#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct person {

string ssn, dob, title, fname, lname, street, town, state, zip;

person(string s, string d, string ti, string f, string l, string str, string to, string st, string z) {

ssn = s;

dob = d;

title = ti;

fname = f;

lname = l;

street = str;

town = to;

state = st;

zip = z;

}

void print() {

cout << ssn << " " << dob << " " << title << " " << fname << " " << lname << " " << street << " " << town << " " << state << " " << zip << "\n";

}

};

struct tree {

protected:

struct node {

person\* data;

node \*left, \*right;

node(person\* p) {

data = p;

left = NULL;

right = NULL;

}

};

node\* root;

int length;

public:

tree() {

root = NULL;

}

void insert(node\*&, person\*);

void printinorder(node\*);

void read\_file(string file);

void printall() {

printinorder(root);

}

void user\_query();

person\* search(string, string);

};

void tree::insert(node\* &p, person\* x) {

if (p==NULL) {

p = new node(x);

length++;

}

else if (p->data->lname==x->lname) { //in the case that a person already exists in the list

if (p==NULL) { //with the same last name, this case checks the fname

p = new node(x);

length++;

}

else if (p->data->fname==x->fname) { //in the case that a person already exists with both the same first

p = new node(x); //and last name, a duplicate will be created anyway

length++;

}

else if (x->fname<p->data->fname) insert(p->left, x);

else if (x->fname>p->data->fname) insert(p->right, x);

}

else if (x->lname<p->data->lname) insert(p->left, x);

else if (x->lname>p->data->lname) insert(p->right, x);

}

void tree::printinorder(node\* p) {

if (p==NULL) return;

printinorder(p->left);

cout << p->data->fname << " " << p->data->lname << "\n";

printinorder(p->right);

}

void tree::read\_file(string file) {

ifstream f(file.c\_str());

if (f.fail()) {

cout << "Error opening file\n";

return;

}

for (int i=0; i<5000; i++) {

string ssn, dob, zip, title, fname, lname, street, town, state;

f >> ssn >> dob >> title >> fname >> lname >> street >> town >> state >> zip;

person\* p = new person(ssn, dob, title, fname, lname, street, town, state, zip);

insert(root,p);

if (f.fail()) break;

}

f.close();

}

person\* tree::search(string first, string last) {

node\* p = root;

if (p==NULL) return NULL;

while (p != NULL) {

if (last==p->data->lname) {

if (first==p->data->fname)

return p->data;

else if (first<p->data->fname)

p = p->left;

else

p = p->right;

}

else if (last<p->data->lname)

p = p->left;

else

p = p->right;

}

return NULL;

}

void tree::user\_query() {

string x, y, z;

while (true) {

cout << "Input first and last name of person to search:\n";

cin >> x;

if (x=="exit" || x=="Exit")

break;

else cin >> y;

if (x=="\*" && y=="\*") {

printall();

cout << "\n";

}

else {

if (search(x,y)==NULL)

cout << "Person not found.\n";

else {

person\* found = search(x,y);

found->print();

}

}

cout << "Search again? (y/n) ";

cin >> z;

if (z=="y") continue;

else if (z=="n") break;

else {

cout << "Invalid!\n";

break;

}

}

}

void main() {

tree\* database = new tree;

database->read\_file("/home/www/class/een218/ass7f132.txt");

database->printall();

database->user\_query();

}

**Output**

