Hemanth Kumar Y.C
1BM18CS037
sec 'A'

# AVL TREE

```
struct node* insert(struct node *r, int data)
{
    if(r==NULL)
    {
        struct node *n;
        n= new struct node;
        n->data = data;
        r=n;
        r->left = r->right = NULL;
        r-> height =1;
        return r;
    }
    else
    {
        if(data<r->data)
        r-> left = insert(r->left, data);
        else
        r->right = insert(r->right, data);
    }
    r-> height = calheight(r);
    if(bf(r)==2 && bf(r->left)==1)
    {
        r= LL rotation(r);
    }
    else if(bf(r)==2 && bf(r->left==-1)
    {
        r= LR rotation(r);
    }
    return r
}
```

## Deletion

```
struct node * delete node (struct node *p, int data)
{
   if (p->left == NULL && p->right == NULL)
   {
      if ( p == this -> root)
           this -> root = NULL;
           delete p;
           return NULL;
      }
     struct node *t;
     struct node *q;.
   if (p->data < data)
   {
       p -> right = delete node (p-> right, data);
   }
   else if ( p->data > data).
      {
         p-> left = delete node (p-> left, data);
      }
     else
         {
           if (p->left != NULL)
           {
               q = inpre(p->left);
                p-> data = q->data;
         p->left = delete node (p->left, q->data);
           }
          else {
                q = inpre insuc( p -> right);
                 p -> data = q->data;
                   p->right = delete node(p->right, q->data);
                 }
          }.
```

```
if (bf(p)==2 && bf(p->left)==1) {
{
    p = LL rotation (p); }
else if (bf(p)==2 && bf(p->left)==-1)
{
    p = LR rotation (p); }
else if (bf(p)==-2 && bf(p->right)==1)
{
    p = RL rotation (p);
  }
else if ( bf(p)==-2 && bf(p->right)==0)
  {
    p = LL rotation (p);
  }
  return p;
}
```