Introduction to Computer Science II
Lab 5
A Nice Linked List

1 The `NiceList` interface

We start with the `NiceList` interface[1], which defines a *linear, indexed container* that can store any collection of `Objects`. Like any interface, it doesn't define **how** those values are stored; instead, it defines how the container must behave by specifying the methods that it must implement:

- `public void insert (int index, Object value)`
  Insert a `value` at the given `index`.
- `public Object remove (int index)`
  Remove and return the value at the given `index`.
- `public void set (int index, Object value)`
  Replace the value at the given `index` with the new `value`.
- `public Object get (int index)`
  Return the value found at the given `index`.
- `public int length ()`
  Return the current length of the list.

Any class that implements these methods with this behavior is a `NiceList`.

2 `NiceArrayList` and `NiceLinkedList`

For this assignment, there are two classes that implement the `NiceList` interface: `NiceArrayList` and `NiceLinkedList`.

[1] We use the silly prefix `Nice` because the interface `List` is already defined in Java. The `NiceList` interface is our simplified variant of the standard `List` interface.

The `NiceArrayList` class uses basic arrays to store pointer to `Object` values. The array is often larger than needed, and the `_length` data member keeps track of how many values are actually stored at any given moment. When the array gets full and another value is inserted, then a new array, double the size of the old one, is created, and the pointers are copied from the old to the new. This class is an example of one way to implement a `NiceList`. It is completely written, and you can look through it to see how it works and behaves.

The `NiceLinkedList` is only partially implemented. It uses a linked list (where the links are `NiceLink` objects) to store its pointers to `Object` values. The methods `set()` and `get()` are already implemented, as is the private helper method `walkTo()`. You can look at these to see how your code can move through the chained sequence of `NiceLink`s.

3 What you must do

Get the code: Start a *Terminal*. Then, grab some starting source code and open it:

```
$ cd
$ curl -L https://bit.ly/cosc-112-24f-lab-5 -o lab-5.zip
$ unzip lab-5.zip
$ cd lab-5
$ code .
```

Complete the linked list implementation: You must complete write the `insert()` and `remove()` methods of the `NiceLinkedList` class.

A few critical things to notice about the linked list class that you must complete:

- There is a `_length` data member. Notice that some methods depend on this variable being up-to-date. The methods you are writing already contain code to update this data member; be mindful and when and how this value changes when you write and debug your code.

- The methods you must write must implement a specific behavior. Check the interface itself (in `NiceList.java`) and examine the comments to see what the behavior of each should be.

- There is no class with a `main()` method. Write one, and use it to test your own code.

4 How to submit your work

Submit your `NiceLinkedList.java` file by uploading it into the `lab-5` folder in your shared Google Drive folder for this course.

This assignment is due on Sunday, Nov-03, 11:59 pm.