```
#include<iostream>
#include<string.h>
using namespace std;
typedef struct node
{
char k[20];
char m[20];
class node *lnode;
class node *rnode;
}
node;
class dictionary
{
public:
node *root;
void create();
void display(node *);
void addrecord(node *root,node *temp);
int search(node *,char []);
int update(node *,char []);
node *delnode(node *,char []);
node *min(node *);
};
void dictionary ::create()
{
class node *temp;
int ch;
do
{
temp= new node;
cout<<"\n Enter the Keyword: ";
cin>>temp->k;
cout<<"\nEnter Meaning of the Keyword : ";
cin>>temp->m;
temp->lnode=NULL;
temp->rnode =NULL;
if(root== NULL)
{
root=temp;
}
else
{
addrecord(root, temp);
}
cout<<"\nDo u want to add more record ? (y-1/n-0):";
cin>>ch;
}
while(ch==1 );
}
void dictionary :: addrecord(node *root,node *temp)
{
if(strcmp (temp->k, root->k) <0)
{
if(root->lnode==NULL)
root->lnode=temp;
else
addrecord(root->lnode,temp);
}
else
{
  if(root->rnode== NULL)
  root->rnode=temp;
else
addrecord(root->rnode,temp);
}
}
void dictionary:: display(node *root)
{
if( root != NULL)
{
display(root->lnode);
cout<<"\n Key Word:"<<root->k;
cout<<"\t Meaning :"<<root->m;
display(root->rnode);
}
}
int dictionary :: search(node *root,char k[20])
{
int c=0;
while(root != NULL)
{
c++;
if(strcmp(k,root->k)==0)
{
    cout<<"\n No of Comparisons:"<<c;
return 1;
}
if(strcmp (k, root->k) <0)
root=root->lnode;
if(strcmp (k, root->k) >0)
root=root->rnode;
}
return -1;
}
int dictionary :: update(node *root,char k[20])
{
while(root !=NULL)
{

if(strcmp (k, root->k)==0)
{
    cout<<"\nEnter New Meaning of Keyword "<<root->k;
    cin>>root->m;
    return 1;
}
if(strcmp (k, root->k) <0)
root=root->lnode;
if(strcmp (k, root->k) >0)
root=root->rnode;
}
```

```
return -1;
}
node *dictionary :: delnode(node *root,char k[20])
{
node *temp;
if(root==NULL)
{
cout<<"\nElement Not Found";
return root;
}
if (strcmp(k,root->k) <0)
{
root->lnode=delnode( root->lnode, k);
return root;
if (strcmp(k,root->k) >0)
root->rnode = delnode(root->rnode,k);
return root;
}
if (root->rnode==NULL && root->lnode==NULL)
{
temp=root;
delete temp;
return NULL;
}
if(root->rnode==NULL)
{
temp=root;
root=root->lnode;
delete temp;
return root;
}
else if(root->lnode==NULL)
{
temp=root;
root=root->rnode;
delete temp;
return root;
}
temp =min(root->rnode);
strcpy(root->k,temp->k);
root->rnode=delnode(root->rnode, temp->k);
return root;
}
node *dictionary::min(node *q)
{
while(q->lnode != NULL)
{
q=q->lnode;
}
return q;
}
int main()
{
int ch;
dictionary d;
```

```
d.root=NULL;
do
{
cout<<"\n1 :Create\n2: Display\n3 : Search \n4 : Update \n5:Delete.";
cout<<"\nSelect your choice:";
cin>> ch;
switch(ch)
{
case 1: d.create();
break;
case 2: if(d.root==NULL)
{
cout<<"\n Dictionary is empty..";
}
else
{
d.display(d.root);
}
break;
case 3: if(d.root== NULL)
{
cout<"\nDictionary is Empty. First add again ";
}
else
{
cout<<"\nEnter Keyword which u want to search:";
char k[20];
cin>>k;
if( d.search(d.root,k) )
cout<<"\nKeyword Found";
else
cout<<"\nKeyword Not Found";
}
break;
case 4:
if(d.root==NULL)
{
    cout<<"\nDictionary is Empty. First add keywords then try again ";
}
else
{

cout<<"\nEnter Keyword which meaning want to update:";
char k[20];
cin>>k;
if(d.update(d.root,k)== 1)
cout<<"\Meaning Found ";
else
cout<<"\nMeaning Not Found";
}
break;
case 5:
if(d.root==NULL)
{
cout<<"\nDictionary is Empty. First add keywords then try again";
}
else
{
cout<<"\nEnter Keyword whichu want to delete: ";
char k[20];
cin>>k;
if(d.root==NULL)
{
cout<<"\nNo any Keyword";
}
else
{
d.root = d.delnode( d.root,k);
}
}
}
}
while(ch<=5);
return 0;
}
```