Sujeev Kumar Singh
1BM18CS093
B5

14-10-20

Lab - 5 - Program 4

// Insert and Delet in AVL tree
// Insert

```
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);
    }
```

Sangeev Kumar Singh
1BM18CS093
B5


```
if (balance < -1 && key < node->right->key)
{
  node->right = rightrotate(node->right);
  return leftrotate(node);
}
return node;
}
```

Delete

```
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->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);
```

Sanjeev Kumar Singh
1BM18CS093
B5


```
else {
Node *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));
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;
 }
```