Name ____ Section Number ________

CS210 Exam #4 *** PLEASE TURN OFF ALL CELL PHONES*** Practice
All Sections Bob Wilson

OPEN BOOK/OPEN NOTES
You will have all 90 minutes until the start of the next class period.
Spend only about one minute per point on each question to complete the exam on time.

1. Data Structures (20 Points)

a. Explain why you may need to add a stack data structure to your code when you re-implement a recursive method without using recursion.

b. Do you need to add a stack when you re-implement a method with “tail recursion”? (Y/N) _____
Explain why or why not.

2. Java Language and Object Oriented Programming (20 Points)
For each of the following snippets of Java code, indicate whether the syntax and/or usage of Object Oriented Programming is correct or not. If correct, explain the semantics (what it does). If not correct, correct it and explain your reasons.

```
a. public class Polynomial             // Correct syntax? Yes_____ No ______
     {                            // Correct, give reason, and/or explain semantics
          private ArrayList<Integer> coefficients;
          // constructor
          public void Polynomial (int...coefficients)
          {
           // initialize this.coefficients from coefficients array
          }
     }
```

```
b.   String s = function(“Hello”); // Correct syntax? Yes_____ No ______
     . . .                         // Correct, give reason, and/or explain semantics
     // elsewhere in the same class
     private String function(String parameter)
     {
          return “Goodbye”;
     }
```

```
c.   int i, j;                     // Correct syntax? Yes_____ No ______
     . . .      // code to initialize values of i and j
     try                           // Correct, give reason, and/or explain semantics
     {
          System.out.println(i/j);
     }
     catch (Exception e)
     {
          System.out.println(“j is zero”);
     }
```

```
d.   ArrayList<String> letters =   // Correct syntax? Yes_____ No ______
          new ArrayList<String>(); // Correct, give reason, and/or explain semantics
     letters.add(‘a’);
```

```
e.   float height = 65.0;          // Correct syntax? Yes_____ No ______
                                   // Correct, give reason, and/or explain semantics
```

3. Data Structures and their Implementation (30 Points)

a. Show the contents of an array used to implement a binary search tree using the computational strategy when the elements A-G are added in the following order: C, A, E, D, G, B, and F. The letter indicates the natural ordering of the objects.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
___	___	___	___	___	___	___	___	___	___	___	___	___	___	___

b. To remove node E, what rule would you use to replace it and why?

c. Show the steps required to remove the object for the letter E from the array and maintain the search tree integrity. (Note: You may not need all of the following lines.)

Element	from Index	to Index
E	______	None (Removed from Tree)
__	______	______
__	______	______
__	______	______
__	______	______
__	______	______

d. If you add E back after removing it, it would be added as the ______ child of node ____.

e. Show the state of the array after removing E from the tree and re-adding it.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
___	___	___	___	___	___	___	___	___	___	___	___	___	___	___

4. Binary Tree Level Order Iterator (30 Points)
Given that you have a generic binary tree implemented in an array according to the class on the next page, write the code for an IteratorLevelOrder class that performs a level order traversal of the elements in the tree. The remove method does not need to be implemented.

In file IteratorLevelOrder.java:

You may tear this page off your exam to use in Problem 4.

In file BinaryTree.java:

```
import java.util.*;

public class BinaryTree<T> implements Iterable<T>
{
     private T [] tree;   // using computational strategy
     private int count;

     public BinaryTree(int size)
     {
          tree = (T []) new Object[size];
          count = 0;
     }

     // code for add, remove, etc. not shown

     public Iterator<T> iterator()
     {
          return new IteratorLevelOrder<T>(tree, count);
     }
}
```

Answer Key:

1. Data Structures

a. When a recursive method executes, the state of its variables at each level in the sequence of recursive calls is kept on the system stack. They are pushed on the system stack each time the method calls itself and they are popped from the system stack each time the recursive method returns.

When you rewrite the recursive method without recursion, if there are any variables that need to be preserved from the first part of one iteration of the loop, reused during the next iteration of the loop, and returned to their original state upon resumption of the rest of the previous iteration, the code in the method must explicitly push those variables on a local stack and pop them off at the correct places in the code. The code simulates the behavior of the system stack.

b. No. You do not need a stack when you re-implement a tail recursive method without recursion because the recursive call is the last action in a tail recursive method and the variables at each level in the recursive calls are never used again after the recursive call returns. Even though the recursive calls preserve the state of these variables, there is no need for it. When you modify the code to use a loop, you do not need to restore the state of the variables from a previous iteration after any subsequent iteration.

2. Java Language and Object Oriented Programming

```
a. public class Polynomial          // Correct syntax? Yes_____ No __X___
     {                              // Correct, give reason, and/or explain semantics
          private ArrayList<Integer> coefficients;
          // constructor
          public ~~void~~ Polynomial (int...coefficients)
          {
           // initialize this.coefficients from coefficients array
          }
     }
```

The constructor should have no return type – not even void.

```
b.   String s = function("Hello"); // Correct syntax? Yes__X__ No ______
     . . .                         // Correct, give reason, and/or explain semantics
     // elsewhere in the same class
     private String function(String parameter)
     {
          return "Goodbye";
     }
```

The method named "function" has the correct parameter type and return type for the usage.

```
c.   int i, j;                     // Correct syntax? Yes__X__ No ______
     . . .     // code to initialize values of i and j
     try                           // Correct, give reason, and/or explain semantics
     {
          System.out.println(i/j);
     }
     catch (Exception e)
     {
          System.out.println("j is zero");
     }
```

The syntax is correct. If j is zero, the exception clause will execute and print its message. An ArithmeticException is a child of the Exception class so the exception type is correct.

```
d.   ArrayList<String> letters =   // Correct syntax? Yes_____ No __X___
          new ArrayList<String>(); // Correct, give reason, and/or explain semantics
     letters.add('a');
```

A character constant 'a' is a primitive data type and not a String object.

```
e.   float height = 65.0;          // Correct syntax? Yes_____ No __X___
                                   // Correct, give reason, and/or explain semantics
```

In Java, a decimal constant is a double type. This line of code need a cast to float.

```
     float height = (float) 65.0;
```

3. Data Structures and their Implementation

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
C	A	E	-	B	D	G	-	-	-	-	-	-	F	-

b. Node E had two children so you replace it with its in order successor F.

c. Show the steps required to remove the object for the letter E and maintain the search tree integrity.

Element	from Index	to Index
E	__2___	None (Removed from Tree)
F_	_13___	__2___

No other nodes need to be moved.

d. If you add E back after removing it, it would be added as the right child of node D.

e. Show the state of the tree after removing E and re-adding E.

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]	[13]	[14]
C	A	F	-	B	D	G	-	-	-	-	-	E	-	-

4. A level order traversal of a tree stored in an array is just in sequence through the array! However, null elements in the array are not included in the count. You have to base the return value of the hasNext method on the remaining count being greater than zero and to decrement the count in the next method only when returning a non-null value.

```
import java.util.Iterator;
public class IteratorLevelOrder<T> implements Iterator<T>
{
      private T[] tree;
      private int count;
      private int cursor;

      public IteratorLevelOrder(T[] tree, int count)
      {
            this.tree = tree;
            this.count = count;
            cursor = 0;
      }

      public boolean hasNext()
      {
            return count > 0;
      }

      public T next()
      {
            if (tree[cursor] != null)
                  count--;
            return tree[cursor++];
      }
}
```