using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Derevia

{

class Trie : INter

{

public Node root;

public PrefixTree()

{

root = getNode();

}

public Node getNode()

{

return new Node();

}

public void Insert(string key)

{

int length = key.Length;

int i;

Node pCrawl = root;

// iterating from the first letter to the last letter

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

{

i = key[level] - 'a';

if (i < 0)

break;

if (pCrawl.children[i] == null)

pCrawl.children[i] = getNode();

pCrawl = pCrawl.children[i];

}

pCrawl.isEndOfWord = true;

}

public int Delete(string key)

{

int length = key.Length;

int i;

Node pCrawl = root;

// iterating from the first letter to the last letter

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

{

i = key[level] - 'a';

if (i < 0)

break;

if (pCrawl.children[i] == null)

pCrawl.children[i] = getNode();

pCrawl = pCrawl.children[i];

}

if (pCrawl.isEndOfWord != true) { return (1); }

else { pCrawl.isEndOfWord = false; return (0); };

}

public int Search(string key)

{

int level;

int length = key.Length;

int i;

Node pCrawl = root;

for (level = 0; level < length; level++)

{

i = key[level] - 'a';

if (pCrawl.children[i] == null)

return 0; //Absent

pCrawl = pCrawl.children[i];

}

if (pCrawl != null)

{

if (pCrawl.isEndOfWord == true)

return 1;

return 1;

}

else

return 0; //Absent

}

public class Node

{

const int ALPHABET\_SIZE = 26;

public Node[] children = new Node[ALPHABET\_SIZE];

public bool isEndOfWord;

}

}

}