Introduction to Computer Science II
Lab 6
A generic, doubly-linked list with sentinels

1 Making things generic

The interface: Let's use generics for our `NiceList` interface, making it `NiceList<E>`, and likewise changing classes that implement it (e.g., `NiceArrayList<E>` and `NiceLinkedList<E>`). The interface now becomes:

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

The container classes: The `NiceArrayList<E>` class is provided. Recall that it uses an array to store the values, and automatically resizes the array when needed. Notice, when you look through it, that Java does not allow an array of type `E[]`; that is, generic arrays are not allowed, due to a problem called *type erasure*.[1] Consequently, array is still of type `Object[]`, and the `remove()` and `get()` methods must handle explicit casting themselves.

The `NiceLinkedList<E>` is again only partially implemented. Your job will be to complete it, much as you did with Lab-5.

[1] Don't worry about exactly what this is. It's a known limitation of Java arrays.

2 Changing the linked list structure

To implement the `NiceLinkedList<E>` class, you should note some changes to how it is now designed.

Double linking: Each `NiceLink<E>` is now not only generic (to match the rest of the code), but also contains both `_next` and `_prev` pointers. All of the methods that modify the structure of the chain must maintain both pointers in each link.

Sentinels: The list now uses *sentinel links* to mark the head and tail. We will go through, in lab, the use of inheritence to make the `HeadSentinel` and `TailSentinel` subclasses, each of which has appropriately liminted capabilities.

Adding these sentinels means that the `insert()` and `remove()` methods should no longer contain any special cases for operating at the ends of the list. Other changes (e.g., in the constructor, in `walkTo()`) also reflect the use of these special links as bookends.

3 What you must do

Get the code: Use the following link to download a zip file with a bunch of source code:

`https://bit.ly/AMHCS-2020S-112-16`

Complete the linked list implementation: You must complete write the `insert()` and `remove()` methods of the `NiceLinkedList<E>` class. You are welcome to borrow from your Lab-5 solution, keeping in mind what is different about how this linked list is structured.

4 How to submit your work

Submit your `NiceLinkedList.java` file via the CS submission system:

`https://www.cs.amherst.edu/submit`

This assignment is due on Sunday, Apr-05, 11:59 pm.