Lab 4: AVL Trees

Sahana L
1BM18CS089

```
//AVL Tree Node
 class Node
 {
  public:
     int key;
     Node *left
     Node *right
     int height
 };

//Insertion
 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 = inseet (node->right, key)
   else
      return node

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

  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~~ right)
    return leftRotate(node)
}

return node
}


// Deletion
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->right)
          root->key= temp->key
          root->right = deleteNode(root->right, temp->key)
        }
  }
```

Teacher's Signature :

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