```
#include<iostream>
#include<string.h>
using namespace std;

class dict
{
        char word[20],mean[50];
        dict *left,*right;
        int ht;
public:

        dict* create(dict *root);
        dict* insert(dict *root,char word[],char mean[]);
        void display(dict *);
        int height(dict *);
        dict* rotateright(dict *);
        dict* rotateleft(dict *);
      int BF(dict *);
      dict* deletion(dict *,char *);
        dict* RR(dict*);
        dict* LL(dict*);
        dict* RL(dict*);
        dict* LR(dict*);
```

```
};

dict* dict::create(dict *root)
{
        int n,i;
        char w[20],m[50];
        cout<<"\n Enter total number of words:";
        cin>>n;
        for(i=0;i<n;i++)
        {
                cout<<"\n Enter word "<<i+1<<" : ";
                cin>>w;
                cout<<"\n Enter meaning : ";
                cin>>m;
                root=insert(root,w,m);

        }
        return root;
}

dict* dict::insert(dict *root,char w[],char m[])
{
   if(root==NULL)
   {
```

```
        root=new dict;
        strcpy(root->word,w);
        strcpy(root->mean,m);
        root->left=NULL;
        root->right=NULL;
        return root;
}
else
{
if(strcmp(w,root->word)>0)
{
        root->right=insert(root->right,w,m);
        if(BF(root)==-2)
        {
                if(strcmp(w,root->word)>0)
                        root=RR(root);
                else
                        root=RL(root);
        }

}
else
{
        if(strcmp(w,root->word)<0)
```

```
        {
           root->left=insert(root->left,w,m);
                if(BF(root)==2)
                {
                        if(strcmp(w,root->word)<0)
                                root=LL(root);
                        else
                                root=LR(root);
                }
        }

   }
 }
root->ht=height(root);
return root;
}

void dict::display(dict* root)
{
        if(root!=NULL)
        {
                display(root->left);
                cout<<"\n Node is:"<<root->word<<"-"<<root->mean;
                display(root->right);
```

```
        }
}


int dict::height(dict *root)
{
        int lh,rh;
        if(root==NULL)
                return 0;
        if(root->left==NULL)
                lh=0;
        else
                lh=1+root->left->ht;
        if(root->right==NULL)
                rh=0;
        else
                rh=1+root->right->ht;

        if(lh>rh)
        {
                return(lh);
        }
        else
        {
```

```
                return(rh);
        }

}

dict* dict::rotateright(dict *x)
{
        dict *y;
        y=x->left;
        x->left=y->right;
        y->right=x;
        x->ht=height(x);
        y->ht=height(y);
        return(y);
}

dict* dict::rotateleft(dict *x)
{
        dict *y;
        y=x->right;
        x->right=y->left;
        y->left=x;
        x->ht=height(x);
        y->ht=height(y);
```

```
        return(y);
}
int dict::BF(dict *root)
{
        int lh,rh;
        if(root==NULL)
                return(0);
        if(root->left==NULL)
                lh=0;
        else
                lh=1+root->left->ht;
        if(root->right==NULL)
          rh=0;
        else
                rh=1+root->right->ht;
        int z=lh-rh;
        return(z);
}

dict* dict:: deletion(dict *T,char *w)
{
        dict *p;
        if(T==NULL)
        {
```

```
        cout<<"\n Word not found!";
        return T;
}
else
if(strcmp(w,T->word)>0)
{
   T->right=deletion(T->right,w);
   if(BF(T)==2)
   {
        if(BF(T->left)>=0)
                T=LL(T);
        else
                T=LR(T);
   }

}
else
if(strcmp(w,T->word)<0)
        {
                T->left=deletion(T->left,w);
                if(BF(T)==2)
                        {
                             if(BF(T->right)<=0)
                                     T=RR(T);
```

```
                                    else
                                            T=RL(T);
                                }
                        }
                else
                {
        if(T->right!=NULL)
        {
                p=T->right;
                while(p->left!=NULL)
                        p=p->left;
                strcpy(T->word,p->word);
                strcpy(T->mean,p->mean);
                T->right=deletion(T->right,p->word);
                if(BF(T)==2)
                {
                        if(BF(T->left)>=0)
                                T=LL(T);
                        else
                                T=LR(T);
                }
        }
        else
                return(T->left);
```

```
        }
        T->ht=height(T);
        return(T);
}


dict* dict::RR(dict *T)
{
        T=rotateleft(T);
        return(T);
}
dict* dict::LL(dict *T)
{
        T=rotateright(T);
        return(T);
}
dict* dict::LR(dict *T)
{
        T->left=rotateleft(T->left);
        T=rotateright(T);
        return(T);
}
dict* dict::RL(dict *T)
{
```

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

int main()
{
        int ch;
        dict d,*root;
        root=NULL;
        char w[20],m[50];
        cout<<"\n ***Dictionary : codyapa***";

        do{
                cout<<"\n\n MENU:";
                cout<<"\n1.Create \n2.Insert \n3.Delete\n4.Display \n5.Exit";
                cout<<"\n Enter your choice:";
                cin>>ch;

                switch(ch)
                {
                case 1: root=d.create(root);
                break;
```

```
        case 2: cout<<"\n Enter word:";
                cin>>w;
                cout<<"\n Enter meaning";
                cin>>m;
                root=d.insert(root,w,m);
                break;

        case 3: cout<<"\nEnter word to delete";
                cin>>w;
                root=d.deletion(root,w);
                break;


        case 4: d.display(root);
        break;


        case 5: break;
        default:cout<<"\n Invalid choice!";

        }
}
while(ch!=5);
return 0;
```

```
}
```