#include <stdio.h>

#include <cs50.h>

#include <string.h>

#include <ctype.h>

#include <stdlib.h>

#define PATHS 26 // число букв в английском алфавите

typedef struct \_trie

{

char\* val;

struct \_trie\* paths[PATHS];

}

tries;

// глобальные переменные

tries\* root;

// прототипы функций

tries\* create();

void insert(char\* word);

bool find(char\* word);

void destroy(tries\* level);

int main(void)

{

root = create();

insert("Qu");insert("Anna");insert("Quiz");insert("A");insert("Anrat");insert("Ceylon");

if(find("Anraty"))

{

printf("Find!\n");

}

destroy(root);

}

void destroy(tries\* level)

{

tries\* ptr = level; // указатель на уровень дерева

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

{

if (ptr->paths[i] != NULL)

{

destroy(ptr->paths[i]);

}

}

free(ptr);

}

// поиск по дереву

bool find(char\* word)

{

tries\* ptr = root; // указатель на корень дерева

char\* s = word; // указатель на слово

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

{

int index = toupper(s[i]) - 65; // получили индекс i-ой буквы слова

if (ptr->paths[index] == NULL)

{

return false;

}

ptr = ptr->paths[index];

}

if (ptr->val == word)

{

return true;

}

else

{

return false;

}

}

// вставляем слово в дерево

void insert(char\* word)

{

tries\* ptr = root; // указатель на корень дерева

char\* s = word; // указатель на слово

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

{

int index = toupper(s[i]) - 65; // получили индекс i-ой буквы слова

// если путь закрыт, выделили новую память, если открыт, то прошли по нему

if (ptr->paths[index] == NULL)

{

tries\* newptr = malloc(sizeof(tries));

// проверка на ошибку

if (newptr == NULL)

{

printf("Error!\n");

exit(1);

}

// Инициализируем всю неиспользуемую память значениями NULL

for (int j = 0; j < PATHS; j++)

{

newptr->paths[j] = NULL;

}

newptr->val = NULL;

ptr->paths[index] = newptr;

ptr = newptr;

}

else

{

ptr = ptr->paths[index];

}

}

ptr->val = word;

}

// создали корень дерева

tries\* create()

{

root = malloc(sizeof(tries));

// проверка на ошибку

if (root == NULL)

{

printf("Error!\n");

exit(1);

}

// Инициализируем всю неиспользуемую память значениями NULL

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

{

root->paths[i] = NULL;

}

root->val = NULL;

//

return root;

}