Binary Search Tree Pseudocode

Here are the procedures for binary search trees discussed in the book.

```
INORDER-TREE-WALK($x$)
if $x \neq$ NIL
    INORDER-TREE-WALK($x.left$)
    print $x.key$
    INORDER-TREE-WALK($x.right$)
```

```
PREORDER-TREE-WALK($x$)
if $x \neq$ NIL
    print $x.key$
    INORDER-TREE-WALK($x.left$)
    INORDER-TREE-WALK($x.right$)
```

```
POSTORDER-TREE-WALK($x$)
if $x \neq$ NIL
    INORDER-TREE-WALK($x.left$)
    INORDER-TREE-WALK($x.right$)
    print $x.key$
```

```
TREE-SEARCH($x, k$)
if $x ==$ NIL or $k == x.key$
    return $x$
if $k < x.key$
    return TREE-SEARCH($x.left, k$)
else
    return TREE-SEARCH($x.right, k$)
```

```
ITERATIVE-TREE-SEARCH($x, k$)
while $x \neq$ NIL and $k \neq x.key$
    if $k < x.key$
        $x = x.left$
    else
        $x = x.right$
return $x$
```

```
TREE-MINIMUM($x$)
while $x.left \neq$ NIL
    $x = x.left$
return $x$
```

```
TREE-MAXIMUM($x$)
while $x.right \neq$ NIL
    $x = x.right$
return $x$
```

```
TREE-PREDECESSOR($x$)
if $x.left \neq$ NIL
    return TREE-MAXIMUM($x.left$)
$y = x.p$
while $y \neq$ NIL and $x == y.left$
    $x = y$
    $y = y.p$
return $y$
```

```
TREE-SUCCESSOR($x$)
if $x.right \neq$ NIL
    return TREE-MINIMUM($x.right$)
$y = x.p$
while $y \neq$ NIL and $x == y.right$
    $x = y$
    $y = y.p$
return $y$
```

```
TREE-INSERT($T, z$)
$y =$ NIL
$x = T.root$
while $x \neq$ NIL
    $y = x$
    if $z.key < x.key$
        $x = x.left$
    else
        $x = x.right$
$z.p = y$
if $y ==$ NIL
    $T.root = z$ // tree $T$ was empty
elseif $z.key < y.key$
    $y.left = z$
else
    $y.right = z$
```

```
TRANSPLANT($T, u, v$)
if $u.p ==$ NIL
    $T.root = v$
elseif $u == u.p.left$
    $u.p.left = v$
else
    $u.p.right = v$
if $v \neq$ NIL
    $v.p = u.p$
```

```
TREE-DELETE$(T, z)$
if $z.left == \text{NIL}$
    TRANSPLANT$(T, z, z.right)$
elseif $z.right == \text{NIL}$
    TRANSPLANT$(T, z, z.left)$
else
    $y = $ TREE-MINIMUM$(z.right)$
    if $y.p \neq z$
        TRANSPLANT$(T, y, y.right)$
        $y.right = z.right$
        $y.right.p = y$
    TRANSPLANT$(T, z, y)$
    $y.left = z.left$
    $y.left.p = y$
```