ADS Lab 5

```
func insert( root, key):
    if not root:
        return TreeNode(key)
    elif key < root.val:
        root.left = insert(root.left, key)
    else:
        root.right = insert(root.right, key)

    balance = get_balance(root)
    height = 1+max( get_height(left) + get_height(right)

    if balance > 1 and key < root.left.val:
        return right_rotate(root)
    if balance < -1 and key > root.right.val:
        return left_rotate(root)
    if balance > 1 and key > root.left.val:
        root.left = left_rotate(root.left)
        return right_rotate(root)
    if balance < -1 and key < root.right.val:
        root.right = right_rotate(root.right)
        return left_rotate(root)

return root
```

```
func delete( root, key):
    if not root:
        return root
    elif key < root.val:
        root.left = delete(root.left, key)
    elif key > root.val:
        root.right = delete(root.right, key)
    else: if root.left == NULL:
            temp = root.right
            root = NULL
            return temp
        elif root.right == NULL:
            temp = root.left
            root = NULL
            return temp
    temp = get_min_node(root.right)
    root.val = temp.val
    root.right = delete(root.right, temp.val)

if root == NULL:
    return root

height = 1 + max(get_height(root.left),
                 get_height(root.right))
balance = get_balance(root)
```

```
if balance > 1 and get_balance(root.left) ≥ 0:
    return right_rotate(root)
if balance < -1 and get_balance(root.right) ≤ 0:
    return left_rotate(root)
if balance > 1 and get_balance(root.left) < 0:
    root.left = left_rotate(root.left)
    return right_rotate(root)
if balance < -1 and get_balance(root.right) > 0:
    root.right = right_rotate(root.right)
    return left_rotate(root)
return root.

func left_rotate(z)
    a = z.right ; temp = y.left
    a.left = z ; z.right = temp
    return a
func right_rotate(z)
    b = z.left ; temp = b.right
    b.right = z ; z.left = temp
    return b
```