**CSCI 235 Spring 2020 Apr. 8**

**Project #6 (Due: 5 pm on Tuesday, Apr. 21)**

**Part A:**

**Documentation and style will be assessed as part of your grade**. **Follow the full code guidelines.**

**Task #1: Finishing IList**

* Update the IList class so that it implements the interface, List235.
* Fill in the methods that are not yet implemented. You can copy the code from the previous lab to the IList class. Test every method.

**Task #2: Writing a recursive list**

* Complete the remaining methods in RList and the corresponding recursive RNode methods.
  + Notice how the methods in the node class that modify the list return **a node representing** **the *rest* of the modified list**.
  + A node keeps itself in the list by modifying its next link and returning this.
  + A node removes itself from the list by returning a different node.
* Test every method.

Example:

**public class RList implements List235** {

private RNode head;

public RList() {

head = null;

}

**public void printList()** {

System.out.print("[ ");

if (head != null) {

head.print();

}

System.out.println(" ]");

}

/\*\*

\* Get the value of the item at a specified position.

\* @param position The position in the list.

\* @return The value at position in the list.

\* PRECONDITION: position >= 0 and position < length of this list

\*/

**public int elementAt(int position)** {

if (head != null) return head.elementAt(position);

throw new RuntimeException("The list is empty");

}

/\*\*

\* Delete the first occurrence of item in this list, if any.

\* @param item The item to delete

\*/

**public void deleteFirstOccurrence(int item)** {

if (head != null) {

head = head.deleteFirstOccurrence(item);

}

}

…

In RNode:

**public class RNode** {

private final int datum;

private RNode next;

public RNode(int datum, RNode next) {

this.datum = datum;

this.next = next;

}

public int datum() { return datum; }

public RNode next() { return next; }

public void print() {

System.out.print(datum+" ");

if (next != null) {

next.print();

}

}

public int elementAt(int position) {

if (position == 0) return datum;

else return next.elementAt(position-1);

}

public RNode deleteFirstOccurrence(int item) {

// Prep #18 for Apr. 13

**CSCI 235 Spring 2020**

**Project #6 (Due: 5 pm on Tuesday, Apr. 21)**

**Part B:**

**Documentation and style will be assessed as part of your grade**. **Follow the full code guidelines.**

Download four more Java files from the course site: DList, DLNode, List235Maker, and ListTest.

**Doubly-linked List**

Each node in a doubly-linked list has two links, one for forward (next) and the other for backward (previous). The classes DList and DLNode provide a framework for implementing a doubly-linked list, based on the iterative version.

The DLNode class has methods for splicing a node into or out of a list. You need to fill in the body of spliceAfter.

Write the bodies of DList methods according to the instructions. You need to provide findLast(), which will make insertAtBack() work. The other missing methods can be copied from your IList with some changes.

Use printListReverse() to see if your methods in DLNode and DList properly work. The method prints the list in reverse order by following its previous links.

**Testing the lists:**

You need two files: List235Maker and ListTest. Your lists will be evaluated by this program.

The ListTest class provides a main method that you can use to test the different versions of List235 interface. Use the following commands to test your lists:

$ java ListTest IList (for testing IList)

$ java ListTest RList (for testing RList)

You can add other tests to the testing driver if necessary.

**What to submit:**

* Upload three Java files, IList, RList, and RNode to Schoology.
  + DList and DNode are optional.

j

}