//============================================================================

// Name        : DictionaryBST.cpp

// Author      : Yash Sonar

// Version     :

// Copyright   :

// Description : Hello World in C++, Ansi-style

//============================================================================

#include <iostream>

#include<string>

using namespace std;

class dictionary;

class node

{

    string word,meaning;

    node \*left,\*right;

public:

    friend class dictionary;

    node()

    {

        left=NULL;

        right=NULL;

    }

    node(string word, string meaning)

    {

        this->word=word;

        this->meaning=meaning;

        left=NULL;

        right=NULL;

    }

};

class dictionary

{

    node \*root;

public:

    dictionary()

{

        root=NULL;

}

    void create();

    void inorder\_rec(node \*rnode);

    void postorder\_rec(node \*rnode);

    void inorder()

        {

            inorder\_rec(root);

        }

    void postorder();

    bool insert(string word,string meaning);

    int search(string key);

};

int dictionary::search(string key)

{

node \*tmp=root;

int count;

if(tmp==NULL)

{

 return -1;

}

if(root->word==key)

        return 1;

while(tmp!=NULL)

{

if(tmp->word<key)

{

    tmp=tmp->left;

    count++;

}

else if(tmp->word>key)

{

    tmp=tmp->right;

    count++;

}

else

{

 return count;

}

}

return -1;

}

void dictionary::postorder()

{

    postorder\_rec(root);

}

void dictionary::postorder\_rec(node \*rnode)

{

if(rnode)

{

    postorder\_rec(rnode->right);

    cout<<" "<<rnode->word<<" : "<<rnode->meaning<<endl;

    postorder\_rec(rnode->left);

}

}

void dictionary::create()

{

    int n;

    string wordI,meaningI;

    cout<<"\nHow many Word to insert?:\n";

    cin>>n;

    for(int i=0;i<n;i++)

    {

        cout<<"\nENter Word: ";

        cin>>wordI;

        cout<<"\nEnter Meaning: ";

        cin>>meaningI;

        insert(wordI,meaningI);

    }

}

void dictionary::inorder\_rec(node \*rnode)

{

    if(rnode)

    {

        inorder\_rec(rnode->left);

        cout<<" "<<rnode->word<<" : "<<rnode->meaning<<endl;

        inorder\_rec(rnode->right);

    }

}

bool dictionary::insert(string word, string meaning)

{

node \*p=new node(word, meaning);

if(root==NULL)

{

    root=p;

    return true;

}

node \*cur=root;

node \*par=root;

while(cur!=NULL) //traversal

{

    if(word>cur->word)

    {par=cur;

        cur=cur->right;

    }

    else if(word<cur->word)

    {

        par=cur;

        cur=cur->left;

    }

    else

    {

        cout<<"\nWord is already in the dictionary.";

        return false;

    }

}

if(word>par->word) //insertion of node

{

    par->right=p;

    return true;

}

else

{

    par->left=p;

    return true;

}

}

int main() {

    string word;

    dictionary months;

    months.create();

    cout<<"Ascending order\n";

    months.inorder();

    cout<<"\nDescending order:\n";

    months.postorder();

    cout<<"\nEnter word to search: ";

    cin>>word;

    int comparisons=months.search(word);

    if(comparisons==-1)

    {

        cout<<"\nNot found word";

    }

    else

    {

        cout<<"\n "<<word<<" found.\nNumber of Comparisons: "<<comparisons;

    }

    return 0;

}