```
       public T findMax()
{
    return findMax(root);
}
private T findMax(BSTNode <T> t)
{
    while(t.right != null)
     t = t.right;
    return t.data;
}

public T findMinRec()
{
    return findMinRec(root);
}

private T findMinRec(BSTNode <T> t)
{
    if (t == null)
     return null;
    else if (t.left != null)
     return findMinRec(t.left);
    else
     return t.data;
}

public T findSuccessor(int tkey)
{
    findkey(tkey);
    return findSuccessor(current);
}

private T findSuccessor(BSTNode <T> t)
{
    return findMin(t.right);
}
```

```
public BSTNode <T> findSmallestKth(int k)
{
   return findKthSmallest(root,k);
}

private BSTNode <T> findKthSmallest(BSTNode <T> root, int k)
{
    if (root == null)
     return null;  // can't find anything, empty
    int numLeft = countNodes(root.left);
    if (numLeft + 1 == k) // current node
       return root;
    else if (numLeft >= k) // in left subtree
       return findKthSmallest(root.left, k);
    else
       return findKthSmallest(root.right, k  (numLeft + 1));
 }
```