Nikhil A.S
1BM18CS061
14/10/2020

ADA

AVL Tree

```
Insert (struct node *r, int data){
  if(r==null){
   struct node *temp;
   n->data = data;
   r= n->left = n->right = Null;
    return n;
  }
  else{
    if (data <=r->data)
      r->left = insert (r->left, data);
    else
      r->right = insert(r->right, data);
  }

  if(bf(r)==2 and bf(r->left)==1)
     r= lrotate(r);
  if(bf(r)==-2 and bf(r->right)==-1)
     r= rrotate(r);
  if(bf(r)==2 and bf(r->left)==-1)
     r= lrrotate(r)
  if(bf(r)==2 and bf(r->right)==1)
     r= rlrotate(r).

  return r;
}
```

Nikhil. A.S
1BM18CS061
14/10/2020

ADA

AVL Trees

```
Delete (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(p->right, data);
  else if (p->data >= data)
    p->left = delete(p->left, data);
  else {
    if (p->left != Null){
      q = inpre(p->left);
      p->data = q->data;
      p->left = delete(p->left, q->data);
    }
    else {
      q = insuc(p->right)
      p->data = q->data
      p->right = delete(p->right, q->data);
    }
  }
  if(bf(p)==2 && bf(p->left)==1){p=lrotate(p);}
  if(bf(p)==2 && bf(p->left)==-1){p=rrotate(p);}
  if(bf(p)==-2 && bf(p->right)==-1){p=rrotate(p);}
  if(bf(p)==-2 && bf(p->right)==1){p=rlrotate(p);}
  if(bf(p)==2 && bf(p->left)==0){p=lrotate(p);}
  if(bf(p)==-2 && bf(p->right)==0){p=lrotate(p);}
}
```