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

// Name        : DictionaryHashing.cpp

// Author      :  Yash Sonar

// Version     :

// Copyright   :

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

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

#include <iostream>

using namespace std;

const int MAX=100;

class dictionary;

class node

{

    string key,value;

    node \*next;

public:

    friend class dictionary;

    node()

    {

        next=NULL;

    }

    node(string key,string value)

    {

        this->key=key;

        this->value=value;

        next=NULL;

    }

};

class dictionary

{

    node \*head[MAX];

public:

    dictionary()

{

        for(int i=0;i<MAX;i++)

            head[i]=NULL;

}

    int hashf(string word);

    bool insert(string,string);

    string find(string word);

    bool deleteWord(string word);

};

bool dictionary::deleteWord(string word)

{

    int index=hashf(word);

    node \*tmp=head[index];

    node \*par=head[index];

    if(tmp==NULL) //if no word is present at that index

    {

        return false;

    }

    if(tmp->key==word && tmp->next==NULL)//only one word is present

    {

        tmp->next=NULL;

        delete tmp;

        return true;

    }

    //tmp=tmp->next;

    while(tmp->key!=word && tmp->next!=NULL)

    {

        par=tmp;

        tmp=tmp->next;

    }

    if(tmp->key==word&&tmp->next!=NULL)

    {

        par->next=tmp->next;

        tmp->next=NULL;

        delete tmp;

        return true;

    }

    else    //delete at end

    {

        par->next=NULL;

        tmp->next=NULL;

        delete tmp;

        return true;

    }

    return false;

}

string dictionary::find(string word)

{

    int index=hashf(word);

    //cout<<"\nIndex in find"<<index;

    node \*start=head[index];

    if(start==NULL)

        return "-1";

    while(start!=NULL)

    {

        if(start->key==word)

            return start->value;

        start=start->next;

    }

    return "-1";

}

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

{

    int index=hashf(word);

    node \*p=new node(word,meaning);

    if(head[index]==NULL)

    {

        head[index]=p;

        cout<<"\n"<<word<<"inserted. ";

        return true;

    }

    else

    {

        node \*start=head[index];

        while(start->next!=NULL)

            start=start->next;

        start->next=p;

        cout<<"\n"<<word<<"inserted. ";

        return true;

    }

    return false;

}

int dictionary::hashf(string word)

{

    int asciiSum=0;

    for(int i=0;i<word.length();i++)

    {

        asciiSum=asciiSum+word[i];

    }

    return (asciiSum%100);

}

int main() {

    dictionary oxford;

    int choice;

    string word,meaning;

    do

    {

        cout<<"\n\*\*\*\* OXFORD DICTIONARY \*\*\*\*\n"

            <<"1.Insert Word\n"

            <<"2.Find Word\n"

            <<"3.Delete Word\n"

            <<"Enter Your Choice :";

        cin>>choice;

        switch(choice)

        {

        case 1:

            cout<<"Enter Word: ";

            cin>>word;

            cout<<"Enter Meaning: ";

            cin>>meaning;

            if(oxford.insert(word,meaning))

                cout<<endl<<word<<" inserted into dictionary.";

            else

                cout<<"\nFailed to insert.";

            break;

        case 2:

            cout<<"Enter Word to Search: ";

            cin>>word;

            meaning=oxford.find(word);

            if(meaning!="-1")

                cout<<" Word Is present.\n"<<word<<" = "<<meaning ;

            else

            {

                cout<<"\n Word Not Present";

            }

            break;

        case 3:

            cout<<"Enter Word to Delete: ";

            cin>>word;

            if(oxford.deleteWord(word))

                cout<<" Word is deleted.";

            else

            {

                cout<<"\nFailed to delete "<<word;

            }

            break;

        default:

            cout<<"\nWrong Choice.";

        }

    }while(choice!=0);

    return 0;

}