```
newNode(val){
    node = new node()
    node->key = val
    node->left = NULL
    node->right = NULL
    node->height = 1
    return node
}

rightRotate(node){
    l = node->left
    r = node->right

    l->right = node
    node->left = r
    node->height = max(height(node->left), height(node->right)) + 1
    l->height = max(height(l->left), height(l->right)) + 1

    return x;
}

leftRotate(node){
    r = node->right
    l = node->left

    r->left = node
    node->right = l
    node->height = max(height(node->left), height(node->right)) + 1
    r->height = max(height(node->left), height(node->right)) + 1

    return node;
}
```

```
insert (node, key){
  if (node == NULL)
     return newnode(key)
  else if (key < node -> key)
     node -> left = insert (node -> left, key)
  else if (key > node -> key)
     node -> right = insert (node -> right, key)
  else return node

node -> height = 1+ max ( height (node -> left) , height (node -> right))

balance = getBalance (node)

if ( balance >1 and key < node -> left -> key)
     return rightRotate (node)
else if ( balance < -1 and key > node -> right -> key)
     return leftRotate (node)
else if ( balance >1 and key > node -> left -> key)
     node -> left = leftRotate (node -> left)
     return RightRotate (node)
else if ( balance < -1 and key < node -> right -> key)
     node -> right = rightRotate (node -> right)
     return leftRotate (node)

  return node
}

getBalance ( node) {
  if (node == NULL) return 0
  else return ( height(node -> left) - height (node -> right))
}
```

```
deleteNode(root, key) {
    if (root == NULL) return root
    else if (key < root->key)
        root -> left = deleteNode (root->left, key)
    else if (key > root->key)
        root->right = deleteNode(root->right, key)
    else
        if (root->left == NULL || root->right == NULL)
            temp = root->left ? root->left : root->right
            if (temp == NULL)
                temp = root
                root = NULL
            else root = temp
            free(temp)
        else
            temp = minValueNode (root->right)
            root->key = temp->key
            root->right = deleteNode(root->right, temp->key)

    if(root == NULL) return root
    root->height = 1+ max (height(root->left), height(root->right))
    balance = getBalance (root)
    if (balance > 1 && getBalance (root->left) >= 0)
        return rightRotate (root)
    else if (balance > 1 && getBalance (root->left) < 0)
        root -> left = leftRotate (root->left)
        return rightRotate (root)
    else if (balance < -1 && getBalance (root->right) <= 0)
        return leftRotate (root)
    else if (balance < -1 && getBalance (root->right) > 0)
        root -> right = rightRotate(root->right)
        return leftRotate(root)
    return root
```