#include <iostream>

#include <set>

#include <map>

#include <vector>

#include <list>

#include <forward\_list>

#include <iomanip>

using namespace std;

//

////1) set

////2) map

////3) multiset

////4) multimap

//

////class Human

////{

//// string name;

//// int age;

////public:

//// Human(const string& name, const int& age)

//// {

//// this->name = name;

//// this->age = age;

//// }

////

//// void Show() const

//// {

//// cout << "Name : " << name << endl;

//// cout << "Age : " << age << endl;

//// }

////};

//

////

////int main()

////{

//// //set<int> s;

////

//// //for (int i = 0; i < 10; i++)

//// //{

//// // int num = rand() % 9;

//// // cout << num << " ";

//// // s.insert(num);

//// //}

//// //cout << endl;

////

//// //for ( auto i : s)

//// //{

//// // cout << i << endl;

//// //}

////

//// /\*map<int, string> mymap;

//// mymap.insert({ 12,"Ferrari" });

//// mymap.insert({ 0,"Ford Mustang" });

//// mymap.insert({ 3,"Chvey Cruze" });

//// mymap.insert({ 17,"Hyundai Sanata" });

//// mymap.insert({ 3,"Maserati" });

////

//// for (auto item : mymap)

//// {

//// cout << item.first << " " << item.second << endl;

//// }\*/

////

//// //map<string, Human> humans;

//// //humans.insert({ "Aze123" , Human("John", 25)});

//// //humans.insert({ "Aze183" , Human("Leyla", 22)});

//// //humans.insert({ "Aze153" , Human("Axmed", 23)});

//// //

//// //for (auto human : humans)

//// //{

//// // cout << "Seria No : " << human.first << endl;

//// // human.second.Show();

//// //}

////

//// map<string, string> ypx;

//// ypx.insert({ "10-ky-248", "chevrolet cruz 2lt" });

//// ypx.insert({ "99-dd-999", "mercedes maybach" });

//// ypx.insert({ "88-bb-888", "tesla" });

//// ypx.insert({ "11-TT-111", "RANGE ROVER" });

////

//// for (auto car : ypx)

//// {

//// cout << car.first << " - " << car.second << endl;

//// }cout << endl;

////

//// ypx["10-KY-248"] = "Range Rover";

////

//// for (auto car : ypx)

//// {

//// cout << car.first << " - " << car.second << endl;

//// }cout << endl;

////

//// ypx["77-ZZ-777"] = "Mercedes";

////

//// for (auto car : ypx)

//// {

//// cout << car.first << " - " << car.second << endl;

//// }

////

//// string car\_no;

//// cout << "Car no : ";

//// cin >> car\_no;

////

//// auto car = ypx[car\_no];

//// cout << "car model :" << car << endl;

////

//// return 0;

////}

//

////struct Car

////{

//// map<string, string> properties;

//// void Print() const

//// {

//// for (auto property : properties)

//// {

//// cout << property.first << " - " << property.second << endl;

//// }

//// }

////};

//

////int main()

////{

// //Car c1;

// //c1.properties["Name"] = "Range Rover";

// //c1.properties["Speed"] = "Range 300";

// //c1.properties["Mass"] = "2000kg";

//

// //Car c2;

// //c2.properties["Name"] = "Nissan CTR350";

// //c2.properties["Speed"] = "320";

// //c2.properties["Mass"] = "1255kg";

// //c2.properties["Engine type"] = "Compressor";

//

// /\*c1.Print();

// cout << endl;

// c2.Print();\*/

//

// //string hasTurbo = "";

// //cout << "Yes or No" << endl;

// //cin >> hasTurbo;

// //if (hasTurbo == "yes")

// //{

// // c2.properties.insert({ "Has Turbo", "Yes" });

// //}

// //cout << endl;

// //c2.Print();

//

//

////}

//

//class Penalty

//{

//private:

// string name;

// double fine;

// string date;

//public:

// Penalty() = default;

//

// Penalty(const string& name, const double& fine, string date)

// {

// this->name = name;

// this->fine = fine;

// this->date = date;

// }

//

// string GetPenaltyName() const

// {

// return name;

// }

// double GetPenaltyFine() const

// {

// return fine;

// }

// string GetPenaltyDate() const

// {

// return date;

// }

//

// void Show() const

// {

// cout << "Name : " << GetPenaltyName() << endl;

// cout << "Fine : " << GetPenaltyFine() << endl;

// cout << "Date : " << GetPenaltyDate() << endl;

// }

//};

//

//class Car

//{

//public:

// map<string, string>properties;

// map<int, Penalty> penalties;

// void Show() const

// {

// for (auto property : properties)

// {

// cout << property.first << " - " << property.second << endl;

// }

// cout << endl;

// for (auto penalty : penalties)

// {

// cout << penalty.first << " - ";

// penalty.second.Show();

// }

//

// }

//};

//

//int main()

//{

// Penalty p1("Cerime 1", 60, \_\_DATE\_\_);

// Penalty p2("Cerime 3", 70, \_\_DATE\_\_);

// Penalty p3("Cerime 2", 90, \_\_DATE\_\_);

// Car c1;

// c1.properties["Name"] = "Range Rover";

// c1.properties["Speed"] = "300";

// c1.properties["Mass"] = "2000kg";

// c1.penalties[1] = p1;

// c1.penalties[2] = p2;

// c1.penalties[3] = p3;

//

// Penalty p4("Cerime 4", 88, \_\_DATE\_\_);

// Penalty p5("Cerime 5", 99, \_\_DATE\_\_);

// Car c2;

// c2.properties["Name"] = "Maserati";

// c2.properties["Speed"] = "345";

// c2.properties["Mass"] = "1000kg";

// c2.properties["Engine type"] = "Compressor";

// c2.penalties[1] = p4;

// c2.penalties[2] = p5;

//

// map<string, Car> ypx;

//

// ypx.insert({ "11-TT-111", c1 });

// ypx.insert({ "22-ZZ-222", c2 });

//

// string no = "";

// cin >> no;

// auto car = ypx[no];

// car.Show();

//

//

//

//

//

// return 0;

//}

//class YPX

//{

//public:

// 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("Kemer 20 azn");

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

//

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

// ypx.cars["10-SS-865"].push\_back("Eks istiqametde hereket 40 azn");

//

// //ypx.Print();

//

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

//

//}

//void main()

//{

// multiset<int> myset;

//

// for (int i = 0; i < 10; i++)

// {

// int number = rand() % 9;

// cout << number << " ";

// myset.insert(number);

// }cout << endl;

//

// for (auto m : myset)

// {

// cout << m << " ";

// }cout << endl;

//

//

//}

//void main()

//{

// 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; // error

// //cout << multimap["BMW"] << endl; // error

//}

//void main()

//{

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

// list<int> dl;

//

// for (int 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;

//

// cout << \*fl.begin() << endl;

//

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

// {

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

// }

// cout << endl;

//}

//class Kitty

//{

// string name;

// string nickname;

// int age;

//public:

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

// {

//

// }

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

// {

// this->name = name;

// this->nickname = nickname;

// this->age = age;

// }

//

// string GetName() const

// {

// return name;

// }

// string GetNickname() const

// {

// return nickname;

// }

// int GetAge() const

// {

// return age;

// }

//};

//

//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 - qisa bool qaytaran funksiya

////Sort ucun

//bool byNameAsc(const Kitty& k1, const Kitty& k2)

//{

// return k1.GetName() < k2.GetName();

//}

//

////Sort ucun

//bool byAgeAsc(const Kitty& k1, const Kitty& k2)

//{

// return k1.GetAge() < k2.GetAge();

//}

//

//// remove\_if ucun

//bool tooOld(const Kitty& kitty)

//{

// return kitty.GetAge() > 20;

//}

//

////remove\_if ucun

//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", 12));

// kitties.push\_back(Kitty("Black", "Jav Jav", 67));

// kitties.push\_back(Kitty("Garfield", "Hungry Cat", 16));

// PrintKitties(kitties);

//

// kitties.sort(byNameAsc);

// cout << endl;

// PrintKitties(kitties);

//

// kitties.sort(byAgeAsc);

// cout << endl;

// PrintKitties(kitties);

//

// kitties.remove\_if(tooOld);

// cout << endl;

// PrintKitties(kitties);

//

// kitties.remove\_if(nickGreaterThan7);

// cout << endl;

// PrintKitties(kitties);

//

// cout << endl;

// PrintKitties(kitties);

//}

// Homework

//#include <assert.h>

//class Phone

//{

//private:

// string name;

// string phone\_number;

//public:

// Phone() = default;

//

// Phone(const string& name, const string& phoneNumber)

// {

// SetName(name);

// SetPhoneNumber(phoneNumber);

// }

//

// string GetName() const

// {

// return name;

// }

// string GetPhoneNumber() const

// {

// return phone\_number;

// }

//

// void SetName(const string& name)

// {

// assert(!name.empty() && "Name should be filled!");

// this->name = name;

// }

//

// void SetPhoneNumber(const string& phoneNumber)

// {

// assert(!phoneNumber.empty() && "Phone number should be filled!");

// this->phone\_number = phoneNumber;

// }

//

// void PrintPhone() const

// {

// cout << "Name : " << GetName() << endl;

// cout << "Phone Number : " << GetPhoneNumber() << endl;

// }

//};

//

//class PhoneBook

//{

// vector <Phone> phones;

//

// void PrintPhones() const

// {

// for (Phone phone : phones)

// {

// phone.PrintPhone();

// }

// }

//

// void AddPhone(const Phone& phone)

// {

// phones.push\_back(phone);

// }

//};

//

//bool byName(const Phone& p1, const Phone& p2)

//{

// return p1.GetName() < p2.GetName();

//}

//

//bool byNameLength(const Phone& p1, const Phone& p2)

//{

// return p1.GetName().length() < p2.GetName().length();

//}

//

//bool nameGreaterThan5(const Phone& p)

//{

// return p.GetName().length();

//}

//

//bool StartsWith070(const Phone& p)

//{

// return p.GetName()

//}

//bool StartsWith077(const Phone& p)

//

//

//int main()

//{

//

//

// return 0;

//}