```
// Program to maintain an AVL tree
#include <conio.h>
#include<stdio.h>
#include <stdlib.h>

#define FALSE 0
#define TRUE 1

struct AVLNode
{
    int data;
    int balfact;
    struct AVLNode *left;
    struct AVLNode *right;
};

struct AVLNode *root=NULL;

struct AVLNode* insert(int data,int *h);
struct AVLNode* buildtree(struct AVLNode *root,int data,int *h);
void display(struct AVLNode *root);
struct AVLNode* deldata(struct AVLNode* root,int data,int *h);
struct AVLNode* del(struct AVLNode *node,struct AVLNode* root,int *h);
struct AVLNode* balright(struct AVLNode *root,int *h);
struct AVLNode* balleft(struct AVLNode* root,int *h);
void deltree(struct AVLNode *);

void main()
{
    int h,c,y,x;
    do
    {
        printf("\n\tOptions");
        printf("\n1.Create a new tree");
        printf("\n2.Insert Data");
        printf("\n3.Delete Data");
        printf("\n4.Display");
        printf("\n5.Exit");
        printf("\nEnter choice :- ");
        scanf("%d",&c);
        switch(c)
        {
            case 1:
                if(root != NULL)
                    deltree(root);
                root=NULL;
                printf("Enter element :- ");
                scanf("%d",&x);
                root=insert (x, &h ) ;
                break;
            case 2:
                do
                {
                    printf("Enter element :- ");
                    scanf("%d",&x);
                    root=insert (x, &h ) ;
                    printf("Do you want to continue?(YES=1,NO=0) :- ");
                    scanf("%d",&y);
                }while(y);
                break;
            case 3:
                printf("Enter element :- ");
                scanf("%d",&x);
                root=deldata (root,x, &h ) ;
                break;
            case 4:
                printf("\nAVL Tree\n");
                display(root);
                break;
            case 5:
                printf("\n");
                exit(0);
            default:
                printf("\nWrong Choice:Re-Enter");
                break;
        }
    }while(1);
}

// inserts an element in a binary tree by calling buildtree
struct AVLNode *insert(int data,int *h)
{
```

```
    root=buildtree(root,data,h);
    return root;
}

// inserts an element into tree
struct AVLNode *buildtree(struct AVLNode *root,int data,int *h)
{
    struct AVLNode *node1,*node2;
    if(root == NULL)
    {
        root=(struct AVLNode *)malloc(sizeof(struct AVLNode));
        root->data=data;
        root->left=NULL;
        root->right=NULL;
        root->balfact=0;
        *h=TRUE;
        return(root);
    }
    if(data < root->data)
    {
        root->left=buildtree(root->left,data,h);
        // If left subtree is higher
        if(*h)
        {
            switch(root->balfact)
            {
                case 1:
                    node1=root->left;
                    if(node1->balfact == 1)
                    {
                        printf("\nRight rotation\n");
                        root->left=node1->right;
                        node1->right=root;
                        root->balfact=0;
                        root=node1;
                    }
                    else
                    {
                        printf("\nDouble rotation,left then right\n");
                        node2=node1->right;
                        node1->right=node2->left;
                        node2->left=node1;
                        root->left=node2->right;
                        node2->right=root;
                        if(node2->balfact == 1)
                            root->balfact=-1;
                        else
                            root->balfact=0;
                        if(node2->balfact == -1)
                            node1->balfact=1;
                        else
                            node1->balfact=0;
                        root=node2;
                    }
                    root->balfact=0;
                    *h=FALSE;
                    break;
                case 0:
                    root->balfact=1;
                    break;
                case -1:
                    root->balfact=0;
                    *h=FALSE;
            }
        }
    }
    if(data > root->data)
    {
        root->right=buildtree(root->right,data,h);
        // If the right subtree is higher
        if(*h)
        {
            switch(root->balfact)
            {
                case 1:
                    root->balfact=0;
                    *h=FALSE;
                    break;
                case 0:
                    root->balfact=-1;
                    break;
                case -1:
```

```
                    node1=root->right;
                    if(node1->balfact == -1)
                    {
                        printf("\nLeft rotation\n");
                        root->right=node1->left;
                        node1->left=root;
                        root->balfact=0;
                        root=node1;
                    }
                    else
                    {
                        printf("\nDouble rotation, right then left\n");
                        node2=node1->left;
                        node1->left=node2->right;
                        node2->right=node1;
                        root->right=node2->left;
                        node2->left=root;
                        if(node2->balfact == -1)
                            root->balfact=1;
                        else
                            root->balfact=0;
                        if(node2->balfact == 1)
                            node1->balfact=-1;
                        else
                            node1->balfact=0;
                        root=node2;
                    }
                    root->balfact=0;
                    *h=FALSE;
                }
            }
        }
        return(root);
}

// prints data
void display(struct AVLNode *root)
{
    if(root != NULL)
    {
        display(root->left);
        printf("%d(%d)\t",root->data,root->balfact);
        display(root->right);
    }
}

// To delete an item from the tree
struct AVLNode *deldata(struct AVLNode *root,int data,int *h)
{
    struct AVLNode *node;
    if(root == NULL)
    {
        printf("\nNo such data\n");
        return(root);
    }
    else
    {
        if(data < root->data)
        {
            root->left=deldata(root->left,data,h);
            if(*h)
                root=balright(root,h);
        }
        else
        {
            if(data > root->data)
            {
                root->right=deldata(root->right,data,h);
                if(*h)
                    root=balleft(root,h);
            }
            else
            {
                node=root;
                if(node->right == NULL)
                {
                    root=node->left;
                    *h=TRUE;
                    free(node);
                }
                else
                {
```

```
                    if(node->left == NULL)
                    {
                        root=node->right;
                        *h=TRUE;
                        free(node);
                    }
                    else
                    {
                        node->right=del(node->right,node,h);
                        if(*h)
                            root=balleft(root,h);
                    }
                }
            }
        }
    }
    return(root);
}

struct AVLNode *del(struct AVLNode *succ,struct AVLNode *node,int *h)
{
    struct AVLNode *temp=succ;
    if(succ->left != NULL)
    {
        succ->left=del(succ->left,node,h);
        if(*h)
            succ=balright(succ,h);
    }
    else
    {
        temp=succ;
        node->data=succ->data;
        succ=succ->right;
        free(temp);
        *h=TRUE;
    }
    return(succ);
}

// To balance the tree, if right sub-tree is higher
struct AVLNode *balright(struct AVLNode *root,int *h)
{
    struct AVLNode *temp1,*temp2;
    switch(root->balfact)
    {
        case 1:
            root->balfact=0;
            break;
        case 0:
            root->balfact=-1;
            *h=FALSE;
            break;
        case -1:
            temp1=root->right;
            if(temp1->balfact <= 0)
            {
                printf("\nLeft rotation\n");
                root->right=temp1->left;
                temp1->left=root;
                if(temp1->balfact == 0)
                {
                    root->balfact=-1;
                    temp1->balfact=1;
                    *h=FALSE;
                }
                else
                {
                    root->balfact=temp1->balfact=0;
                }
                root=temp1;
            }
            else
            {
                printf("\nDouble rotation, right then left\n");
                temp2=temp1->left;
                temp1->left=temp2->right;
                temp2->right=temp1;
                root->right=temp2->left;
                temp2->left=root;
                if(temp2->balfact==-1)
                    root->balfact=1;
                else
```

```
                    root->balfact=0;
                if(temp2->balfact==1)
                    temp1->balfact=-1;
                else
                    temp1->balfact=0;
                root=temp2;
                temp2->balfact=0;
            }
    }
    return(root);
}

// To balance the tree, if left sub-tree is higher
struct AVLNode *balleft(struct AVLNode *root,int *h)
{
    struct AVLNode *temp1,*temp2;
    switch(root->balfact)
    {
        case -1:
            root->balfact=0;
            break;
        case 0:
            root->balfact=1;
            *h=FALSE;
            break;
        case 1:
            temp1=root->left;
            if(temp1->balfact >= 0)
            {
                printf("\nRight rotation\n");
                root->left=temp1->right;
                temp1->right=root;
                if(temp1->balfact == 0)
                {
                    root->balfact=1;
                    temp1->balfact=-1;
                    *h=FALSE;
                }
                else
                {
                    root->balfact=temp1->balfact=0;
                }
                root=temp1;
            }
            else
            {
                printf("\nDouble rotation, left then right\n");
                temp2=temp1->right;
                temp1->right=temp2->left;
                temp2->left=temp1;
                root->left=temp2->right;
                temp2->right=root;
                if(temp2->balfact == 1)
                    root->balfact=-1;
                else
                    root->balfact=0;
                if(temp2->balfact == -1)
                    temp1->balfact=1;
                else
                    temp1->balfact=0;
                root=temp2;
                temp2->balfact=0;
            }
    }
    return (root);
}

void deltree(struct AVLNode *root)
{
    if(root != NULL)
    {
        deltree(root->left);
        deltree(root->right);
    }
    free(root);
}
```