1)

Claim: # of edges in case of n nodes, *f(n)* = n - 1
Proof: Base case: *n = 1*, # of edges *f(1) = 1 -1 =0*
Inductive step: Assume # of edges for *n* nodes *f(n) = n – 1*
For *n + 1* nodes # of nodes, *f(n + 1) = f(n) + 1 = n -1 + 1 = n*

2)

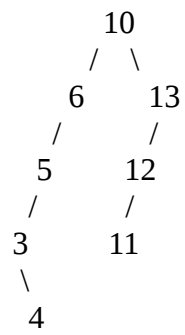
Max. height of BST -- *n-1*
Min. height of BST -- *floor($log_2 n$)*

Max. # of leaves – *floor((n+1)/2)*
Min. # of leaves – 1

3) Yes. Insertion to BST always creates a leaf node.

4) 2 * (4! + 2 * 3! + 2 * 2!) = 80
or
[C(6,3) * C(1,1) * C(2,1) * C(1,1)] * [C(3,3) * C(1,1) * C (2,1) * C(1,1)] = 80
or
1 * C(6,3) * 2 *2 = 80
1; since 4 must be the first insertion. 6 choose 3; to pick which 3 of the following 6 insertions will be the set {1, 2, 3}. 2; since the insertion ordering for subtree {1, 2, 3} can be 213 or 231. 2; since the insertion ordering for subtree {5, 6, 7} can be 657 or 675.

5)



6)

```
public void inorderPrint ( BinaryTreeNode <K > node ) {
// Create a stack to hold nodes
        Stack < BinaryTreeNode <K > > stack = new Stack < BinaryTreeNode <K > >() ;
        while (! stack . isEmpty () || node != null ) {
                if ( node != null ) {
                        stack . push ( node ) ;
                        node = node . getLeft () ;
                } else {
                        node = stack . pop () ;
                        System . out . print ( node . getKey () + " " ) ; // space as delimiter
                        node = node . getRight () ;
                }
        }
        System. out . println () ; // newline after all keys are printed
}
```

EC:

```
public void prune ( BinaryTreeNode <K > node , int depth ) {
        // Handle base case
        if ( depth == 0) {
                node . setLeft ( null ) ;
                node . setRight ( null ) ;
                return ;
        }
        // Recurse into subtrees , if they exist
        if ( node . getLeft () != null ) {
                prune ( node . getLeft () , depth -1) ;
        }
        if ( node . getRight () != null ) {
                prune ( node . getRight () , depth -1) ;
        }
}
```