# Assignment 2

This assignment is designed to introduce you to Linked Lists, and further your understanding of sorting algorithms.

# Description

You will implement List, Stack and Queue interfaces, specifically for individual character data elements. You will be provided a partially working doubly-linked List implementation to get you started.

# Overview

The project zip file contains a variety of files (in addition to the assignment document):

1. CharListable.java
2. CharStackable.java
3. CharQueueable.java
4. CharList.java
5. Test\*.java

## Files Overview and requirements

Details on the five files.

### CharListable.java

CharListable.java is an interface that declares the List operations required by this assignment.

### CharStackable.java

CharStackable.java is an interface that declares the required Stack operations.

### CharQueueable.java

CharQueueable.java is an interface that declares the required Queue operations.

### CharList.java

CharList.java is a partially implemented character List. Several of the methods declared in the related interface (CharListable.java) are “stubbed” only to the point that they will satisfy the compiler, but are not truly concrete. Your job will be to fully define these methods so that the class truly fulfills the expectations of the interface. You can also add extra methods if you deem them necessary. For example, the partially implemented class includes a toString() method to facilitate testing (not to mention it’s simply good practice). NOTE: CharList.java is doubly-linked, and should remain that way.

### Test\*.java

These are a variety of test programs that you can use to test your work.

# Requirements

1. Complete CharList.java so that it fully implements all the methods declared in CharListable.java. For the sort() method, you can choose to use Merge Sort or Quick Sort.
2. Create and write a class name CharStack.java that fully implements the CharStackable.java interface.
3. Create and write a class name CharQueue.java that fully implements the CharQueueable.java interface.

Note: “fully implement” means that all methods declared in each interface are concrete and working properly.

# Hints

You should implement CharList.java first, because then implementing CharStack.java and CharQueue.java should be relatively simple. For example, assuming that you’ve composed a CharList object named “cl” in your Stack (or Queue), then you can implement toString() as simply as the following code snippet:

public String toString() {

return cl.toString();

}

In other words, use the List’s operations (methods) to accomplish the other ADT’s operations. To put it in context, my “solution” implementation of CharStack.java is only 24 lines of code (and only 10 of those are “new” lines of code, because the other 14 are defined by the method declarations in the interface), and my CharList.java is only 105 lines (including 64 from the partially completed version here).

# Extra Credit Opportunities

## 3 points each for any or all of the following:

1. *Re*-implement CharStackable.java with a singly linked list (name file SLCharStack.java), in addition to CharStack.java.
2. *Re*-implement CharQueueable.java with a singly linked list (name file SLCharQueue.java), in addition to CharQueue.java.
3. Implement both Merge Sort and Quick Sort in CharList.java (instead of only one).

# Deliverables

You should submit a zip file containing only your java source code, including the original interfaces (for easy compiling).