/\*

This programs uses trie data structure to find out

whether a particular word is present in the file

"robo.txt"

If not present user get a chance to see the words predicted

auto prediction

\*/

#include<iostream>

#include<fstream>

#include<string.h>

using namespace std;

#define ALPHA 26

#define INDEX(c) ((int)c - (int)'a')

int search\_node(char w[]);

struct trie\_node

{

int value;

trie\_node \*children[ALPHA];

};

trie\_node \*create\_node()

{

trie\_node \*p;

p=new trie\_node;

if(p)

{

p->value=0;

for(int i=0;i<ALPHA;++i)

{

p->children[i]=NULL;

}

}

return p;

}

trie\_node \*root = create\_node();

int count=0;

void insert\_node(char word[])

{

int length=strlen(word);

int index;

trie\_node \*a;

count++;

a=root;

for(int i=0 ; i<length ; ++i)

{

index = INDEX(word[i]);

if(!a->children[index])

{

a->children[index]=create\_node();

}

a=a->children[index];

}

a->value=count;

}

void copystr(char b[], char a[])

{

int i;

for(i=1;i<strlen(a);++i)

{

b[i-1]=a[i];

}

b[i-1]='\0';

}

static int flag=0;

int fail\_search(trie\_node \*x,char word[])

{

if(strlen(word)<2)

return 0;

// cout<<"\n\n fail : word: "<<word;

char w[20];

w[0]=NULL;

strcpy(w,word);

trie\_node \*t;

t=x;

int i;

char p[][2]={"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"};

for(i=0; i<ALPHA; ++i)

{

if(t->children[i]!=NULL)

{

flag=1;

strcat(w,p[i]);

if(search\_node(w))

{

cout<<endl<<w;

break;

return 1;

}

fail\_search(t->children[i],w);

w[0]='\0';

strcpy(w,word);

}

}

}

int search\_node(char word[])

{

int length=strlen(word);

int index,t=0;

char w[20],z;

w[0]='\0';

trie\_node \*a;

char p[][2]={"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r","s","t","u","v","w","x","y","z"};

a=root;

for(int i=0 ; i<length ; i++)

{

t++;

index=INDEX(word[i]);

if(!a->children[index])

{

if(t>=3)

{

flag=1;

cout<<"\nThe given word was not present\n";

cout<<"Do you want to see the words suggested by our auto-prediction.\nY for YES\nN for NO\n\n";

cin.get(z);

if(z=='y' || z=='Y')

{

int h = fail\_search(a,w);

if(h)

cout<<"\n\nThe above words are suggested by auto prediction\n\n";

return 2;

}

}

else

{

return 0;

}

}

a=a->children[index];

if(a==NULL)

{

return 0;

}

strcat(w,p[index]);

//cout<<w<<endl;

//cout<<endl<<"a value is "<<a->value;

}

return(0!=a && a->value);

}

void fail\_handle(char word[])

{

trie\_node \*tp;

tp=root;

int k=0,i;

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

{

k=int(word[i])-97;

if(tp->children[k]==NULL)

return ;

else

tp=tp->children[k];

}

fail\_search(tp,word);

return ;

}

void control(char p[])

{

char s[20],z;

int result;

ifstream file;

file.open("OfficialWordlist.txt",ios::in);

while(file)

{

file.getline(s,20);

insert\_node(s);

}

int h = search\_node(p);

if(h==1)

cout<<"\nThe word is present.\n\n";

else if(!h && h!=2)//for goo

{

cout<<"\nThe given word was not present\n";

cout<<"Do you want to see the words suggested by our auto-prediction.\nY for YES\nN for NO\n\n";

cin.get(z);

if(z=='y' || z=='Y')

fail\_handle(p);

cout<<"\n\nThe above words are suggested by auto prediction\n\n";

}

}

int main()

{

char p[20];

cin.getline(p,20);

control(p);

return 0;

}