```
Group D1
Title:-  Construct Height Balance Tree (AVL)

#include<iostream>
#include<string.h>
using namespace std;
struct AVL_Node{
    string name;
    string mean;
    int height;
    AVL_Node *left;
    AVL_Node *right;
};
class AVL_Tree{
    AVL_Node *root;
    public:
        AVL_Tree(){
            root=NULL;
        }
        AVL_Node *Create(AVL_Node *x){
            root=Insert(root,x);
            return root;
        }
        AVL_Node *Insert(AVL_Node *,AVL_Node * );
        AVL_Node *LL(AVL_Node *);
        AVL_Node *RR(AVL_Node *);
        int bf(AVL_Node *);
        int height(AVL_Node *);
        AVL_Node *leftrotate(AVL_Node *);
        AVL_Node *rightrotate(AVL_Node *);
};
AVL_Node* AVL_Tree::leftrotate(AVL_Node *temp){
    AVL_Node *y = temp->right;
  AVL_Node *T2 = y->left;

  y->left = temp;
  temp->right = T2;

  temp->height = max(height(temp->left),
    height(temp->right)) +1;
  y->height = max(height(y->left),
    height(y->right)) +1;
}
AVL_Node* AVL_Tree::rightrotate(AVL_Node *temp){
```

```
    string key;
    key=temp->name;
    int balance=bf(temp);
    if(balance>1 && key < temp->left->name){
        return rightrotate(temp);
    }
}
AVL_Node* AVL_Tree::Insert(AVL_Node *troot,AVL_Node *key){
    int i=0;
    if(troot==NULL){
        troot= new AVL_Node;
        troot->name=key->name;
        troot->mean=key->mean;
        troot->left=NULL;
        troot->right=NULL;
        troot->height=1;//initially Will be 1
        cout<<"\nInserted Data is "<<troot->name<<"|"<<troot->mean;
    }
    else if(key->name.at(i)>troot->name.at(i)){
        troot->right=Insert(troot->right,key);
        if(bf(troot)==2 || bf(troot)==-2){
            if(key->name.at(i)>troot->right->name.at(i)){
                troot->right=RR(troot);
                cout<<"\nInserted Data is "<<troot->name<<"|"<<troot->mean;
            }
        }
    }
    else if (key->name.at(i)<troot->name.at(i)){
        troot->left=Insert(troot->left,key);
        if(bf(troot)==2 || bf(troot)==-2){
            if(key->name.at(i)<troot->left->name.at(i)){
                troot->left=LL(troot);
                cout<<"\nInserted Data is "<<troot->name<<"|"<<troot->mean;
            }
        }
    }
    return (troot);
}
int AVL_Tree::bf(AVL_Node *temp){
    if (temp == NULL)
        return 0;
    else
        return (height(temp->left) - height(temp->right));
}
int AVL_Tree::height(AVL_Node *htemp){
```

```
    int lh,rh;
    if(htemp==NULL)
        return 0;
    if(htemp->left==NULL){
        lh=0;
    }else{
        lh=1+htemp->left->height;
    }
    if(htemp->right==NULL){
        rh=0;
    }else{
        rh=1+htemp->right->height;
    }
    if(lh>rh)
        return (lh);
    return (rh);
}
AVL_Node* AVL_Tree::LL(AVL_Node *root){
    root=rightrotate(root);
    return root;
}
AVL_Node* AVL_Tree::RR(AVL_Node *root){
    root=leftrotate(root);
    return root;
}
int main(){
    AVL_Tree t;
    char ans;
    AVL_Node *root;
    int ch=1;
    do{
        if(ch==1){
            root= new AVL_Node;
            cout<<"\nEnter The name";
            cin>>root->name;
            cout<<"\nEnter the Meaning";
            cin>>root->mean;
            cout<<t.Create(root);
            ch++;
        }
        else
        {
                AVL_Node *temp= new AVL_Node;
                cout<<"\nEnter The name";
                cin>>temp->name;
```

```
                cout<<"\nEnter the Meaning";
                cin>>temp->mean;
                cout<<t.Create(temp);
        }
        cout<<"Continue??";
        cin>>ans;
    }while(ans=='y' ||ans=='Y');
    return 0;
}
```

Output:-

Enter The name Ulkesh

Enter the Meaning Wani

Inserted Data is Ulkesh|Wani|0x1156dd0|Continue??y

Enter The name Vipul

Enter the Meaning Chandankar

Inserted Data is Vipul|Chandankar|0x1156dd0|Continue??y

Enter The name Sujit

Enter the Meaning Gosavi

Inserted Data is Sujit|Gosavi|0x1156dd0|Continue??y

Enter The name Omkr

Enter the Meaning Jadhav

Inserted Data is Omkr|Jadhav