#include <iostream>

#include <stdlib.h>

#include<string.h>

using namespace std;

struct node

{

char kwd[20];

char meaning[40];

node \*left,\*right;

};

class tree

{

public:

node \*root,\*temp;

tree()

{

root=NULL;

}

void create();

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

void inorder(node \*);

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

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

int comparisons(node \*);

};

void tree::create()

{

root=NULL;

char ch;

do{

temp=new node;

cout<<" enter keyword"<<endl;

cin>>temp->kwd;

cout<<" enter meaning"<<endl;

fflush(stdin);

gets(temp->meaning);

temp->left=NULL;

temp->right=NULL;

if(root==NULL)

root=temp;

else

{

insert(root,temp);

}

cout<<"do u want to continue"<<endl;

cin>>ch;

}

while(ch=='y');

}

void tree::insert(node \*trav,node \*temp)

{ char ch1;

if(strcmp(temp->kwd,trav->kwd)<0)

{

if(trav->left==NULL)

trav->left=temp;

else

insert(trav->left,temp);

}

else if(strcmp(temp->kwd,trav->kwd)>0)

{

if(trav->right==NULL)

trav->right=temp;

else

insert(trav->right,temp);

}

}

int tree::comparisons(node \*T)

{

if(T==NULL)

return(0);

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

return(0);

return(max(comparisons(T->left),comparisons(T->right))+1);

}

node \* tree:: Delete (node \* root, char x[])

{

if (root == NULL)

{

cout << "Node not found ";

return NULL;

}

if (strcmp(x,root->kwd)<0 )

{

root->left = Delete (root->left, x);

}

else if (strcmp(x,root->kwd)>0)

{

root->right = Delete (root->right, x);

}

else

{

if (root->left == NULL)

{

node \*temp = root->right;

free (root);

return temp;

}

else if (root->right == NULL)

{

node \*temp = root->left;

free (root);

return temp;

}

else

{

node \*temp = root->right;

while (temp->left != NULL)

temp = temp->left;

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

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

root->right = Delete (root->right, temp->kwd);

}

}

return root;

}

void tree::inorder(node \*root)

{

if(root!=NULL)

{

inorder(root->left);

cout<<" "<<root->kwd;

cout<<"("<<root->meaning<<")";

inorder(root->right);

}

}

node \* tree::search(node \* temp,char x[])

{

//int flag=0;

while(temp!=NULL)

{

if(strcmp(x,temp->kwd)<0)

{

temp=temp->left;

}

else if(strcmp(x,temp->kwd)>0)

{

temp=temp->right;

}

else if(strcmp(x,temp->kwd)==0)

{

break;

}

}

return temp;

}

int main()

{

node \*temp;

tree t1;

char key[20];

int xx,op,x,c;

do

{

cout<<"\n\n1.Create\n2.Insert";

cout<<"\n3.Update \n4.inorder display\n5.delete\n6.Search\n7.Max comparisons \n8.Exit";

cout <<"\nEnter Your Choice :"<<endl;

cin>>op;

switch(op)

{

case 1:

t1.create();

break;

case 2:

temp=new node;

temp->left=NULL;

temp->right=NULL;

cout<<"\nenter a new keyword you want to add\n";

cin>>temp->kwd;

cout<<"\nenter meaning of keyword you want to add\n";

fflush(stdin);

gets(temp->meaning);

t1.insert(t1.root,temp);

break;

case 3:

cout<<"\nenter a keyword which you want to update\n";

cin>>key;

temp=t1.search(t1.root,key);

if(temp==NULL)

cout<<"Sorry No such keyword is found in dictionary\n";

else

{

cout<<"\nenter new meaning of keyword you want to update\n";

fflush(stdin);

gets(temp->meaning);

}

break;

case 4:

t1.inorder(t1.root);

break;

case 5:

int x;

cout<<"\nEnter a keyword to delete\n";

cin>>key;

t1.Delete(t1.root,key);

break;

case 6:

cout<<"enter keyword to search";

cin>>key;

temp=t1.search(t1.root,key);

if(temp==NULL)

cout<<"Sorry No such keyword is found in dictionary\n";

else

{

cout<<"\nData Found\n";

}

break;

case 7:

c=t1.comparisons(t1.root);

cout<<"\n Maximum number of comparisons to search any node in this tree is"<<c+1;

case 8:

exit(0);

}

}

while(op!=7);

return 0;

}