LAB 5 **09/21/2012**

I500 /CSCI 609

The keys in a binary search tree are always stored in such a way as to satisfy the *binary-search-tree property*:

For all nodes x and y, if y belongs to the left subtree of x, then the key at y is less than the key at x, and if y belongs to the right subtree of x, then the key at y is greater than the key at x.

Each node has the following attributes:

- *p, left, and right, which are pointers to the parent, the left child, and the right child, respectively, and*
- *key, which is key stored at the node*

How a tree is ordered depends on how it is going to be accessed. The process of accessing each node in a tree is called a *tree traversal*. There are three types of traversal

- *Inorder. The ordering is: the left subtree, the current node, the right subtree.*
- *Preorder. The ordering is: the current node, the left subtree, the right subtree.*
- *Postorder. The ordering is: the left subtree, the right subtree, the current node.*

Inorder(node)

```
if node == null then return
else
    Inorder(node.left)
    print node.key
    Inorder(node.right)
```

preorder(node)

```
if node == null then return
```

```
else
    print node.key
    preorder(node.left)
    preorder(node.right)
```

postorder(node)

```
if node == null then return
else
    postorder(node.left)
    postorder(node.right)
    print node.key
```

Maxdepth(node)

```
if node == null return 0
else
    Maxdepth(node.left)
    Maxdepth(node.right)
    check the largest branch left or right
    return (left+1 or right+1)
```

Insertion

Suppose that we need to insert a node z such that k = key[z]. Using binary search we and a nil such that replacing it by z does not break the BST-property.

BtreeInsert(x, z, k)

```
 if x = NIL then return “Error”
 y ← x
 while true do {
      if key[y] < k
        then z←left[y]
     else z←right[y]
     if z = NIL break
 }
if key[y] > k then left[y] ←z
else right[p[y]] ← z
```

Problem: Complete the Inorder, preorder, postorder and insert function of the code given in btree.c file.