This document provides the necessary instructions for completing the Week 7 lab exercises.

**Exercise 1: Dynamic Arrays**

The ArrayList class from Lesson7Examples, uses the doubling strategy to increase the size of the aary when its capacity is reached. We know from amortized analyses, lecture 7, that **n** push operations will run in **O(n)** time.

What would happen if you, instead of increasing the size of the array by two, increase it by **N/2**, where is the size?

Provide the implementation and explain the result in terms of Big O notation. Hint: **n** push operations will still run in **O(n)** time.

**Exercise 2: LinkedPositionalList**

Create a main method in **LinkedPositionalList** class from Lesson7Examples. Test the following PositionalList methods:

1. addFirst
2. addLast
3. addBetween
4. remove
5. first
6. last
7. toString

Use **String** objects and then **Integer** objects to test the methods.

**Exercise 3: LinkedPositionalList**

Create a class CustomersList, similar to FavoritesList.java from Lesson7Examples. Create a nested Customer class in CustomerList, similar to Item class in FavoritesList. Create a LinkedPositionalList of customers. Add the same methods as in FavoritesList and test the CustomersList in the same ways as in FavoritesList.

**Exercise 4: Iterators**

Redesign the **Progression** class, from Lesson1Examples, so that it formally implements the Iterator<long> interface. You will have to implement methods **hasNext**, **next,** and **remove**. Write a **main** method to test your implementation. The implementation could make calls to existing methods in Progression class. Hint: For an infinite progression, you may have hasNext() return true.