#ifndef NODE\_H

#define NODE\_H

#include<string>

using namespace std;

class Node

{

public:

Node();

Node(string k);

virtual ~Node();

void setKey(string val){ key = val;}

string getKey(){ return key;}

void setCount(int cnt) {count = cnt; }

int getCount(){ return count;}

void setRight(Node \*val){ right = val;}

Node \*getRight(){ return right;}

void setLeft(Node \*val){ left = val;}

Node \*getLeft(){ return left;}

protected:

private:

string key;

int count;

Node \*right;

Node \*left;

};

#endif // NODE\_H

#include "Node.h"

Node::Node()

{

this->key = "";

this->count = 1;

this->left = nullptr;

this->right = nullptr;

}

Node::Node(string k)

{

this->key = k;

this->count = 1;

this->left = nullptr;

this->right = nullptr;

}

Node::~Node()

{

}

#ifndef BST\_H

#define BST\_H

#include"Node.h"

class BST

{

public:

BST();

virtual ~BST();

void setRoot(Node \*val){ root = val;}

Node\* getRoot(){ return root;}

void insertNodeRE(Node \*p);

void LNR(Node \*r);

int countBeInputWord(Node \*r, Node \*p);

protected:

private:

Node \*root;

Node\* insertNodeRe(Node \*r, Node \*p);

};

#endif // BST\_H

#include "BST.h"

#include<iostream>

using namespace std;

BST::BST()

{

this->root = nullptr;

}

BST::~BST()

{

//dtor

}

Node\* BST::insertNodeRe(Node \*r, Node \*p)

{

if(r == nullptr)

return p;

else

{

if(r->getKey().compare(p->getKey()) > 0)

r->setLeft(insertNodeRe(r->getLeft(), p));

else if(r->getKey().compare(p->getKey()) < 0)

r->setRight(insertNodeRe(r->getRight(), p));

else

r->setCount(r->getCount() + 1);

}

}

void BST::insertNodeRE(Node \*p)

{

setRoot(insertNodeRe(this->root, p));

}

void BST::LNR(Node \*r)

{

if(r != nullptr)

{

LNR(r->getLeft());

cout << r->getKey() << " " << r->getCount() << endl;

LNR(r->getRight());

}

}

int BST::countBeInputWord(Node \*r, Node \*p)

{

// neu duyet ht van ko co thi return 0

if(r == nullptr )

return 0;

else

{

if(r->getKey().compare(p->getKey()) > 0)

countBeInputWord(r->getLeft(), p);

else if(r->getKey().compare(p->getKey()) < 0)

countBeInputWord(r->getRight(), p);

else return r->getCount();

}

}

#include <iostream>

#include<sstream>

#include"Node.h"

#include"BST.h"

using namespace std;

int main()

{

BST \*a = new BST();

Node \*n;

string s = "hoc sinh di hoc mon sinh hoc";

stringstream ss(s);

string tmp;

while(ss >> tmp)

{

n = new Node(tmp);

a->insertNodeRE(n);

}

a->LNR(a->getRoot());

return 0;

}