CSC 403 Sections 901,902,910,911 Spring 2015  
Practice Problems

1. The Money class below is intended to represent monetary values in dollars and cents. Such a class might be used to prevent rounding errors. Add the necessary method(s) to this class so that Money objects can be stored in Java TreeSets in their natural ordering. If necessary, also specify any interfaces that the Money class must implement.

```java
public class Money {
    private int dollars, cents;

    public Money(int d, int c) {
        if (dollars < 0 || (c < 0 || c > 99))
            throw new IllegalStateException();
        dollars = d;
        cents = c;
    }

    public String toString() {
        // should be fixed so that cents has leading 0 if necessary
        return "$" + dollars + "." + cents;
    }
}
```

2. The Array class below implements some of the methods of the Java List interface. Fill in the iterator method for the Array so that it behaves appropriately.

```java
public class Array<T> implements Iterable<T> {
    private static final int MAX_ITEMS = 100;
    private T[] t = (T[]) new Object[MAX_ITEMS];
    private int size = 0;

    public boolean add(T item) {
        if (size < MAX_ITEMS) {
            t[size++] = item;
            return true;
        }
        else return false;
    }

    public boolean remove(T item) {
        for (int i=0; i<size; i++)
            if (t[i].equals(item)) {
                for (int j=i+1; j<size; j++)
                    t[j-1] = t[j];
                size--;
                return true;
            }
        return false;
    }
}
```
public Iterator<T> iterator() {
    // fill this in; replace the return statement
    return null;
}

3. Give the Θ-complexity of the following code, as a function of n.

   a. int sum = 0;
      for (int i=1; i<n; i+=2)
         for (int j=1; i<i; i*=2)
            sum++;

   b. int sum = 0;
      for (int i=1; i<n; i+=2)
         sum++;
      for (int i=1; i<n; i*=2)
         sum++;

4. Consider the following functions:

   a. f(n) = n^2 - 2n + 13
   b. f(n) = 2\log_2 n - 1

   For each, show that f(n) = Θ(g(n)) by finding constants C_1, C_2, and x such that C_1
   g(n) ≤ f(n) ≤ C_2 g(n) for all n ≥ x.

5. Starting with an empty binary search tree (no balancing involved in this problem),
   show the results of inserting in sequence the numbers 5, 2, 10, 1, 6, 4, 3, 11, 8, 7, 9.

6. Starting with the binary search tree pictured below, show the results of deleting in
   sequence the letters G, D, C, and F.

```
     D
    / \        
   C   G
  /   /        
 A   F   M
 /     /        
 B   L   N
 /   /        
 H   J
  /   /        
 I   K
```
7. Starting with the 2-3 tree below, inserting F, B, and C in sequence. Then delete E and A.

```
        _
       |D|
      /     \
     _      ___
    |A|     |E|G|
   ___    ___
```

8. Insert in sequence into an initially empty red-black tree: 3, 2, 1, 4, 7, 5, 6, 8. Then delete 1.

9. Fill in the keys in the root of the B-tree of order 6, pictured below. Then, show the results of inserting 3 and 13 into this tree.

```
        __________
       |            |
      /            |
     |              |
   ___            ___
  |2|4|6|8|  |10|1|2|4|6|1|8|  |20|2|2|4|  |2|6|2|8|3|0|  |32|3|4|3|6|  |
```

10. Write a method to calculate the height of a binary search tree, assuming that the method is a member of the BST403 class.

Recall that BST403 has an inner class called Node, which has instance variables left, right, and n (the number of nodes in the subtree).

```java
public int height() {
    return height(root);
}

public int height(Node n) {
}
```