#include <iostream>

#include <string>

#include <fstream>

#include <vector>

using namespace std;

class connection{

string name, type;

int a, b;

float len;

public:

connection() {}

connection(string n, string t, int A, int B, float l) {

name = n;

type = t;

a = A;

b = B;

len = l; }

void info(int p) {

cout << name << ", " << len << " miles from location ";

if(a == p)

cout << b;

else if(b == p) //I could use else, but else if is more cautious

cout << a;

else {

cout << "Error\n";

exit(1); } //if this happens, then there is a major flaw with the

return; } //program, and it should end immediately

int getInter(int current) {

if(current == a)

return b;

else if(current == b)

return a;

else {

cout << "Error\n";

exit(1); } } };

class inter { //short for intersection

float longi, lati, dist;

string name;

vector<connection\*> listCons;

short int numCons; //Dear Grader,

//I really didn't want to use numCons.

//I tried to do listCons[listCons->size()] for adding cons

//I tried listCons->push\_back(), but neither one worked

//push\_back() didn't work for my interList either

//could you explain why these two approaches didn't work

//and also suggest a better way to append items to my vectors

//thanks, Shing

public:

inter(float lo, float la, float d, string n) {

longi = lo;

lati = la;

dist = d;

name = n;

vector<connection\*> myVector(3);

listCons = myVector;

numCons = 0; }

void print() {

cout << name << endl; }

void addCon(connection \* con) {

if(numCons > listCons.capacity()-1)

listCons.resize(listCons.capacity() \* 2);

listCons[numCons] = con;

numCons++; }

void info(int p) {

cout << "\nLocation " << p << ", " << dist << " miles from " << name << endl;

if(p == 0)

return;

if(!listCons.empty())

cout << "roads leading away:\n";

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

cout << " " << i+1 << ": ";

listCons[i] -> info(p);

cout << endl; }

return; }

int getNextInt(int l, int c) {

if(c > numCons || c == 0)

return 0;

return listCons[c-1] -> getInter(l); } };

class interList {

vector<inter\*> data;

//location # is line #, so vectors/array make things simple

public:

interList() {

vector<inter\*> da(1);

data = da; }

interList(int i) {

vector<inter\*> da(i);

data = da; }

bool isInRange(int i) {

if(i > 0 && i < data.size() - 1)

return true;

else

return false; }

void add(inter \* da, int pos) {

if(pos >= data.capacity()-1)

data.resize(data.capacity()\*2);

data[pos] = da; }

void addCons(connection \* con, int n1, int n2) {

data[n1] -> addCon(con);

data[n2] -> addCon(con); }

void info(int i) {

if(i > data.size()) {

data[0] -> info(0);

return; }

data[i] -> info(i); }

int getNext(int l, int c) {

return data[l] -> getNextInt(l, c);} };

interList \* readInters() {

float longi, lati, dist;

string state, name;

int pos = 0;

interList \* myList = new interList(1000);

ifstream insects("intersections.txt");

if(insects.fail()) {

cout << "Error\n";

exit(1); }

while(true) {

insects >> lati;

if(insects.fail()) {

insects.close();

break; }

insects >> longi >> dist >> state;

getline(insects, name);

name = name.substr(1) + ", " + state;//combines name and state since they

//will only be printed in this format

//substr removes the space from getline

myList -> add(new inter(longi, lati, dist, name), pos);

pos++; }

return myList; }

void readCons(interList \* list) {

connection \* ptr; //Why is my pointer called ptr? I'm not creative.

string name, type;

int n1, n2;

float dist;

ifstream cons("connections.txt");

if(cons.fail()) {

exit(1); }

while(true) {

cons >> name;

if(cons.fail()) {

cons.close();

break; }

cons >> type >> n1 >> n2 >> dist;

ptr = new connection(name, type, n1, n2, dist);

list -> addCons(ptr, n1, n2); }

return; }

void navigate() {

interList \* myList;

myList = readInters();

readCons(myList);

int location, choice, lastWorking = 0;

string apology;

cout << "Location to start: ";

cin >> location;

while(true) {

if(!myList -> isInRange(location)) {

cout << "\nCONGRATS! You've ruined my program; I hope you're happy now.\n";

if(lastWorking != 0) {

cout << "You tried to take a road that doesn't exist.\n";

cout << "You will be escorted back to your last valid location\n";

location = lastWorking;

cout << "Input 'sorry' to continue.\n";

cin >> apology;

while(apology != "sorry") {

cout << "REPENT!\n";

apology = "";

cin >> apology;

if(apology == "sorry")

cout << "Apology accepted.\n"; } }

else {

cout << "The location you entered is not a valid location.\n";

cout << "The program will end now, and it's all your fault.\n";

exit(0); } }

lastWorking = location;

myList -> info(location);

cout << "take which road? ";

cin >> choice;

location = myList -> getNext(location, choice);

}

}

int main() {

navigate(); }