Week 2 Workshop: Review, Recursion, and Iteration

- **Resources:** Week 1 Diagnostic Quiz and Code Bundle.
- **To submit this week's work:** Submit your solution to Exercise 3 to your teacher in class. You should hand in a single piece of paper with your solution on it. Be sure you include your name, student number, and the week number in the top-right of your submission.

Questions that ask about linked lists are asking about the *concept* of linked lists we spoke about in the lecture and which we *implemented* in our `LinkedList` class. There are infinitely many different linked list implementations, please use ours.

Exercises

1 Java Review: Definitions

As precisely as you can, define the following terms (which you learned in COMP1010):

1. method
2. class
3. reference
4. value

2 Java Review: Java Program

Write a Java program which reads data from the user on the console and stores each response they give. When the user gives the special response "over and out", the program will replay their responses.

An example interaction would be

```
> ? hi
> ? my
> ? knee is
> ? sore
> ? over and out
hi my knee is sore
```

You may assume there will be no more than 10 lines input before "over and out" is input.

3 Recursive Data: Concepts - Submission

Draw diagrams of the three linked lists that result from the following code being executed.

```
LinkedList small = new LinkedList('m');

LinkedList matt =
       new LinkedList('m',
         new LinkedList('a',
           new LinkedList('t',
             new LinkedList('t'))));

LinkedList notMatt = new LinkedList('s');
notMatt.add('o');
notMatt.add('m');
notMatt.add('e');
notMatt.add('o');
notMatt.add('n');
notMatt.add('e');
notMatt.add('e');
notMatt.add('l');
notMatt.add('s');
notMatt.add('e');
```

4 Recursive Data: More

We've seen that linked lists are a type of recursive data. In this task we will try to come up with some other type of recursive data and think about how it might be structured. Recall

> Recursion refers to the situation where the smaller part shares key characteristics of the whole.

We chose to use linked lists of characters because *Strings* are another type of recursive data. A string missing one of its characters is also a string! Thinking along those lines, think of another kind of data that could be treated recursively.

Note: This will require either imagination or some research since you haven't seen that many other data types so far.

5 Recursion: Write a recursive function

Write a *concatenation* function for the `LinkedList` class. The method should concatenate the given list to `this` list, updating `this` in-place. The method signature should be:

```
void concat(LinkedList other);
```

and it should pass *at least* the following tests (assuming the definitions of `small`, `matt`, and `notMatt` from above)

```
matt.concat(notMatt);
matt.concat(small);

assertTrue(matt.toString().equals("mattsomeoneelsem"));
```

6 Recursion Iteration Iso: Iterative Version

Write an iterative version of the `concat` function you wrote above.