14/10/2020 AVL tree

Revanth K
IBM18CS082

```
Node* insert (Node* node, int key)
{
    if (node == NULL)
        return (newNode (key));
  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));

    int balance = getBalance (node);
  if (balance > 1 && key < node -> left -> key)
        return rightRotate (node);
  if (balance < -1 && key > node -> right -> key)
     return leftRotate (node);
     if (balance > 1 && key > node -> left -> key)
  {    node -> left = leftRotate (node -> left);
       return rightRotate (node);
  }

  if (balance < -1 && key < node -> right -> key)
  {    node -> right = rightRotate (node -> right);
       return leftRotate (node);
  }

       return node;
}
```

REVANTH.R
1BM18CS082

```
Node * deleteNode (Node * root,
                   int key)
if (root == NULL)
    return root;
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))
{
    Node * temp = root -> left
                  root -> left = root -> right;
    if (temp == NULL)
    {   temp = root;
        root = NULL;
    } else
    * root = * temp          free (temp);
    }
    else { Node * temp = minValueNode (root -> righ
        root -> key = temp -> key;
    root -> right = deleteNode (root -> right,
                                temp -> key);
    } }
```