INTRODUCTION TO COMPUTER SCIENCE II
LAB 5
A Singly Linked List

1 A nice singly-linked list

We have, in class, worked out in some detail how to construct a linked list container that implements a linear-container interface (`NiceList<E>`) that can store any object type (i.e., is *generic*). It used *sentinels* as beginning and ending markers of the list, thus avoiding the need for special cases when inserting and deleting items.

That work was on a *doubly-linked list*—that is, a list whose links contain both *next* and *previous* pointers. Linked lists, however, may also be *singly linked*—each link contains a *next* pointer, but no *previous* one. Our goal, in this lab, is to implement the `NiceList<E>` interface with a singly linked list structure.

2 What you must do

As usual, create a new `lab-5` directory, and then grab some new starting source code:

- **On `remus/romulus`**, perform the following command:
  ```
  $ cp ~sfkaplan/public/COSC-112/lab-5/*.java .
  ```
- **On your own computer**, download and extract this zip file.

You will find the following in this collection of source code files:

- `NiceList.java`: The interface that defines what a `NiceList` class must do.
- `NiceSingleLinkedList.java`: The beginnings of our new linked list class. So far, it declares some data members and a constructor, but the rest of the methods must be filled in. You **are** allowed to add additional methods and data members as you see fit.
- `NiceSingleLink.java`: Defines one link in a singly linked-list chain.
- `SingleSentinel.java`: A sentinel for a singly-linked list. This class overrides the *value* getter and setter, making any attempted use throw an exception.
- `SingleHeadSentinel.java`: A special type of sentinel to be used only to mark the head of a list.

- `SingleTailSentinel.java`: A special type of sentinel to be used only to mark the tail of a list. Its *next* getter and setter are overriden, where any attempted use triggers an exception.

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

- There is a `_length` data member. If your methods keep this instance variable updated, then they may also use it. Doing so may simplify the code for a number of methods.
- The methods you must write should be 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 a `reverse()` method required by the interface. You may have this method initially do nothing, and completing it is not necessary for the assignment. Consider it a personal challenge. It must be written to reverse the order of the links in the list (and not just the values) without creating any new links.
- There is no class with a `main()` method. Write one. Use it to test your own code.

3 How to submit your work

Use the CS submission systems to submit your `NiceSingleLinkedList.java` source code file. Use one of these two methods:

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

This assignment is due on Sunday, Apr-09, 11:59 pm.