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

// Name        : SymbolTable.cpp

// Author      :  Yash Sonar

// Version     :

// Copyright   : Your copyright notice

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

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

#include <iostream>

using namespace std;

const int MAX=100;

class dictionary;

class node

{

    string identifier,scope,type;

    int lineNo;

    node \*next;

public:

    friend class SymbolTable;

    node()

    {

        next=NULL;

    }

    node(string key,string value,string type,int lineNo)

    {

        this->identifier=key;

        this->scope=value;

        this->type=type;

        this->lineNo=lineNo;

        next=NULL;

    }

    void print()

    {

        cout<<"\nIdentifier's Name:"<<identifier

                <<"\nType:"<<type

                <<"\nScope: "<<scope

                <<"\nLine Number: "<<lineNo<<endl;

    }

};

class SymbolTable

{

    node \*head[MAX];

public:

    SymbolTable()

{

        for(int i=0;i<MAX;i++)

            head[i]=NULL;

}

    int hashf(string id);

    bool insert(string id,string scope,string Type,int lineno);

    string find(string id);

    bool deleteRecord(string id);

    bool modify(string id);

};

bool SymbolTable::modify(string id)

{

    int index=hashf(id);

    //cout<<"\nIndex in find"<<index;

    node \*start=head[index];

    if(start==NULL)

        return "-1";

    while(start!=NULL)

    {

        if(start->identifier==id)

        {

                cout<<id<<" is present\n";

                cout<<"Enter Scope: ";

                cin>>(start->scope);

                cout<<"Enter Type: ";

                cin>>(start->type);

                cout<<"Enter Line Number: ";

                cin>>(start->lineNo);

            return true;

        }start=start->next;

    }

    return false;

}

bool SymbolTable::deleteRecord(string id)

{

    int index=hashf(id);

    node \*tmp=head[index];

    node \*par=head[index];

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

    {

        return false;

    }

    if(tmp->identifier==id && tmp->next==NULL)//only one identifier is present

    {

        tmp->next=NULL;

        delete tmp;

        return true;

    }

    //tmp=tmp->next;

    while(tmp->identifier!=id && tmp->next!=NULL)

    {

        par=tmp;

        tmp=tmp->next;

    }

    if(tmp->identifier==id&&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 SymbolTable::find(string id)

{

    int index=hashf(id);

    //cout<<"\nIndex in find"<<index;

    node \*start=head[index];

    if(start==NULL)

        return "-1";

    while(start!=NULL)

    {

        if(start->identifier==id)

        {

            start->print();

            return start->scope;

        }start=start->next;

    }

    return "-1";

}

bool SymbolTable::insert(string id,string scope,string Type,int lineno)

{

    int index=hashf(id);

    node \*p=new node(id,scope,Type,lineno);

    if(head[index]==NULL)

    {

        head[index]=p;

        cout<<"\n"<<id<<"inserted. ";

        return true;

    }

    else

    {

        node \*start=head[index];

        while(start->next!=NULL)

            start=start->next;

        start->next=p;

        cout<<"\n"<<id<<"inserted. ";

        return true;

    }

    return false;

}

int SymbolTable::hashf(string id)

{

    int asciiSum=0;

    for(int i=0;i<id.length();i++)

    {

        asciiSum=asciiSum+id[i];

    }

    return (asciiSum%100);

}

int main() {

    SymbolTable  st;

        int choice,lineno;

        string id,scope,type;

        do

        {

            cout<<"\n\*\*\*\* SYMBOL\_TABLE \*\*\*\*\n"

                <<"1.Insert IDENTIFIER\n"

                <<"2.Find Identifier\n"

                <<"3.Delete identifier\n"

                <<"4.Modify attributes\n"

                <<"Enter Your Choice :";

            cin>>choice;

            switch(choice)

            {

            case 1:

                cout<<"Enter Identifer's Name: ";

                cin>>id;

                cout<<"Enter Scope: ";

                cin>>scope;

                cout<<"Enter Type: ";

                cin>>type;

                cout<<"Enter Line Number: ";

                cin>>lineno;

                if(st.insert(id,scope,type,lineno))

                    cout<<endl<<id<<" inserted into Symbol Table.";

                else

                    cout<<"\nFailed to insert.";

                break;

            case 2:

                cout<<"Enter Identifier to Search: ";

                cin>>id;

                scope=st.find(id);

                if(scope!="-1")

                    cout<<" Identifier Is present.\n";

                else

                {

                    cout<<"\n Identifier Not Present";

                }

                break;

            case 3:

                cout<<"Enter Identifier to Delete: ";

                cin>>id;

                if(st.deleteRecord(id))

                    cout<<" Identifier's Record is deleted.";

                else

                {

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

                }

                break;

                case 4:

                    cout<<"Enter Identifier to Modify: ";

                    cin>>id;

                    st.modify(id);

                    break;

            default:

                cout<<"\nWrong Choice.";

            }

        }while(choice!=0);

    return 0;

}