This document provides the necessary instructions for completing the Week 3 lab exercises. This exercises are mainly slight modifications of Week 3 class examples already provided.

**Exercise 1: CircularlyLinkedList**

Give 3-4 examples of cyclic order systems that can be implemented using a circularly linked list. Explain why using a circularly linked list is more efficient than using a regular linked list.

**Exercise 2: Using CircularlyLinkedList**

Run the **CircularlyLinkedList.java** from the Lesson3Examples. Create a new **CircularlyLinkedList** of integers. Use *addFirst* and *addLast* methods to populate the list with some integer numbers. Print the list, perform a *rotate* operation and print the list again.

**Exercise 3: CircularlyLinkedList**

Give an implementation of the *size( )* method for the *CircularlyLinkedList* class, assuming that we did not maintain *size* as an instance variable. **Hint**: Use a loop to traverse the list while counting.

**Exercise 4: Equivalence testing**

Run the **ArrayEquivalencyTest.java** from Lesson3Examples. Practice the use of **==** operator, **equals** method of *Object* class, and **equals** method of *Arrays* class.

Create two two-dimensional arrays of characters. Test the **shallow comparison** using *equals* method of *Arrays* class and deep comparison using *deepEqual*s method of *Arrays* class. Explain the difference.

**Exercise 5: Cloning**

Run the SinglyLinkedList.java from Lesson3Examples. Experiment with a list of different types of elements.

In the code of clone() method below, note the added line:

**//This line is absent in book's code. This is a bug. This line must be present.**

**other.tail = otherTail;**

This bug has been identified by my colleague Arindam Das, and has been reported to the main author of the textbook. Add the missing line in the Lesson3Examples code of the *clone()* method. Run the example again. Interpret the results.

**public** SinglyLinkedList<E> clone() **throws** CloneNotSupportedException {

// always use inherited Object.clone() to create the initial copy

SinglyLinkedList<E> other = (SinglyLinkedList<E>) **super**.clone(); // safe cast

**if** (size > 0) { // we need independent chain of nodes

other.head = **new** Node<>(head.getElement(), **null**);

Node<E> walk = head.getNext(); // walk through remainder of original list

Node<E> otherTail = other.head; // remember most recently created node

**while** (walk != **null**) { // make a new node storing same element

Node<E> newest = **new** Node<>(walk.getElement(), **null**);

otherTail.setNext(newest); // link previous node to this one

otherTail = newest;

walk = walk.getNext(); //in the original list

}

**//This line is absent in book's code. This is a bug. This line must be present.**

**other.tail = otherTail;**

}

**return** other; //returns the clone

}