CMSC 341 Data Structure Midterm I review

These questions and those in your homework and projects 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. Note that there will be programming in the midterm.

Asymptotic Analysis Review

These questions will help test your understanding of the asymptotic analysis material discussed in class and in the text.

1. What is the purpose of asymptotic analysis?
2. Define "Big-Oh" using a formal, mathematical definition.
3. Let $T_1(x) = O(f(x))$ and $T_2(x) = O(g(x))$. Prove $T_1(x) + T_2(x) = O(\max(f(x), g(x)))$.
4. Let $T(x) = O(cf(x))$, where c is some positive constant. Prove $T(x) = O(f(x))$.
5. Let $T_1(x) = O(f(x))$ and $T_2(x) = O(g(x))$. Prove $T_1(x) * T_2(x) = O(f(x) * g(x))$
6. Prove $2^{n+1} = O(2^n)$.
7. Prove that if $T(n)$ is a polynomial of degree x, then $T(n) = O(n^x)$.
8. Number these functions in ascending (slowest growing to fastest growing) Big-Oh order:

Number	Big-Oh
	$O(n^3)$
	$O(n^2 \lg n)$
	$O(1)$
	$O(\lg^{0.1} n)$
	$O(n^{1.01})$
	$O(n^{2.01})$
	$O(2^n)$
	$O(\lg n)$
	$O(n)$
	$O(n \lg n)$
	$O(n \lg^5 n)$

9. Determine, for the typical algorithms that you use to perform calculations by hand, the running time to:

    a. Add two N-digit numbers
    b. Multiply two N-digit numbers

10. What is the asymptotic performance of each of the following?
    Select among:

    a. $O(n)$ b. $O(n^2)$ c. $O(n \lg n)$ d. $O(n^3)$ e. $O(\lg n)$
    f. $O(1)$ g. $O(n!)$ h. None of these

    (a) _____ Squaring each element of an NxN matrix

    (b) _____ Finding the smallest value in a sorted array of N integers

    (c) _____ Finding a value in a sorted array using binary search

    (d) _____ Pushing N elements onto a stack, then popping them and printing them

    (e) _____ Finding the largest 3 values in an unsorted array

11. What is the asymptotic performance of the following Java code fragment?
    Justify your answer.

```
for (int i = 0; i < N; i++)
{
      for (int j = 10; j >= 0; j--)
      {
            int count = 1;
            while (count < N)
                  count *= 2;
      }
}
```

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.

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 )
```

Stack and Queue Review

The class definitions for stack and queue are provided at the end of the questions in this section.

Stacks

1. Using only the operations of the stack, write a static function that determines if a string is a palindrome (i.e. reads the same backward and forward; e.g. "level"). The prototype for this function is given below.

```
public static
bool isPalindrome(String theString);
```

2. What is the output of the following code?

```
int [] values = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19 };
Stack<Integer> s = new Stack<Integer>( );

for (int 1 = 0; i < values.length; i++)
     s.push( values[ i ] );

int n = 25;
for (int i = 0; i < 4; i++)
{
     n += s.pop( );
}
for (int i = 0; i < 2; i++)
```

```
{
     n -= s.pop( );
}
System.out.println( n );
```

3. Discuss the advantages and disadvantages if the text's array implementation and the lecture notes layered implementation of the stack ADT. At a minimum, consider the asymptotic time performance of the ***isEmpty( ), pop( )*** and ***push()*** operations.

4. Using only the operations of the stack given in the class, write a Java function that returns a copy of the user specified stack. The prototype for the function is given below.

```
public static<AnyType>
Stack<AnyType> CopyStack(Stack<AnyType> otherStack)
```

Queues

1. Using the operations of the stack and queue, write a static function that determines if a string is a palindrome (i.e. reads the same backward and forward; e.g. "level"). The prototype for this function is given below.

```
public static
bool isPalindrome(String theString );
```

2. Suppose that Q is an initially empty array-based queue of size 5. Show the values of the data members front and back after each statement has been executed. Indicate any errors that might occur.

```
Queue<Character> Q( 5 );     front = _____    back = _____

Q.enqueue ( 'A' );           front = _____    back = _____

Q.enqueue ( 'B' );           front = _____    back = _____

Q.enqueue ( 'C' );           front = _____    back = _____

char c = Q.dequeue( );       front = _____    back = _____

Q.enqueue ( 'A' );           front = _____    back = _____
```

3. Discuss the advantages and disadvantages of the link list and array-based implementations of a queue.

4. Describe three "real life" applications of a queue.

5. Explain how to implement a queue using two stacks.

Definition of the Stack Class

This is the definition of the array based stack from the text:

```
public class Stack<AnyType>
{
     public Stack ( int capacity );
     public boolean isEmpty( );
     public boolean isFull( );
     public void makeEmpty( );
     public AnyType pop( );
     public AnyType top( );
     public void push( AnyType element );
     private ArrayList< AnyType > theArray;
     private int topOfStack;
}
```

Definition of the Queue Class

This is the definition of the array based queue from the text:

```
public class Queue<AnyType>
{
     public Queue( int capacity );
     public boolean isEmpty( );
     public boolean isFull( );
     public AnyType getFront( );
     public void makeEmpty( );
     public AnyType dequeue( );
     public void enqueue( AnyType element );
     private ArrayList< AnyType > the Array
     private int currentSize, front, back;
     private void increment( int x );
}
```

General Tree Review

These questions will help test your understanding of the general tree material discussed in class and in the text.

General Trees

1. Define *tree.*

2. Define *k-ary tree.*

3. For any tree, T, define the following

a. path in T
b. length of a path in T
c. height of a node in T
d. depth of a node in T
e. height of T
f. depth of T
g. external node
h. internal node
i. leaf

4. Given the drawing of an arbitrary tree, draw the first-child, next-sibling representation of the tree.

5. Given the first-child, next-sibling representation of a tree, draw the tree.

6. Prove that there are $n - 1$ edges in any tree with n nodes.

7. What is the worst-case Big-Oh performance for the **insert, find** and **remove** operations in a general tree? Why is this so?

8. Write a recursive member function of the "static K-ary" tree class that counts the number of nodes in the tree.

Binary Trees

1. Define *binary tree*, *full binary tree, complete binary tree* and *perfect binary tree*

2. Prove that a perfect binary tree of height h has 2^h leaf nodes.

3. Prove that a perfect binary tree of height h has $2^{h+1} - 1$ nodes.

4. Prove that a full binary tree with n internal nodes has $n + 1$ leaf nodes.

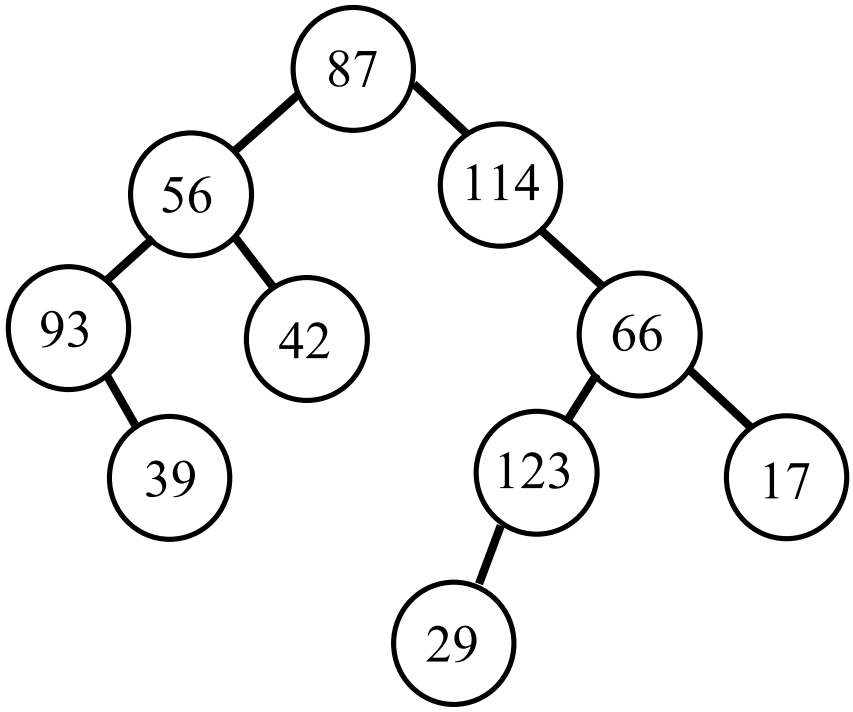
5. Prove that in any binary tree with n nodes there are $n + 1$ "null pointers".

6. Suppose that you have two traversals from the same binary tree. Draw the tree.
   pre-order: A D F G H K L P Q R W Z
   in-order: G F H K D L A W R Q P Z

7. Write a recursive member function of the BinaryTree class that counts the number of nodes in the tree.

8. Write a recursive member function of the BinaryTree class that counts the number of leaves in the tree.

9. Given the following binary tree containing integers, list the output from a *pre-order traversal*, an *in-order traversal*, a *post-order traversal*, and a *level-order traversal* of the tree.



Binary Search Tree (BST)

These questions will help test your understanding of the binary search tree material presented in class and in the text.

1. Define ***binary search tree***.

2. Write the syntactically correct Java code for the private BST member function `find(T x)` whose function prototype is found below and cannot be changed. This private function is initially called by the public `find(T x)` with a reference to the root. If x is in the tree, return element in the node; otherwise return NULL.

```
private T find(BinaryNode<T> root, T x)
```

3. Describe how deletion is performed in a BST. Be sure to mention all relevant cases.

4. The number of comparisons to build a BST of n elements is $O(n^2)$ in the worst case and $O(n \lg n)$ in the best case. Describe the best and worst case and explain why this is so.

5. Insert the following Java reserved words into an initially empty BST in the order given: *break, operator, if, typedef, else, case, while, do, return, byte, for, true, double, void.*

    a. Show the output from the pre-order, post-order, in-order, and level-order traversals of the BST.

b. Which word is the predecessor of *while*?
c. Which word is the successor of *for*?
d. Draw the new tree that results from each of the following operations. Each problem is separate and begins with the tree created above.
    i. Insert *int* and *float*, in that order.
    ii. Delete *double*, *case*, and *typedef*, in that order.

6. Prove that if a node in a BST has two children, its successor has at most one child.