Data Structures and Algorithms
Week 4 problem sheet

## A. ADTs vs data structures

1. Explain which of the following are *abstract data types*, and which are (concrete) *data structures*:
	* linked list
	* array
	* queue
	* binary search tree
	* map
	* binary tree
*
1. Suppose we wanted an abstract data type that represents a *Door*, for a computer system that remotely controls all the doors in the building (whether they are locked, unlocked, open, or closed).
* What *operations* would you define for this abstract data type?
* How would you implement it in Java (as an interface)?

## B. Collections API

1. Consider the four core interfaces of the Collections API: Set, List, Queue, Map. For each of the four assignments below, specify which of the four core interfaces is best-suited to the problem, and explain how to use an implementation of it to implement the assignment. You can complete the code for this in CollectionsDemo.java.
	1. Whimsical Toys Inc (WTI) needs to record the names of all its employees. Every month, an employee will be chosen at random from these records to receive a free toy.
	2. WTI has decided that each new product will be named after an employee but only first names will be used, and each name will be used only once. Prepare a list of unique first names.
	3. WTI decides that it only wants to use the most popular names for its toys. Count up the number of employees who have each first name.
	4. WTI acquires season tickets for the local lacrosse team, to be shared by employees. Create a waiting list for this popular sport.

## C. Priority Queues

1. Suppose we perform the following operations on a Priority Queue that uses ints for priority, and stores Strings as values:
* enqeue("alice", 1),
 enqeue("bob", 5),
 enqeue("carol", 3),
 deqeue(),
 deqeue(),
 deqeue()
	1. What Java code would we use to implement this (using the linked list implementation from the sample code)?
	2. What will be the contents of the queue after each operation?

## D. Graphs

1. Describe (in words) an algorithm to count the number of edges in a **directed** graph using the adjacency matrix.
2. Describe how to count the number of edges in an **undirected** graph using the adjacency matrix.
3. Describe how to count the number of edges in an **undirected** graph using an **adjacency list** representation of a graph.
4. Describe how to generate the list of neighbours for a given node using an **adjacency matrix**.
5. Describe how to generate the list of neighbours for a given node using an **adjacency list**.