#include<stdio.h>

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

#define N 26

struct node{

struct node\* child[N];

int end;

char meaning[100];

};

typedef struct node node;

node\* load()

{

int i;

node\* root=NULL;

if ((root = (node\*)malloc (sizeof(node))) == NULL)

{

printf ("Out of memory. Dictionary could not be loaded.\n");

}

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

root -> child[i] = NULL;

root->end=0;

int index;

node\* tmp = root;

FILE \*dict\_file=fopen("dict.txt","r+");

if (dict\_file == NULL)

printf("Error opening dictionary");

char word\_str[20];

char meaning[200];

while((fscanf(dict\_file,"%s",word\_str))!=EOF)

{ fgets(meaning,200,dict\_file);

int word\_len = strlen(word\_str);

for(i=0;i<word\_len;i++)

{

if (isalpha (word\_str[i]))

index = (int)tolower (word\_str[i]) - 'a';

if (index > 25 || index < 0)

continue;

if (tmp -> child[index] == NULL)

{

if ((tmp -> child[index] = (node\*)malloc ( sizeof (node))) == NULL)

{

printf ("Out of memory. Dictionary could not be loaded.\n");

}

tmp = tmp -> child[index];

int j;

for ( j = 0; j < 26; j++)

tmp -> child[j] = NULL;

tmp->end=0;

}

else

tmp = tmp -> child[index];

if (i == word\_len - 1){

tmp -> end =1;

strcpy(tmp->meaning,meaning);

}

} tmp=root;

} fclose(dict\_file);

return root;

}

void unload\_rec (node\* dict\_rem)

{

node\* tmp = dict\_rem;

int i;

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

if (tmp -> child[i] != NULL)

unload\_rec (tmp -> child[i]);

free (tmp);

tmp = NULL;

}

char\* search(node \*root, char\* key)

{

node \*p = root;

int i;

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

{

int index = (int)tolower(key[i]) - 'a';

if (p->child[index]==NULL)

return NULL;

p = p->child[index];

}

if((p->end==1))

return p->meaning;

else return NULL;

}

void view()

{

FILE \*dict\_file=fopen("dict.txt","r");

if (dict\_file == NULL)

printf("Error opening dictionary");

char word\_str[20];

char meaning[200];

while((fscanf(dict\_file,"%s",word\_str))!=EOF)

{ fgets(meaning,200,dict\_file);

printf("%s\n",word\_str);

}

fclose(dict\_file);

}

void add(node\* root)

{

char word[20];

char meaning[200];

char tmpmeaning[20];

printf("Enter the word to add\n");

scanf("%s",word);

printf("Enter the meaning\n");

scanf("%s",meaning);

int word\_len = strlen(word);

int index,i;

node\* tmp=root;

for(i=0;i<word\_len;i++)

{

if (isalpha (word[i]))

index = (int)tolower (word[i]) - 'a';

if (index > 25 || index < 0)

continue;

if (tmp -> child[index] == NULL)

{

if ((tmp -> child[index] = (node\*)malloc (sizeof (node))) == NULL)

{

printf ("Out of memory. Dictionary could not be loaded.\n");

}

tmp = tmp -> child[index];

int j;

for (j = 0; j < 26; j++)

tmp -> child[j] = NULL;

tmp->end=0;

}

else

tmp = tmp -> child[index];

if (i == word\_len - 1){

tmp -> end =1;

strcpy(tmp->meaning,meaning);

}

}

FILE \*dict\_file=fopen("dict.txt","a");

fprintf(dict\_file,"\n%s ",word);fputs(meaning,dict\_file);

fclose(dict\_file);

}

int main(){

node\* root=load();

int ch;

char word[50];

char meaning[200];

printf("\n=============TRIE DICTIONARY==============\n");

while(1)

{

printf("Enter your choice.\n1.View\n2.Add\n3.Search\n0.Exit\n");

scanf("%d",&ch);

switch(ch)

{

case 0: printf("BYE!!!!");

unload\_rec(root);

exit(0);

break;

case 1: view();

break;

case 2: add(root);

break;

case 3: printf("Enter the word:\n");

scanf("%s",word);

if(search(root,word))

{

strcpy(meaning,search(root,word));

printf("Meaning:%s\n",meaning);

}

else printf("Word not found\n");

break;

default : printf("\nenter a valid key\n");

}

}

return 0;

}