ECE 242
Fall ‘13 – Exam II
Profs. Wolf and Tessier

Name: __________________________________________________________
ID Number: ______________________________________________________

<table>
<thead>
<tr>
<th>Question</th>
<th>Maximum</th>
<th>Achieved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>19</td>
<td></td>
</tr>
<tr>
<td>Question 2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>Question 3</td>
<td>28</td>
<td></td>
</tr>
<tr>
<td>Question 4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>Question 5</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

This exam is closed book, closed notes. No electronic devices (including calculators) are allowed. Be concise, but show your work. Write legibly. When writing code, please indent appropriately and give your variables meaningful names.

Time: 120 minutes.
Question 1 (19 points):
Answer the following questions about the recursive definition of multiplication, \( m\times n = n + (m-1)\times n \) and \( 1\times n = n \), and the following implementation in Java. You may assume that \( m \geq 1 \) for all initial calls to this method.

```java
public static int mult(int m, int n) {
    if (m==1) {
        return n;
    }
    return n + mult(m-1, n);
}
```

a) For the computation of \( \text{mult}(5, 3) \), list all calls to the method \text{mult} with the values of \( m \) and \( n \). (5 points)

\[
\begin{align*}
\text{mult}(5, 3) \\
\text{mult}(4, 3) \\
\text{mult}(3, 3) \\
\text{mult}(2, 3) \\
\text{mult}(1, 3)
\end{align*}
\]

b) How many times is \text{mult} called for the computation of \( \text{mult}(1000, 3) \)? (2 points)

1000 times

c) How many times is \text{mult} called for the computation of \( \text{mult}(3, 1000) \)? (2 points)

3 times

d) Write an improved version of \text{mult} that is still recursive and makes use of the property \( m\times n = n\times m \). Your method should swap \( m \) and \( n \) when necessary to minimize the overall number of calls to \text{mult}. (10 points)

```java
public static int mult(int m, int n) {
    if (m==1) {
        return n;
    }
    if (m<n) {
        return n + mult(m-1, n);
    } else {
        return m + mult(n-1, m);
    }
}
```
Question 2 (20 points):
Answer the following questions regarding partitioning and sorting.

a) The following method performs partitioning on an array of integers. Describe the properties of the integer array after partitioning has been completed. (Do not describe the partitioning process itself in detail (e.g., what values are taken on by left and right, etc.). Only state what the effect of this partitioning code is on the array.) Note, that this partitioning code has nothing to do with any sorting algorithm. (4 points)

```java
private static void partition(int[] array) {
    int left = -1;
    int right = array.length;

    while (true) {
        while (left < array.length && array[++left] % 2 == 1) { ; }
        // search from left
        while (right > 0 && array[--right] % 2 == 0) { ; }
        // search from right
        if (left >= right) {
            // cross-over indicates end of partitioning process
            break;
        } else {
            // found misplaced items; swap
            int temp = array[left];
            array[left] = array[right];
            array[right] = temp;
        }
    }
}
```

The partitioning code separates the array into odd values (on the left side of array) and even values (on the right side of the array).

b) Assume that quicksort is used to sort an array of 7 integers. Assume that during partitioning, quicksort uses the rightmost element of the partitioning area in the array as the pivot element. Assume that the following 6 integer values are given and the 7th value (denoted by *) can be chosen by you:

```
[18, 5, 23, 2, 7, 10, *]
```

What value of * leads to the most balanced partitioning in the first iteration of quicksort? If multiple values are possible, choose any one. (4 points)

Either 8 or 9 are correct.
c) The following implementation of mergesort sorts an integer array in **descending** order. However, the code has three errors. Identify the errors and provide the correct replacement code. (12 points)

```java
public static void mergeSort(int[] values) {
    if (values == null || values.length <= 1) {
        // check for termination condition
        return;
    } else {
        int middle = values.length; // divide array
        int[] left = new int[middle];
        for (int i = 0; i < middle; i++) {
            left[i] = values[i];
        }
        int[] right = new int[values.length - middle];
        for (int i = 0; i < values.length - middle; i++) {
            right[i] = values[middle + i];
        }
        mergeSort(left); // recursively call sorting function
        mergeSort(right);
        int l = 0, r = 0; // combine sorted arrays
        for (int i = 0; i < values.length; i++) {
            if (r >= right.length || (l < left.length && left[l] > right[r])) {
                values[i] = left[l];
                l++;
            } else {
                values[i] = right[r];
                r++;
            }
        }
        return;
    }
}
```

Error 1 (write down full line of code and circle error above):
```java
int middle = values.length;
```

Replacement code 1: `int middle = values.length / 2;`

Error 2 (write down full line of code and circle error above): `mergeSort(left);`

Replacement code 2: `mergeSort(right);`

Error 3 (write down full line of code and circle error above):
```java
if (r >= right.length || (l < left.length && left[l] > right[r])) {
```

Replacement code 3:
```java
if (r >= right.length || (l < left.length && left[l] > right[r])) {
```
Question 3 (28 points):
Answer the following question regarding binary trees. Note that these binary trees are not binary search trees and may contain nodes with any value at any location. The node of the tree is defined as:

```java
public class Node {
    public Node left;
    public Node right;
    public int value;

    public Node(int v) {
        value = v;
    }
}
```

Assume the tree is defined as the following and properly initialized:

```java
public class Tree {
    public Node root;
}
```

a) Write a recursive method that returns the number of nodes in a binary tree. Assume the method is called with `nodeCount(t.root)`, where t is the tree. (6 points)

```java
public int nodeCount(Node n) {
    if (n==null) {
        return 0;
    }
    return nodeCount(n.left)+nodeCount(n.right)+1;
}
```
b) Write a recursive method that finds a value in the binary tree and returns the node with that value. If the value does not exist, the method should return null. Assume the method is called with `findValue(i, t.root)`, where `i` is the search value and `t` is the tree. Again, note that the tree is not a binary search tree. (12 points)

```java
public Node findValue(int i, Node n) {
    if (n == null) {
        return null;
    }
    if (n.value == i) {
        return n;
    } else {
        Node l = findValue(i, n.left);
        Node r = findValue(i, n.right);
        if (l != null) {
            return l;
        } else if (r != null) {
            return r;
        } else {
            return null;
        }
    }
}
```

c) Write a recursive method that compares two trees and returns true if they are identical in structure and node values. Assume the method is called with `areEqualTrees(t1.root, t2.root)`, where `t1` is one tree and `t2` is the other tree. (10 points)

```java
public static boolean areEqualTrees(Node n1, Node n2) {
    if (n1 == null || n2 == null) {
        return n1 == n2;
    }
    if (n1.value == n2.value) {
        return areEqualTrees(n1.left, n2.left) &&
                areEqualTrees(n1.right, n2.right);
    } else {
        return false;
    }
}
```
Question 4 (10 points):
Answer the following questions regarding binary search trees.

a) Assume the following binary search. Draw the binary search tree after each of the following operations. (Operations are cumulative.) Assume the remove operations on nodes with two children take a replacement node from the right child’s subtree. (6 points)

Operation 1: Remove 8

Operation 2: Insert 19

Operation 3: Remove 16
b) What is a potential problem with binary search trees when inserting nodes in a “bad” order? Your answer should be about one sentence. (2 points)

The binary tree may degenerate to a linked list (or a very unbalanced tree).

c) Why do red-black trees solve the problem stated in b)? (Do not describe how red-black trees work in detail or what rules they use. Instead state very briefly what they do avoid the above problem.) Your answer should be about one sentence. (2 points)

Red-black trees restructure the tree to ensure some sort of balance.
Question 5 (23 points):
Answer the following question regarding graphs.

a) List the order of nodes that are visited in the following graph when performing the state traversal. Assume that neighboring nodes are visited in alphabetical order. (8 points)

Depth-first traversal starting at node A:
A-B-D-G-C-F-H-E

Depth-first traversal starting at node G:
G-C-A-B-D-E-F-H

Breadth-first traversal starting at node F:
F-C-G-H-A-D-E-B

Breadth-first traversal starting at node B:
B-A-D-E-C-G-F-H
b) Consider a graph that represents a state machine that can match regular expressions. The state machine can match expressions based on inputs of ‘0’ and ‘1’. The vertex of this state machine is defined as follows and has exactly two edges, one for each possible input. If a neighbor for a given input does not exist, the edge is null.

```java
class Vertex {
    public Vertex zero;
    public Vertex one;
    boolean isAcceptingState;

    public Vertex (boolean accepting) {
        isAcceptingState = accepting;
    }

    public void setNeighbors(Vertex v0, Vertex v1) {
        zero=v0;
        one=v1;
    }
}
```

Write a match method that matches a string consisting of ‘0’s and ‘1’s using the state machine. The method should return true if the string leads to an accepting state and false in all other cases. Assume that the method is called with start pointing to the starting state of the state machine. (15 points)

```java
public static boolean match(Vertex start, String s) {
    char[] c = s.toCharArray();
    Vertex current = start;

    for (int i=0; i<c.length; i++) {
        if (current==null) {
            return false;
        }
        if (c[i]=='0') {
            current = current.zero;
        } else if (c[i]=='1') {
            current = current.one;
        } else { // not necessary
            return false;
        }
    }
    if (current==null) {
        return false;
    } else {
        return current.isAcceptingState;
    }
}
```