CS 2413 001 Summer 2000 Homework #3

Quiz to be held in class 1:20pm Friday 30 June 2000

You will need extra sheets of notebook paper to write your answers on.

1. Put check marks to indicate the statements that are true for each data structure.

	Array	Vector	Linked List
Contiguous in memory			
Can access an element in O(1) time			
Length can change			
Causes fragmentation of the free store			
Can add a new element to the front of the list in O(1) time			

   Note: O(1) time is also called *constant time*, meaning that the amount of time the operation takes does not grow with input size.

2. **Write** the following methods for `LinkedList`:

   (a) `insertBefore (Object& target, Object& newObject)`: inserts a node containing `newObject` into the list immediately before the node containing `target`, or throws a `LinkedListNotFound` exception.

   (b) `insertAfter (Object& target, Object& newObject)`: inserts a node containing `newObject` into the list immediately after the node containing `target`, or throws a `LinkedListNotFound` exception.

   (c) `remove (Object& target)`: removes all nodes containing `target` from the list.

   (d) Two linked lists are *equivalent* if they contain the same set of elements, regardless of order. Overload the == operator to determine whether two linked lists are equivalent. You should not assume that either list has a particular order, nor that the two lists have the same order.

3. For each of the methods in the previous question, **what is the time complexity? Explain your answer.** You do not need a formal proof, nor to choose values for c and n_0.

4. A linked list is *ordered* if it stores objects in a sorted order based on some data field (or fields) of the objects, called the *key* field(s). (Therefore, the type of the key field would need to have inequality operators defined.) **What would be the time complexity** of a binary search function on an ordered linked list? (Hint: what is the time complexity of the `LinkedList` method `infoAt`? What is the time complexity of binary search?) Note: you do not have to implement the linked list binary search function.

5. **Rewrite** the `merge` function (*OODS*, chapter 3) to operate on two linked lists instead of two arrays. You may assume that both linked lists are sorted in ascending order.

References

S. Rhadakrishnan, L. Wise & C. N. Sekharan, *Object-Oriented Data Structures Featuring C++*, 1999.

A. Drozdek, *Data Structures and Algorithms in C++*, PWD Publishing Co., New York, 1996.