

Program-8: Design, develop a program in C to implement AVL tree operations.

```
#include<stdio.h>
typedef struct node
{
    int data;
    struct node *left,*right;
    int ht;
}NODE;

int height(NODE *T)
{
    if (T==NULL)
    {
        return -1;
    }
    else
    {
        int lheight = height(T->left);
        int rheight = height(T->right);
        if (lheight > rheight)
            return(lheight + 1);
        else
            return(rheight + 1);
    }
}

int BF(NODE *T)
{
    int lh,rh;
    if(T==NULL)
        return 0;
    if(T->left==NULL)
        lh=0;
    else
        lh=1+T->left->ht;
    if(T->right==NULL)
        rh=0;
    else
        rh=1+T->right->ht;
    return(lh-rh);
}

NODE * rotateright(NODE *x)
{
    NODE *y;
    y=x->left;
    x->left=y->right;
    y->right=x;
```

```
x->ht=height(x);
y->ht=height(y);
return y;
}
```

NODE * rotateleft(NODE *x)

```
{
    NODE *y;
    y=x->right;
    x->right=y->left;
    y->left=x;
    x->ht=height(x);
    y->ht=height(y);
    return y;
}
```

NODE* RR(NODE *T)

```
{
    T=rotateleft(T);
    return T;
}
```

NODE* LL(NODE *T)

```
{
    T=rotateright(T);
    return T;
}
```

NODE* LR(NODE *T)

```
{
    T->left=rotateleft(T->left);
    T=rotateright(T);
    return T;
}
```

NODE* RL(NODE *T)

```
{
    T->right=rotateright(T->right);
    T=rotateleft(T);
    return T;
}
```

NODE* insert(NODE *T, int x)

```
{
    if(T==NULL)
    {
        T=(NODE*)malloc(sizeof(NODE));
        T->data=x;
        T->left=T->right=NULL;
    }
}
```

```

    }
    else
    if(x > T->data)
    {
        T->right=insert(T->right,x);
        if(BF(T)==-2)
        if(x>T->right->data)
            T=RR(T);
        else
            T=RL(T);
    }
    else
    if(x<T->data)
    {
        T->left=insert(T->left, x);
        if(BF(T)==2)
        if(x < T->left->data)
            T=LL(T);
        else
            T=LR(T);
    }
    T->ht=height(T);
    return(T);
}

```

void inorder(NODE *T)

```

{
    if(T!=NULL)
    {
        inorder(T->left);
        printf("%d(Bf=%d)",T->data,BF(T));
        inorder(T->right);
    }
}

```

int main()

```

{
    NODE *root=NULL;
    int x,n,i,ch;
    while(1)
    {
        printf("\n 1.Create\t 2.Insert\t 3.Display\t 4.Exit\n");
        printf("\nEnter Your Choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\nEnter no. of elements:");
                    scanf("%d",&n);
                    printf("\nEnter tree data:");

```

```
        root=NULL;
        for(i=0;i<n;i++)
        {
            scanf("%d",&x);
            root=insert(root, x);
        }
        break;
case 2: printf("\nEnter a data:");
        scanf("%d",&x);
        root=insert(root,x);
        break;
case 3: printf("\nInorder sequence:\n");
        inorder(root);
        break;
case 4:exit(0);
}
}
return 0;
}
```