INTRODUCTION TO COMPUTER SCIENCE II
LAB 6
Simple lists

1 The `SimpleList` interface

Some code that you will get from me (see Section 2) contains an *Interface* for a `SimpleList`. It is a highly simplified version of what the Java `List` interface demands. Specifically, it requires only the following capabilities of any container object that wants to qualify as a `SimpleList`:

- *insert* a value at an index
- *remove* an index from the list
- *get* at an index
- *set* a value at an index
- *find* the index of a value
- *length*[1]

We will create and work with a couple of container classes that *implement this interface*.

2 Getting started

Begin by obtaining the source code that is the starting point for this lab, either by clicking this link or by logging into `remus/romulus` and using this command:

```
$ cp ~sfkaplan/public/COSC-112/lab-6/* .
```

You should end up with a collection of five source code files:

- `SimpleList.java`: Defines the `SimpleList` interface.
- `SimpleArrayList.java`: A complete container class that implements the `SimpleList` interface and internally uses an array to store the values.
- `SimpleLinkedList.java`: An incomplete container class that implements the `SimpleList` interface and internally uses a linked list to store the values.
- `SimpleLink.java`: A (very simple) class whose objects are used by a `SimpleLinkedList` to construct its chain.
- `TestList.java`: A static class that is used to create a `SimpleList` and use its methods to test and debug a `SimpleList` container class (e.g., `SimpleArrayList`, `SimpleLinkedList`).

[1] Unfortunately, this one is a noun and not a verb, so it looks a little awkward here, but I think you know what it does.

3 Your assignment

There are a few steps to be performed with this source code:

1. **Comment the interface**: Examine both `SimpleList.java` (the interface itself) and `SimpleArrayList.java` (a container class that implements the interface). See how each method behaves. For example, if there is a call to *insert* a value at a position that is beyond the current list's length, what should happen? See what the actual container class does. Then, **comment** `SimpleList.java` to explain how each method in the interface should behave to properly conform to the interface's expectations.

2. **Complete the linked list container:** Open `SimpleLinkedList.java` and you will find three methods, marked by the comment, `// COMPLETE ME`, that are not complete. Fill in those methods to behave just as you described in the interface's comments.

3. **Test your linked list container:** Modify and use the tester class, `TestList.java`, to debug your work in `SimpleLinkedList.java`.

4 How to submit your work

Use the CS submission systems to submit your `SimpleList.java` and `SimpleLinkedList.java` source code files:

- **Web-based:** Visit the submission system web page.
- **Command-line based:** Use the `~lamcgeoch/submit` command at your shell prompt.

This assignment is due on Sunday, Dec-04, 11:59 pm.