//DICTIONARY FORMED FROM THE LIST OF WORDS IN TEXT FILE

//WITH THEIR MEANINGS SEPERATED BY A SPACE. SERACH WORD USING

//BINARY SEARCH TREE

#include<iostream>

#include<fstream>

#include<stdlib.h>

#include<string.h>

#include<cstdio>

#include<conio.h>

#include<windows.h>

using namespace std;

FILE \*fp;

void gotoxy(int x,int y)

{

COORD coord;

coord.X=x;

coord.Y=y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),coord);

}

void spiral(int a, int b)

{

system("cls");

int i,x1=0,x2=a,y1=0,y2=b;

while(x1<x2 && y1<y2)

{

for(i=x1;i<x2;++i)

{

gotoxy(y1\*1,i\*1);

cout<<".";

Sleep(1);

}

y1++;

for(i=y1;i<y2;++i)

{

gotoxy(i\*1,(x2-1)\*1);

cout<<".";

Sleep(1);

}

x2--;

if(y1<y2)

for(i=x2-1;i>=x1;--i)

{

gotoxy((y2-1)\*1,i\*1);

cout<<".";

Sleep(1);

}

y2--;

if(x1<x2)

for(i=y2-1;i>=y1;--i)

{

gotoxy(i\*1,x1\*1);

cout<<".";

Sleep(2);

}

x1++;

}

system("cls");

}

class node

{

public:

string a;

string b;

node \*right, \*left;

node(){a='\0';}

}\*root;

void make\_tree(string x,string y)

{

node \*temp;

temp=new node;

if(temp==NULL)

cout<<"Not enough memory";

else

{

temp->a=x;

temp->b=y;

temp->left=NULL;

temp->right=NULL;

if(root==NULL)

{

cout<<"\nThis is the root node\n";

root = temp;

}

else

{

node \*p,\*q;

p=root;

while(p!=NULL && p->a!=x)

{

q=p;

if(p->a > x)

{

p=p->left;

}

else if(p->a < x)

{

p=p->right;

}

}

if(p!=NULL)

cout<<"\nELEMENT "<<p->a<<" ALREADY EXISTS\n";

else if(x < q->a)

q->left=temp;

else

q->right=temp;

}

}

}

int check=0;

void show\_word(node \*n, string x)

{

if(n==NULL)

return;

if(n->a==x)

{

cout<<n->a<<": "<<n->b<<endl;

check = 1;

return ;

}

else if(x < n->a)

show\_word(n->left,x);

else

show\_word(n->right,x);

}

node\* delete\_tree(node\* r)

{

if(r!=NULL)

{

delete\_tree(r->left);

delete\_tree(r->right);

delete r;

r=NULL;

return r;

}

}

void format(char s[],char w[], char m[])

{

int i,j;

for(i=0;s[i]!=' ';i++)

w[i]=s[i];

w[i++]=NULL;

for(j=0;s[i]!='\0';i++,j++)

{

m[j]=s[i];

}

m[j]=NULL;

}

void check\_word(char o[])

{

for(int i=0;i<strlen(o);++i)

{

if(o[i]==' ')

cout<<"invalid";

}

}

void add\_new\_words()

{

char w[20],m[100];

cout<<"\nEnter the word you want to add to the Dictionary\n";

cin>>w;

check\_word(w);

if(check==0)

{

ofstream fout;

fout.open("DICTIONARY.txt",ios::out | ios::app);

cout<<"\nEnter the meaning of the word\n";

char f;

cin.get(f);

cin.getline(m,100);

char n[10];

cout<<"Enter whether its a verb(v) or noun(n) or adjective(adj)";

cin>>n;

char s[190];int i;

for(i=0;i<strlen(n);++i)

s[i]=n[i];

for(i;i<strlen(m);++i)

s[i]=m[i];

for(i;i<190;++i)

s[i]=NULL;

make\_tree(w,s);

fout<<endl<<w<<" "<<n<<". "<<m<<endl;

cout<<"\nThe word is successfully added to the dictionary";

return;

}

else

cout<<"This word is already present in the dictionary";

show\_word(root,w);

check = 0;

}

void head()

{

char \*a,s[200],word[20],meaning[180];

int choice;

fp=fopen("DICTIONARY.txt","r");

while(!feof(fp))

{

a=fgets(s,200,fp);

format(s,word,meaning);

make\_tree(word,meaning);

}

here:

system("cls");

cout<<"\t\tWELCOME TO iDictionary ^\_^\n";

cout<<"\n1.Look Up word\n2.Advanced Options\n3.EXIT APPLICAION\n";

cin>>choice;

if(!cin.fail())

switch(choice)

{

system("cls");

case 1: cout<<"Enter the word you want to search\n";

cin>>word;

for(int i=0;i<strlen(word);i++)

{

word[i]=tolower(word[i]);

}

check\_word(word);

getch();

check=0;

system("cls");

show\_word(root,word);

if(check==0)

cout<<"Sorry this word is not present in the dictionary";

getch();

break;

case 2: cout<<"1.Add a new word in the dictionary\n";

add\_new\_words();

getch();

break;

case 3: exit(1);

default: cout<<"ERROR";

goto here;

}

else

{

cout<<"ERROR ....EXITING";

return ;

}

goto here;

}

int main()

{

// spiral(40,50);

head();

}