Binary Search Tree Pseudo-Code

Inorder-Tree-Walk(x)  
1 Inorder-Tree-Walk(x.left)  
2 print x.key  
3 Inorder-tree-walk(x.right)

Tree-Search(x, k)  
1 if x == NIL or k == x.key  
2 return x  
3 if k < x.key  
4 return Tree-Search(x.left, k)  
5 else return Tree-Search(x.right, k)

Iterative-Tree\_Search(x, k)  
1 while x != NIL and k != x.key  
2 if k < x.key  
3 x = x.left  
4 else x = x.right  
5 return x

Tree-Minimum(x)  
1 while x.left != NIL  
2 x = x.left  
3 return x  
  
  
Tree-Maximum(x)  
1 while x.right != NIL  
2 x = x.right  
3 return x  
  
Tree-Successor(x)  
1 if x.right != NIL  
2 return Tree-Minimum(x.right)  
3 y = x.p  
4 while y != NIL and x == y.right  
5 x = y  
6 y = y.p  
7 return y

Tree-Insert(T, z)  
1 y = NIL  
2 x = T.root  
3 while x != NIL  
4 y = x  
5 if z.key < x.key  
6 x = x.left  
7 else x = x.right  
8 z.p = y  
9 if y == NIL  
10 T.root = z //tree T was empty  
11 elseif z.key < y.key  
12 y.left = z  
13 else y.right = z

Transplant(T, u, v)   
1 if u.p == NIL  
2 T.root = v  
3 elseif u == u.p.left  
4 u.p.left = v  
5 else u.p.right = v  
6 if v != NIL  
7 v.p = u.p

Tree-Delete(T, z)  
1 if z.left == NIL  
2 Transplant(T, z, z.right)  
3 elseif z.right == NIL  
4 Transplant(T, z, z.left(  
5 else y = Tree-minimum(z.right)  
6 if y.p != z  
7 Transplant(T, y, y.right)  
8 y.right = z.right  
9 y.right.p = y  
10 Transplant(T, z, y)  
11 y.left = z.left  
12 y.left.p = y