// Degree: MSc Computing

// Module: 70036 Object Oriented Design and Programming

//

// Add all of your code that pertains to question 1 to this file.

//

#include <iostream>

#include <string>

#include <list>

using namespace std;

// namespace std {

// template <typename T> class list {

// /\*\*

// \* Appends the given element value to the end of the container.

// \*/

// void push\_back(T&& value);

// /\*\*

// \* Returns a reference to the last element in the container.

// \*/

// T& back();

// };

// } // namespace std

int now() { return 0; }

class Client;

class Agent;

class Property;

class Appointment;

class DB {

    list<Appointment> db;

    list<Agent> agents;

    list<Property> properties;

    list<Client> clients;

public:

    Appointment& makeAppointment(Agent& a, Property& p, Client& v, int time);

    bool clientIsAvailable(Client& p, int time);

    bool agentIsAvailable(Agent& p, int time);

    bool propertyIsAvailable(Property& p, int time);

    Property& getProperty(int latitude, int longitude);

    Agent& getAgent(int phoneNumber, string name);

    Client& getClient(int phoneNumber);

};

//////////////////// Do not change anything above this line! ///////////////////

// Your code goes here

class Client{

public:

Client(int phoneNumber) : phoneNumber(phoneNumber) {}

int phoneNumber;

};

class Agent{

public:

Agent(int phoneNumber, string name) :phoneNumber(phoneNumber), name(name) {}

string name;

int phoneNumber;

};

class Property{

public:

Property(int latitude, int longitude) : latitude(latitude), longitude(longitude){}

int latitude;

int longitude;

};

class Appointment{

public:

Appointment(Agent& agent, Property& property, Client& client, int time, DB& db)

: agent(agent), property(property), client(client), time(time), db(db) {}

Appointment& cancel();

Agent agent;

Property property;

Client client;

int time;

DB& db;

};

Appointment& Appointment::cancel() {

for(int i=0;i<now()+1000;i++){

if(db.clientIsAvailable(client, i) && db.agentIsAvailable(agent, i) && db.propertyIsAvailable(property, i)){

time = i;

return \*this;

}

}

throw runtime\_error("No appointment could be found.");

}

bool DB::clientIsAvailable(Client& p, int time) {

for(Appointment& app : db)

if(app.client.phoneNumber == p.phoneNumber && app.time ==time)

return false;

return true;

}

bool DB::agentIsAvailable(Agent& p, int time) {

for(Appointment& app : db)

if(app.agent.phoneNumber == p.phoneNumber && app.time ==time)

return false;

return true;

}

bool DB::propertyIsAvailable(Property& p, int time) {

for(Appointment& app : db)

if(app.property.latitude == p.latitude && app.property.longitude == p.longitude && app.time ==time)

return false;

return true;

}

Property& DB::getProperty(int latitude, int longitude) {

for(Property& it: properties){

if(it.latitude== latitude && it.longitude == longitude)

return it;

}

Property prop(latitude, longitude);

properties.push\_back(prop);

return properties.back();

}

Agent& DB::getAgent(int phoneNumber, string name) {

for(Agent& it: agents){

if(it.phoneNumber == phoneNumber)

return it;

}

Agent agent(phoneNumber, name);

agents.push\_back(agent);

return agents.back();

}

Client& DB::getClient(int phoneNumber) {

for(Client& it: clients){

if(it.phoneNumber == phoneNumber)

return it;

}

Client client(phoneNumber);

clients.push\_back(client);

return clients.back();

}

Appointment& DB::makeAppointment(Agent& a, Property& p, Client& v, int time) {

Appointment app(a, p, v, time, \*this);

db.push\_back(app);

return db.back();

}

//////////////////// Do not change anything below this line! ///////////////////

int main(int /\*argc\*/ , char\* /\*argv\*/ []) {

    DB db;

    auto p1 = db.getProperty(0, 0);

    auto a1 = db.getAgent(12345, "Jules");

    auto c1 = db.getClient(54321);

    auto app = db.makeAppointment(a1, p1, c1, now() + 8);

    auto newAppointment = app.cancel();

    std::cout << newAppointment.time << std::endl;

    return 0;

}