CMSC 341 Data Structures

List Review Questions

Please refer to the textbook for List, ListNode, and ListIterator class definitions. These definitions are the same as those found in the class notes. You may assume that all member functions of these classes have been written and work properly when answering the questions below. These questions will help test your understanding of the list material discussed in class and in the text. These questions are only a study guide. Questions found here may be on your exam, although perhaps in a different format. Questions NOT found here may also be on your exam.

1. Write a new member function of the List class named **ReversePrint** that uses an iterator to display the elements of the List in reverse order. The data elements should be enclosed in angle brackets ("< >") and separated by commas. Do not construct a copy of the list that is in reverse order, just use iterators. The prototype for **ReversePrint( )** is shown below

```
void ReversePrint ( );
```

2. Write a new function named **Splice( )** whose prototype is shown below. ***Note that Splice( )is not a member function of the List class.*** This function "splices" L2 into L1 at the specified position (**`pos`** is a ListIterator over L1). If **`pos`** is past end, **Splice( )** does nothing. For example, suppose L1 = {1, 2, 3, 4, 5} and L2 = {10, 20, 30} and **`pos`** is constructed as list1.iterator( ) and then advanced twice (pos.next()) so it is positioned before the "3". Then the function call **`Splice( L1, L2, pos )`**; causes L1 to become { 1, 2, 10, 20, 30,3, 4, 5} and L2 is unchanged.

   What is the asymptotic performance of **Splice( )**? Bear in mind that there are two lists which may be of different lengths.

```
public static<T>
void Splice( List<T> L1, const List<T> L2, ListIterator<T> pos);
```

3. Complete the following table of Big-Oh, worst-case asymptotic time performance for the given operations and implementations of the List ADT. Give your answers in terms of n, the number of elements in the list.

Operation	Double Linked List	ArrayList
insert (index, x)		
ind(x)		
remove(x)		
makeEmpty		
add(x)		

4. Suppose you are provided with a set of N random numbers which are to be inserted into a sorted List (smallest to largest). What would be the worst-case asymptotic time performance for building the entire list?

5. One advantage of using a double-linked list is the availability of bi-directional list iterators – iterators that move "forward" and move "backward". Suppose that in order to save memory, you (or your boss) decide to use a single linked list. Can a single linked list still support bi-directional iterators? If so, briefly describe how the iterator would move forward and backward. Be sure to include the Big-Oh performance of these operations. If bi-directional iterators are not possible with a single linked list, explain why not.

6. Describe the advantages (if any) and disadvantages (if any) of each of the List ADT implementation – singly linked list, doubly linked list, and arraylist.

7. Write the code for a new LinkedList method that removed the node AFTER the one specified by the parameter. The signature of this new method is provided below.

```
private AnyType removeAfter( Node<AnyType> p )
```