#include <iostream>

#include <string>

#include <cstring>

#include <cstdlib>

using namespace std;

int op;

int cnt=0;

class node

{

public:

node \*left;

char word[50],mean[50];

node \*right;

};

class BT

{

public:

node \*root;

BT()

{

root=NULL;

}

void create();

node\* insert(node \*,node \*);

void inorder(node \*);

void preorder(node \*);

void postorder(node \*);

void search(node \*, char []);

void modify(node \*, char []);

node \*dlt(node \* ,char []);

node \*FindMin(node \* );

};

void BT::create()

{

int op;

node \*temp;

do

{

temp=new node;

cout<<"Enter A word ";

cin>>temp->word;

cout<<"Enter A Meaning : ";

cin>>temp->mean;

temp->left=temp->right=NULL;

if(root==NULL)

{

root=temp;

}

else

{

root=insert(root,temp);

}

cout<<"Want to insert again : ";

cin>>op;

}while(op==1);

}

node\* BT::insert(node \*root,node \*temp)

{

if(strcmp (temp->word, root->word) < 0 )

{

if(root->left == NULL)

root->left = temp;

else

insert(root->left,temp);

}

else

{ if(root->right == NULL)

root->right = temp;

else

insert(root->right,temp);

}

return root;

}

void BT::inorder(node \*temp)

{

if(temp!=NULL)

{

inorder(temp->left);

cout<<temp->word<<" -> "<<temp->mean<<" , ";

inorder(temp->right);

}

}

void BT::preorder(node \*temp)

{

if(temp!=NULL)

{

cout<<temp->word<<"-> "<<temp->mean<<" , ";

preorder(temp->left);

preorder(temp->right);

}

}

void BT::postorder(node \*temp)

{

if(temp!=NULL)

{

postorder(temp->left);

postorder(temp->right);

cout<<temp->word<<"-> "<<temp->mean<<" , ";

}

}

void BT::search(node \*temp , char src[])

{

if(temp != NULL)

{

if((strcmp(temp->word , src)) == 0)

{

cout<<"\n Word Found ";

cout<<"\n Word : "<<temp->word;

cout<<"\n meaning : "<<temp->mean;

cnt++;

}

else

{

if((strcmp( src, temp->word )) > 0)

{

search(temp->right , src);

cnt++;

}

else

{

search(temp->left , src);

cnt++;

}

}

}

else

cout<<"\n Word Not Found ";

cout<<"\n Total no of Comparisions to search an element is: "<<cnt;

}

void BT::modify(node \*temp , char src[])

{

if(temp != NULL)

{

if((strcmp(temp->word , src)) == 0)

{

cout<<"\n Word Found ";

cout<<"\n Enter New Meaning Of Word "<<temp->word;

cin>>temp->mean;

}

else

{

if((strcmp(temp->word , src)) < 0)

{

modify(temp->right , src);

}

else if((strcmp(temp->word , src)) > 0)

{

modify(temp->left , src);

}

}

}

else

cout<<"\n Word Not Found ";

}

node\* BT::dlt(node \*root , char src[])

{

if(root != NULL)

{

if((strcmp(root->word , src)) > 0)

{

root->left = dlt(root->left , src);

}

else if((strcmp(root->word , src)) < 0)

{

root->right = dlt(root->right , src);

}

else

{

if(root->left == NULL && root->right == NULL)

{

delete(root);

root = NULL;

}

else if(root->left == NULL && root->right!=NULL)

{

node \*temp = root;

root = root->right;

strcpy(root->word , temp->word);

strcpy(root->mean , temp->mean);

temp->right=NULL;

delete(root);

}

else if(root->right == NULL)

{

node \*temp = root;

root = root->left;

strcpy(root->word , temp->word);

strcpy(root->mean , temp->mean);

temp->left=NULL;

delete(root);

}

else

{

node \*temp = FindMin(root->right);

strcpy(root->word , temp->word);

strcpy(root->mean , temp->mean);

root->right = dlt(root->right , temp->word);

}

}

}

return root;

}

node\* BT:: FindMin(node\* root)

{

while(root->left != NULL) root = root->left;

return root;

}

int main()

{

BT b;

int op;

char src[100];

while(1)

{

cout<<"\n ";

cout<<"\n 1. Insert Binary Search Tree ";

cout<<"\n 2. Display Inorder,preorder and postorder ";

cout<<"\n 3. Search The Word ";

cout<<"\n 4. Modify The Meaning Of Word ";

cout<<"\n 5. Delete Word From Dictionary ";

cout<<"\n 6.Exit";

cout<<"\n Enter your choice:";

cin>>op;

switch(op)

{

case 1:

b.create();

break;

case 2:

cout<<"\n Inorder : ";

b.inorder(b.root);

cout<<"\n Preorder : ";

b.preorder(b.root);

cout<<"\n Postorder : ";

b.postorder(b.root);

break;

case 3:

cnt=0;

cout<<"\n Enter The Word Want To Search : ";

cin>>src;

b.search(b.root , src);

break;

case 4:

cout<<"\n Enter The Word Want To Modify ";

cin>>src;

b.modify(b.root , src);

break;

case 5:

cout<<"\n Enter The Word Want To Delete ";

cin>>src;

b.dlt(b.root , src);

break;

case 6:

exit(0);

break;

default :

cout<<"\n Invalid Option ";

break;

}

}

}





