ADS - Lab 4 (AVL Trees)

```
class node
{ public:
        int data;
        node * left;
        node * right;
        int height;
};
node * getnode (int data)
{
      node *p = new node();
        p->data = data;
        p->left = null;
        p->right = null;
        p->height = 1;
        return p;
}
int tree_height (node *p)
{        if(p==NULL)
              return 0;
        return p->height;
}
node *    rot_right (node *b)
{       node *a = b->left;
        node *s = b->right;
          a->right = b;
           b->left = s;
          b->height = max (tree_height (b->left), tree_height (b->right))
                          +1;
          a->height = max (tree_height (a->left), tree_height (a->right))+1;
          return a;
}
```

Scanned with CamScanner

```
node * rot_left(node *a)
{       node *b = a->right;
        node *s = a->left;
          b->left = a;
          a->right = s;
        a->height = max(tree_height(a->right), tree_height(a->left))+1;
        b->height = max(tree_height(b->left), tree_height(b->right))+1;
        return b;
 }
int     bal(node *p)
{         if(p == NULL)
               return 0;
          return  height(p->left) - height(p->right);
}
node  * insertion(node *root, int data)
{         if (root == NULL)
              return getnode(data);
          if( data < root->data)
                  root->left = insertion(root->left, data);
          else  if (data > root->data)
                root->right = insertion(root->right, data);
          else
                return root;
          root->height =
                      max(tree_height(root->left), tree_height(root->right))
                          +1;
           int  bl = bal(root);
             if(bl>1  && data < root->left->data)
                     return   rot_right(root);
                if(bl<-1  &&  data > root->right->data)
                     return   rot_left(root);
            if (bl>1   && data > root->left->data)
            {
```

ABHAYAN
1BM18CJ001

```
    root->left = rot_left(root->left);
    return rot_right(root);
  }
  if (bf < -1 && data < root->left->data)
  {       root->right = rot_right(root->right);
          return rot_left(root);
  }
  return root;
}
node * deletion (node *root, int item)
{ if (root == NULL)
       return root;
  if (item < root->data)
       root->left = deletion (root->left, item);
  else if (item > root->data)
       root->right = deletion (root->right, item);
  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 = minvalue (root->right);
        root->data = temp->data;
        root->right = deletion (root->right, temp->data);
     }
  }
```

ABHAYAN
1BM18CS001

```
if(root == NULL)
        return root;
root->height = max(tree_height(root->left), tree_height(root->right))
                                                              +1;

 int  bl = bal(root);
  if(bl>1  &&  ~~bal~~ (root->left) >=0)
           return  rot_right(root);
  if(bl <-1  &&  bal(root->right)<=0)
           return  rot_left(root);
  if(bl>1   &&   bal(root->left)<0)
  {      root->left =  rot_left(root->left);
         return    rot_right(root);
  }
  if(bl<-1   &&  balance(root->right) >0)
  {      root->right = rot_right(root->right);
         return    rot_left(root);
  }
  return root;
}
```

Scanned with CamScanner