#include<iostream>

#include<map>

#include<vector>

#include <set>

using namespace std;

struct YPX {

map<string, vector<string>>cars;

void Print() {

for (auto car : cars) {

cout << "car no : " << car.first << endl;

for (auto p : car.second) {

cout << "\t\t\t" << p << endl;

}

}

}

void showCarPenalties(string car\_no) {

auto penalties = cars[car\_no];

for (auto p : penalties) {

cout << "\t\t\t" << p << endl;

}

}

};

//void main() {

//

// YPX ypx;

//

// ypx.cars["99-CC-999"].push\_back("Suret heddi 60 azn");

// ypx.cars["99-CC-999"].push\_back("Dayanma durma 20 azn");

// ypx.cars["99-CC-999"].push\_back("Kemer 40 azn");

//

// ypx.cars["10-SS-865"].push\_back("Qosha Xett 40 azn");

// ypx.cars["10-SS-865"].push\_back("Eks Istiqametde hereket 60 azn");

//

// //ypx.Print();

// ypx.showCarPenalties("10-SS-865");

//

//

//}

//void main() {

//MultiSet

/\*multiset<int>myset;

for (size\_t i = 0; i < 10; i++)

{

int random = rand() % 9;

cout << random << " ";

myset.insert(random);

} cout << endl;

for (auto m : myset) {

cout << m << " ";

} cout << endl;\*/

/\*multimap<string, string>multimap;

multimap.insert({ "BMW","BLACK" });

multimap.insert({ "Audi", "White" });

multimap.insert({ "BMW","Yellow" });

for (auto car : multimap) {

cout << car.first << " = " << car.second << endl;

}

cout << multimap["Audi"] << endl;\*/

//}

//#include<list>

//#include<forward\_list>

//void main() {

//

// forward\_list<int>fl;//only next

//

// list<int>dl;//both prev and next

//

// for (size\_t i = 0; i < 10; i++)

// {

// fl.push\_front(i + 1);

// dl.push\_front(i + 1);

// }

//

// for (auto i = fl.begin(); i != fl.end(); i--) {

// cout << \*i << " ";

// }cout << endl;

//

// for (auto i = dl.begin(); i != dl.end(); i--) {

// cout << \*i << " ";

// }cout << endl;

//}

#include<list>

#include<iomanip>

using namespace std;

class Kitty {

string name;

string nickname;

int age;

public:

Kitty():name(""),nickname(""),age(0)

{

}

Kitty(const string& name, const string& nickname, const int& age)

:name(name), nickname(nickname), age(age) {

}

int GetAge()const {

return age;

}

string GetName()const {

return name;

}

string GetNickName() const {

return nickname;

}

};

void printKitties(const list<Kitty> kitties) {

for (Kitty kitty : kitties) {

cout << setw(15) << left << kitty.GetName() <<

setw(20) << left << kitty.GetNickName() << kitty.GetAge() << endl;

}

}

//Predicates

bool byNameAsc(const Kitty& cat1, const Kitty& cat2) {

return cat1.GetName() < cat2.GetName();}

bool byAgeAsc(const Kitty& cat1, const Kitty& cat2) {

return cat1.GetAge() < cat2.GetAge();

}

bool byAgeDesc(const Kitty& cat1, const Kitty& cat2) {

return cat1.GetAge() > cat2.GetAge();

}

bool tooOld(const Kitty& kitty) {

return kitty.GetAge() > 20;

}

bool nickGreaterThan7(const Kitty& kitty) {

return kitty.GetNickName().length() > 7;

}

void main() {

//list<Kitty>kitties;

//kitties.push\_back(Kitty("Toplan", "Toplan123", 35));

//kitties.push\_back(Kitty("Mestan", "Qara", 10));

//kitties.push\_back(Kitty("Toplan", "Black jav jav", 7));

//kitties.push\_back(Kitty("Garfield", "Hungry Cat", 50));

//printKitties(kitties);

//cout << endl;

////kitties.sort(byNameAsc);

////kitties.sort(byAgeDesc);

////bool(\*ptr)(const Kitty&) = nickGreaterThan7;

////kitties.remove\_if(tooOld);

//kitties.remove\_if(nickGreaterThan7);

//printKitties(kitties);

//class Phone = > name, number

// class Phonebook = > list<Phone>phones;

//add

//showAll

//adi 5 den qisa olanlari gostermek

// ada gore siralama

// adin uzunluguna gore siralama

//elave

//telefon nomresi 070 ve ya 077 olanlari gostermek

}