#include <string>

#include <iostream>

#include <fstream>

#include <cstdlib>

using namespace std;

const int size = 25375;

// that is a HASH function

struct Node

{

int numeric\_code;

string state\_abbreviation; //key

string name; //key

int population;

double area;

double latitude;

double longitude;

int code;

double distance;

};

struct Link

{

Node data;

Link\* next;

Link(Node s, Link\* l = NULL)

{

data = s;

next = l;

}

};

int Hash(string s)

{

int val = 1294721;

for (int i = 0; i < s.length(); i += 1)

val = val \* 691 + s[i];

if (val < 0)

val = -val;

return val % size;

}

void trim(string& s)

{

if (!s.empty())

{

s.erase(0, s.find\_first\_not\_of(" "));

s.erase(s.find\_last\_not\_of(" ") + 1);

}

}

int main()

{

Link\* data[size];

for (int i = 0; i < size; i += 1)

data[i] = NULL;

int uniques = 0;

ifstream myfile;

myfile.open("/home/www/class/een318/named-places.txt");

cout << "open file success" << endl;

string strline;

getline(myfile, strline);

//cout << strline << endl;

int i;

while (true)

{

if (myfile.fail())

break;

getline(myfile, strline);

if (strline.length() == 0) break;

Node node;

string temp;

node.numeric\_code = strtol(strline.substr(0, 8).c\_str(), NULL, 10);

node.state\_abbreviation = strline.substr(8, 2);

node.name = strline.substr(10, 47);

trim(node.name);

temp = strline.substr(57, 10);

node.population = strtol(temp.c\_str(), NULL, 10);

temp = strline.substr(67, 14);

node.area = strtof(temp.c\_str(), NULL);

temp = strline.substr(81, 10);

node.latitude = strtof(temp.c\_str(), NULL);

temp = strline.substr(91, 11);

node.longitude = strtof(temp.c\_str(), NULL);

temp = strline.substr(102, 4);

node.code = strtol(temp.c\_str(), NULL, 10);

temp = strline.substr(106, 8);

node.distance = strtof(temp.c\_str(), NULL);

int pos = Hash(node.state\_abbreviation + node.name);

//cout << node.state\_abbreviation << ": " << pos << "\n";

bool found = false;

Link\* ptr = data[pos];

while (ptr != NULL)

{

if ((ptr->data.state\_abbreviation + ptr->data.name) == (node.state\_abbreviation + node.name))

{

found = true;

break;

}

ptr = ptr->next;

}

if (!found)

{

uniques += 1;

data[pos] = new Link(node, data[pos]);

}

else

{

//cout << "repeated string\n";

}

}

myfile.close();

cout << "read file success" << endl;

string operation, operation1;

while (true)

{

cout << "--->operation e.c [S] [N] [Q]<---" << endl;

cin >> operation;

if (operation == "Q")

return 0;

if (operation == "S")

{

cout << "input placename state:";

cin >> operation1;

cout << operation1 << endl;

int pos = Hash(operation1);

Link\* ptr = data[pos];

if (ptr != NULL)

{

cout << ptr->data.numeric\_code << "|";

cout << ptr->data.state\_abbreviation << "|";

cout << ptr->data.name << "|";

cout << ptr->data.population << "|";

cout << ptr->data.area << "|";

cout << ptr->data.latitude << "|";

cout << ptr->data.longitude << "|";

cout << ptr->data.code << "|";

cout << ptr->data.distance << endl;

}

}

if (operation == "N")

{

cout << "input placename:";

cin >> operation1;

for (i = 0; i < size; i ++)

{

if (data[i] == NULL) continue;

if (data[i]->data.name == operation1)

{

cout << data[i]->data.numeric\_code << "|";

cout << data[i]->data.state\_abbreviation << "|";

cout << data[i]->data.name << "|";

cout << data[i]->data.population << "|";

cout << data[i]->data.area << "|";

cout << data[i]->data.latitude << "|";

cout << data[i]->data.longitude << "|";

cout << data[i]->data.code << "|";

cout << data[i]->data.distance << endl;

}

}

}

}

}

