```
Node* insert (node, data)
{
    if (node == NULL)
        return (newnode(data))
    if (data < node -> data)
        node -> left = insert (node -> left, data)
    else if (data > node -> data)
        node -> right = insert (node -> right, data)
    else
        return node
    // height
    node -> ht = 1 + max(node -> left -> ht,
                         node -> right -> ht)
    // balance
    bal = (node == N) ? 0 : (node -> left -> ht)
                          - (node -> right -> ht)

    if (bal > 1)
    {
        if (data < node -> left -> data)
            return right (Rotate (node))
        if (data > node -> left -> data)
        {
            node -> left = leftRot (node -> left)
            return rightRot (node)
        }
    }
```

```
if (bal < -1)
{
    data >
    if (node -> right -> data)
    return leftRot(node)
    if (data < node -> right -> data)
    {
        node -> right = rightRot(node -> right)
        return leftRot(node)
    }
}
return node
}
```

```
Node* del(root,data)
{
    if (root == NULL)
        return root

    if (data < root -> data)
        root -> left = delete(root->left,data)

elseif( data > root -> data)
    root -> right = delete(root->right,data)

else {
    if (root->left == NULL)||(root->right
    == NULL))

// height
root -> ht = 1 + max(root -> left->ht,
                     root -> right-> ht)

// balance
bal = root -> left -> ht - root-> right-> ht.

if ( bal >1)
{
    if (getbal( root -> left) >= 0)
        return rightRot ( root)
```

```
        if( getbal(root->left) < 0)
        {
            root->left = leftRot(root->left)
            return rightRot(root)
        }
    }
    if(bal < -1)
    {
        if( getbal(root->right) >= 0)
        return leftrot(root)
                (root->right)
        if(getbal(  ) > 0)
        {  ~~return leftrot(r~~
            root->right = rightRot(root->right)
            return leftRot(root)
        }
    }
    return root
}
```