# HW3: Writing a Linked List

DUE DATE: Friday June 13, 2008

A linked list is a dynamic data structure that stores information in nodes. Each node is a data structure itself that stores information and the next node that it references. Unlike an array, a linked list can be adjusted without moving any data over to a new linked list or without shifting over any information. Therefore, it can be beneficial in many ways depending on the task at hand.
As a linked list, you only know about the first node. That node has its data element and then points you to the next node. To find the last element in the list you need to start at the first node and traverse the entire list node by node until you run out of elements (if the next of a node points to null, then you are at the end of the list).

For Homework 3 you will be creating a linked list containing Strings. You will be turning in 2 files: `StringList.java` and `StringNode.java`

## StringNode.java

This is the node that will make up your linked list. It needs to have a data element (of type `String`) and it will need to have a data element that will be the next node in the list (of type `StringNode`). These elements also have to be private and have the appropriate accessors and modifiers. You will be required to have a constructor, `public StringNode(String data)`, that creates a stand alone node with `data` as its data element. You will also be required to have a `toString()` that summarizes the information in your data element (the string in the node).

**Note**: Although only one constructor is required, you may find it useful to have more (like an empty one and one that takes in data and the next node)

## StringList.java

This is the actual linked list. It will not have access to every element, just the first one (typically called head). Once again your head data element will be private and will need to include the appropriate accessors and modifiers. This is where you will insert and remove elements. Be sure to include a constructor that initializes the list by setting head to null. `StringList` will need to have the following methods:

```
public StringList(){…}
public StringList(StringNode head){…}
public StringNode getHead(){…}
public void setHead(StringNode head){…}
public void addToFront(StringNode contactToInsert){…}
public void addToTail(StringNode contactToInsert){…}
public boolean insertAfter(StringNode node, StringNode
stringToInsert){…}
public boolean delete(StringNode node){…}
public int size(){…}
public String toString(){…}
```

**Note on toString():** The `toString()` method should print out a `String` representation of the node's data (the String within the node)

## Bonus

10pts -Allow your users to sort your list. Add method `sortList()` to your `StringList`.
5pts -Allow your users to add at any point in the list. Add method `addAt(StringNode sn, int i)` to your `StringList`.
10pts -Allow your users to delete from the beginning and end of the list. Add `deleteFromFront()` and `deleteFromBack()` methods.

**WARNING:** You are NOT allowed to use the built-in `LinkedList` in java. You should not have to import anything for this assignment.

## Random Notes:

-Make sure your homework compiles before turning it in.
-Make sure to turn in the .java file
-COMMENT YOUR CODE! If you don't comment, we assume you copied from someone and have no idea what you're doing. As usual, lack of commenting will cost you a substantial amount of points.

## What to Turn In

- StringNode.java
- StringList.java

## How to Turn In

- Turn in via TSquare