constructor	**Description and Preconditions (if any)**
`StringsList ()`	Instantiates an empty `StringsList` with maximum capacity of 10, that will store strings (String values). Precondition: None.
public Methods	**Description and Preconditions (if any)**
`int size( )`	Returns the number of elements currently stored in the list. (NOTE: this is not the same as the capacity.) Precondition: none
`void add ( String a )`	Inserts `String` **a** at the end of the list. Postcondition: list has **a** appended, size is updated accordingly.
`void insert( String a, int location )`	Inserts `String` **a** in the specific user-numbered **location** in the list. (The first element is at location 1 to a user, even though it is stored at index 0 for the programmer.) If there is already an item at position **location**, that element and all elements following it are pushed one position over (to the next highest index) in the list. If the list is already full, at capacity, then the array size is doubled (see enlarge method below). Precondition: location is in the range of **1** to **size+1** Postcondition: contents modified as necessary and size is updated accordingly
`boolean contains (String a)`	Returns true if String a is in the list, false otherwise.
`int find (String a )`	Returns the index (a value from 1 to size) of **a** in the list, or -1 if a is not in the list.
`OPTIONAL BONUS:` `void delete ( String a )`	Removes the String a, **if it is in the list**. Any elements that follow a in the list are moved over one position (to the next lowest index) so there are no holes in the list. If a is not in the list, the list is not changed. Postcondition:contents modified as necessary and size is updated accordingly
`void displayList(  )`	Prints the contents of the `StringsList` for every location within the <u>capacity</u> of the list, showing the location of each element. For those locations where no String is stored, print null as the value. (See example below)
private Methods	**Description and Preconditions (if any)**
void enlarge( )	Doubles the capacity of the list, whenever needed. Called by **insert** method and add when a user inserts an element into a full list. This is done by creating a new

	list double the size and copying the elements from the original into this list Precondition: none Postcondition: instance variables modified as necessary

Notes:

1. You many not use any existing methods that you don't write yourself to solve this problem.
2. The class and all methods must be named and declared exactly as specified.
3. The instance data for the class must include a string array (String [ ]) to store the list, and at most two other ints ((1) to store capacity, the size of array allocation and (2) another to store size, the actual number of elements stored). No other instance variables are allowed.
4. There is a distinction between the **capacity** and **size** of a list. **capacity** is the maximum number of elements that can be stored in the list. **size** is the actual number of elements stored in the list. (Watch for the use of each term in the descriptions above.)
5. Users will refer to locations starting with 1, they don't know, and should not care, that values are stored in an array with indexing starting at zero. So, if a user asks to delete the value at location 1, **delete** should remove the first item in the list (the one stored at index zero).
6. StringList maintains elements clustered near the front of the list – that is, a list will have no holes in it. If the capacity of a list is 20, but the list only contains 11 elements, those 11 elements are in locations 1 through 11 for the user (but stored at array indices 0 through 10). Note the precondition for the insert(String, int) method.
7. A call to delete might ask to delete an element at an unused location, this call can be ignored.
8. **Include the signatures of all methods, even if you don't implement them. For all unimplemented methods, print "Not implemented. If a value must be returned, return 0 for int methods and true for boolean methods.**
9. displayList( ) should display both the user-understood index of each data item, and its contents. Sample output is:

```
StringList: capcity 5, size 3
[1] hello          a
[2] wow
[3] how
[4] null
[5] null
```

10. Hint, first develop constructor so you can make a StringsList, then the displayList method second so you can see what was created, then build the other methods in the order shown, using displayList as a way to test that the methods work. Include signatures for all.

Program Documentation
Use the simple standard demonstrated in HW 2 for each method – most of the comments are actually the descriptions provided above. All files should have an opening description and your name.

Program Submission
Create a ZIP file that only contains a folder named <yournameHW4> that contains only the completed `StringList.java` class file. When you zip up your folder, its name should be: *<lastname>*HW4.zip e.g., McCauleyHW4.zip

If the assignment is not submitted in the correct format – it will not be accepted. Submit the ZIP file via OAKS in the Dropbox that corresponds to the assignment. Resubmit, as many times as you like, the newest submission will be the graded submission.