**Homework1 (Due Feb 16th)**

1. Recall that in a binary tree, a node may have 0, 1, or 2 children. In the following questions about a binary tree, the *height* of a tree is the length (number of edges) of the longest path. A tree consisting of just one node has height 0. [2pts]
2. What is the maximum number of nodes in a binary tree of height *d*?

**2^(d+1)-1**

1. What is the minimum number of nodes in a binary tree of height *d*?

**(d+1)**

1. What is the maximum height of a binary tree containing *n* nodes?

**(n-1)**

1. What is the minimum height of a binary tree containing *n* nodes?

**D=log2(n+1) -1**

1. A binary tree of height 3 [1pt]
2. Must contain a minimum of 4 nodes (True or False) **True**
3. Could contain 20 nodes (True or False) **False**
4. Suppose you insert the values 40, 30, 60, 50, and 10 into a binary search tree, and use the following function:

void someOrder(Node p)

{

if (p != null)

{ someOrder(p.right);

System.out.print(p.idata + “ ”);

someOrder(p.left);

}

}

What the output of calling someOrder(root) would be? [1pt]

**60, 50, 40, 30, 10.**

1. Explain why a Huffman coding table, instead of a Huffman tree, is used at the encoding side. Explain why a Huffman tree, instead of a coding table, is used at the decoding side. [1pt]

Hash table can be used for encoding, so the time complexity of encoding will be O(1). If Huffman tree is used for encoding, the time complexity goes to O(log n). Using Huffman tree for decoding yields an algorithm with time complexity of O(log n). If a coding table is used, the time complexity easily goes to O(n).

1. Given four letters A, B, C and D with frequencies of 32, 20, 8, and 40, respectively. What are the Huffman **codes** of these four letters? [2pts]

**D=0**

**C= 100**

**B= 101**

**A= 11**

1. Show the red-black trees that result after successively inserting the keys 10, 20, 30, 40, 50, 60, 70, 80 into an initially empty red-black tree. You could use bold circles to denote red nodes in your solution. [3pts]

40

20 60

10 30 50 70

80

**Note: Non highlighted characters are black nodes.**