AVL Tree

```
#include <stdio.h>

struct Node
{
    struct Node *lchild;
    int data;
    int height;
    struct Node *rchild;
}*root=NULL;


int NodeHeight(struct Node *p)
{
    int hl,hr;
    hl=p && p->lchild?p->lchild->height:0;
    hr=p && p->rchild?p->rchild->height:0;

    return hl>hr?hl+1:hr+1;
}

int BalanceFactor(struct Node *p)
{
    int hl,hr;
    hl=p && p->lchild?p->lchild->height:0;
    hr=p && p->rchild?p->rchild->height:0;

    return hl-hr;

}

struct Node * LLRotation(struct Node *p)
{
    struct Node *pl=p->lchild;
    struct Node *plr=pl->rchild;

    pl->rchild=p;
    p->lchild=plr;
    p->height=NodeHeight(p);
    pl->height=NodeHeight(pl);

    if(root==p)
        root=pl;

    return pl;
}

struct Node * LRRotation(struct Node *p)
{
    struct Node *pl=p->lchild;
    struct Node *plr=pl->rchild;

    pl->rchild=plr->lchild;
    p->lchild=plr->rchild;

    plr->lchild=pl;
    plr->rchild=p;

    pl->height=NodeHeight(pl);
    p->height=NodeHeight(p);
    plr->height=NodeHeight(plr);

    if(root==p)
        root=plr;
    return plr;

}

struct Node * RRRotation(struct Node *p)
{
    return NULL;
}
struct Node * RLRotation(struct Node *p)
{
    return NULL;
}

struct Node *RInsert(struct Node *p,int key)
```

```
{
    struct Node *t=NULL;

    if(p==NULL)
    {
        t=(struct Node *)malloc(sizeof(struct Node));
        t->data=key;
        t->height=1;
        t->lchild=t->rchild=NULL;
        return t;
    }
    if(key < p->data)
        p->lchild=RInsert(p->lchild,key);
    else if(key > p->data)
        p->rchild=RInsert(p->rchild,key);

    p->height=NodeHeight(p);

    if(BalanceFactor(p)==2 && BalanceFactor(p-
>lchild)==1)
        return LLRotation(p);
    else if(BalanceFactor(p)==2 && BalanceFactor(p-
>lchild)==-1)
        return LRRotation(p);
    else if(BalanceFactor(p)==-2 && BalanceFactor(p-
>rchild)==-1)
        return RRRotation(p);
    else if(BalanceFactor(p)==-2 && BalanceFactor(p-
>rchild)==1)
        return RLRotation(p);
    return p;

}



int main()
{
    root=RInsert(root,50);
    RInsert(root,10);
    RInsert(root,20);


    return 0;
}
```