1. [10 points] The absolute value of a negative number n is 0 - n. The absolute value of a non-negative number n is n. Complete the following Java method:

```java
/** Returns the absolute value of i */
public static int absolute( int i ) {
    // Your implementation here
}
```

2. [5 points] The ImRect class is supposed to be an immutable rectangle class. Cross out any method signatures that should not appear in an immutable class.

```java
public class ImRect {
    ...
    public boolean contains( int x, int y ) { ... } // Cross out
    public int area( ) { ... } // Cross out
    public boolean containedIn(ImRect r) { ... } // Cross out
    public ImRect intersect(ImRect r) { ... } // Cross out
    public void move(int dx, int dy) { ... } // Cross out
}
```
3. [15 points] What does this program (the “main” method from Foo.java) print?

```java
public class Bar {
    private Foo myFoo;
    Bar( Foo _f ) { myFoo = _f; }
    public void poke() { myFoo.react(); }
}

public class Foo {
    String reaction;
    int weight;
    Bar myBar = new Bar(this);
    public Foo(int _weight, String _reaction) {
        weight = _weight;
        reaction = _reaction;
    }
    public void add(int _w) { weight = weight + _w; }
    public void react() { System.out.println( reaction ); }
    public Bar getBar() { return myBar; }
    public int getWeight() { return weight; }
    public static void main( String[] args) {
        Foo foo1 = new Foo(3, "Ouch");
        Bar bar1 = foo1.getBar();
        Foo foo2 = new Foo(7, "Yumm");
        Bar bar2 = foo2.getBar();
        Foo foo3 = foo2;
        Bar bar3 = foo3.getBar();
        foo1.add(10); foo2.add(20); foo3.add(30);
        System.out.println(foo1.getWeight());
        System.out.println(foo2.getWeight());
        System.out.println(foo3.getWeight());
        bar1.poke(); bar2.poke(); bar3.poke();
    }
}
```
4. [20 points] Complete the method for translating an integer to its String representation in base k, where k is any integer between 2 and 16 inclusive. Note the use of `sb.insert(0, ch)` to insert a character `ch` at the beginning of a string buffer.

```java
static char[] Digits = // For bases 2 through 16
{ '0', '1', '2', '3', '4', '5', '6', '7',
'8', '9', 'A', 'B', 'C', 'D', 'E', 'F' };

public static String toBaseK( int i, int radix ) {
    // Handle special cases first
    if (radix < 2 || radix > 16) {
        return( " *** Radix " + radix + " out of range *** ");
    }
    if (i == 0) { return "0"; }

    // Then handle the usual cases ...
    StringBuffer sb = new StringBuffer();
    boolean negative = false;
    if (i < 0) {
        negative = true;
        i = 0 - i;
    }
    /**** Insert your code below ******/
    if (negative) {
        sb.insert(0, '-');
    }
    return sb.toString();
}
```
5. [25 points] We can define equality of two lists l1 and l2 like this:

- If l1 and l2 are both empty, then they are equal.
- If either l1 or l2 is empty, and the other is not, then they are not equal.
- If neither l1 nor l2 is empty, then they are equal if the first item in l1 is equal to the first item in l2, and the rest of l1 is equal to the rest of l2.

Complete the recursive comparison method based on the inductive definition:

```java
class Node {
    int val;
    Node next;
    public Node( int _val, Node _next ) {
        val = _val; next = _next;
    }
}
...
static boolean eqRecursive(Node l1, Node l2) {
    /* Fill in the body of eqRecursive below */
```
6. [25 points] Class DayDate provides a method for converting an integer day of year (from 1 to 365) into a month and day (from “Jan 1” to “Dec 31”). Fill in the missing part of the toDate method.

class DayDate {
    class Month {
        String name; // Name of month (3 letter abbreviation)
        int days; // Number of days in month
        Month( String _n, int _d ) { name = _n; days = _d; }
    }
    Month month(String _n, int _d) { return new Month(_n, _d); }
    Month[] months = {
        month("Jan", 31), month("Feb", 28), month("Mar", 31),
        month("Apr", 30), month("May", 31), month("Jun", 30),
        month("Jul", 31), month("Aug", 31), month("Sep", 30),
    }

    /* Convert day of year to month and day of month.
     * Example: toDate( 364 ) = "Dec 30"
     */
    String toDate( int day ) {
        if (day < 1 || day > 365) {
            return "Day of year " + day + " out of range";
        }
        int m = 0;
        /* Your code goes after this */
        return months[m].name + " " + Integer.toString(day);
        /* Your code goes before this */
    }
}