Tutorial 8

Objectives

Practice with Linked Lists and the Java Collections Framework

---

Attendance Quiz

Please log on to cuLearn using one of the computers in the tutorial room and complete the attendance quiz. You can only access the quiz if you log in using one of the computers in the lab. You cannot use a laptop for this. This is a time limited quiz. Be sure to do this as soon as you arrive.

At the end of the tutorial a TA will assign you a grade, call it G, which is 0, 1 or 2, depending on the progress you make during the tutorial. If you spend your time reading mail and chatting with your friends you will receive 0. If you have completed the attendance quiz on time then G will be your tutorial grade. If you have not completed the attendance quiz on time, then your tutorial grade will be max(0, G - 1/2). Each tutorial grade will therefore be one of 0, 0.5, 1.5 or 2.

In order to receive full marks for this tutorial, you must fully complete part 1. If you intend to leave in the first hour of the tutorial then you are expected to complete the entire tutorial.

---

0) Make a directory called **comp1406t8**. Download all the tutorial files to this directly. Run the **javadoc** program to generate the API for the classes.

1) Recall that a **list** is a collection of ordered items x_0, x_1, x_2, …, x_n-1.
Here, the length of the list is n.

Download the **Node**, **LList**, **LinkedList** and **TestLL** classes. Read through the classes and be sure you understand all of the **Node** class. Look at what is provided and what is missing in the **LList** class. Your task for this tutorial is to complete the **LList** class and build up the **TestLL** class.

1. implement the **find(String s)** method: returns first k such that s_k = s, returns -1 if s is not in

the list
2. implement the **set(int k, String s)** method: sets x_k to be s.
3. implement the **removeFront()** method: removes x_0, the list adjusts itself to be length n-1.
4. implement the **remove(int k)** method: removes x_k, the list adjusts itself to be length n-1

Add code to the **TestLL** program to test the **find** method.

Change the *strings* array to be sure your code works correctly with different inputs.

2) The **HashSet** class in the Java Collections Framework implements the Set ADT. A set is an unordered collection of unique items. Important methods include **add()**, **contains()**, **remove()**, **isEmpty()** and **size()**.

https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html

```
import jave.util.HashSet;
...
HashSet<Integer> set = new HashSet<Integer>();
```

Run the **SetORList** program. Rerun the program several times with different values of *size* (see the main method). Try 1000, 10000, 20000, 50000, etc.

What can we say about using list versus a set for simply checking if an item is an element of the collection?

3) What is the probability that there are two people in a room that have the same birthday? How many people do we need in the room so that the probability is 0.5? This is a problem that you will solve in COMP2804. It is called the *Birthday Paradox*.

https://en.wikipedia.org/wiki/Birthday_problem

In this problem, you will investigate the birthday problem using random numbers. Write a program, called **BirthdayParadox**. Your program will randomly generate numbers in a set range until you have a number that is repeated twice. Do this for several ranges (one of the ranges should be the entire range of non-negative integers[0,2^31-1], another should be range [0,365]). Your program should output the size of the range, the number of numbers needed to find a repeated one, and that number squared.

How do you do this? Keep all the numbers that you see in a set. Each time you generate a new random number, check if the number is in the set already or not. If the number is in the set you have a match (and you are done). If the number is not in the set then add it to the set and generate another random number.

Are your results consistent with the results of the birthday problem?