Binary Search Tree Stored in a Binary File

Address	Contents			
Root	0	160		
Free	8	184		
16	200	3	88	0
40	50	1	280	136
64	232			
88	150	2	0	208
112	0			
136	75	2	256	0
160	100	1	40	16
184	64			
208	175	3	0	0
232	112			
256	60	5	0	0
280	10	1	0	0
	Key	Count	Left	Right

Node Structure

```
private class Node {
        private long left;
        private int key;
        private int count;
        private long right;

        private Node(long l, int k, long r) {
        //constructor for a new node
                left = l;
                key = k;
                right = r;
                count = 1;
        }
```

Node Structure

```
private Node(long addr) throws IOException{
//constructor for a node that exists and is stored in the file
        f.seek(addr);
        key = f.readInt();
        count = f.readInt();
        left = f.readLong();
        right = f.readLong();
        }

        private void writeNode(long addr) throws IOException {
        //writes the node to the file at location addr
                f.seek(addr);
                f.writeInt(key);
                f.writeInt(count);
                f.writeLong(left);
                f.writeLong(right);
        }
}
```

Instance Variables

```
private RandomAccessFile f;
long root; //the address of the root node in the file
long free; //the address in the file of the first node in the free list
```

Constructor

```
public BinarySearchTree(String fname, int mode)  throws IOException {
        File path = new File(fname);
        if (mode == CREATE && path.exists())
                path.delete();
        f = new RandomAccessFile(path, "rw");
        if (mode == CREATE) {
                root = 0;
                free = 0;
                f.writeLong(root);
                f.writeLong(free);
        } else {
                f.seek(0);
                root = f.readLong();
                free = f.readLong();
        }

}
```

Insert

```
public void insert(int k)  throws IOException {
//insert k into the tree
//if k is in the tree increment the count field associated with k
        root = insert(root, k);
}
```

Insert

```
private long insert(long r, int k)  throws IOException {
//insert k into the subtree whose root is found at r
//if k is in the tree increment the count field associated with k
        Node x;
        if (r == 0) {
                x = new Node(0, k, 0);
                long addr = getFree();
                x.writeNode(addr);
                return addr;
        }
        x  = new Node(r);
        if (k < x.key) x.left = insert(x.left, k);
        else if (k > x.key) x.right = insert(x.right, k);
        else x.count++;
        x.writeNode(r);
        return r;
}
```

Print

```
public void print()  throws IOException {
        print(root);
        System.out.println();
}

private void print(long r)  throws IOException  {
        if (r == 0) return;
        Node x = new Node(r);
        print(x.left);
        System.out.print(" ("+x.key+","+x.count+") ");
        print(x.right);
}
```