#include <iostream>

#include <iomanip>

#include <algorithm>

#include <list>

#include <fstream>

#include <vector>

#include <numeric>

using namespace std;

//bool desc(int a, int b)

//{

// return a > b;

//}

//

//int main()

//{

// list<int> mylist;

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

// {

// mylist.push\_back(rand() % 20);

// }

// cout << "Unsorted array" << endl;

// for (auto d : mylist)

// {

// cout << d << " ";

// }cout << endl;

//

// cout << endl << "Sorted" << endl;

// mylist.sort(desc);

//

// for (auto d : mylist)

// {

// cout << d << " ";

// }cout << endl;

//

// return 0;

//}

//

//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;

//}

//class byAge

//{

// bool order;

//public:

// byAge(bool order) : order(order)

// {

//

// }

//

// bool operator()(const Kitty& cat1, const Kitty& cat2)

// {

// if (order)

// {

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

// }

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

// }

//};

//

//class StringHelper

//{

//public:

// static string ToLower(const string& text)

// {

// string empty = "";

// for (int i = 0; i < text.length(); i++)

// {

// if (text[i] >= 65 && text[i] <= 90)

// {

// empty += text[i] + 32;

// }

// else

// {

// empty += text[i];

// }

// }

// return empty;

// }

//};

//

//class byAge

//{

// string order;

//public:

// byAge(string order)

// {

// this->order = StringHelper::ToLower(order);

// }

//

// bool operator()(const Kitty& cat1, const Kitty& cat2)

// {

// if (order == "asc")

// {

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

// }

// else if (order == "desc")

// {

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

// }

// return false;

// }

//};

//

//class FunctorConstants

//{

//public:

// const static string Ascending;

// const static string Descending;

// const static string AscendingNick;

// const static string DescendingNick;

//};

//

//const string FunctorConstants::Ascending = "asc";

//const string FunctorConstants::AscendingNick = "asc-nick";

//const string FunctorConstants::Descending = "desc";

//const string FunctorConstants::DescendingNick = "desc-nick";

//

//class byName

//{

// string order;

//public:

// byName(const string& order)

// {

// this->order = StringHelper::ToLower(order);

// }

//

// bool operator()(const Kitty& cat1, const Kitty& cat2)

// {

// if (order == FunctorConstants::Ascending)

// {

// return cat1.GetName() > cat2.GetName();

// }

// else if (order == FunctorConstants::Descending)

// {

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

// }

// else if (order == FunctorConstants::AscendingNick)

// {

// return cat1.GetNickname() > cat2.GetNickname();

// }

// else if (order == FunctorConstants::DescendingNick)

// {

// return cat1.GetNickname() < cat2.GetNickname();

// }

// return false;

// }

//};

//

//void main()

//{

// list<Kitty> kitties;

// kitties.push\_back(Kitty("Aslan", "Aslan123", 35));

// kitties.push\_back(Kitty("Black", "Black123", 67));

// kitties.push\_back(Kitty("Z-JavJav", "Z-Jav jav", 12));

// 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);\*/

// kitties.sort(byName(FunctorConstants::Descending));

//

// cout << endl;

// PrintKitties(kitties);

//}

// Functor vs Predicate

// predicates

//int increment(int data)

//{

// return data + 1;

//}

//int doubleIncrement(int data)

//{

// return data \* 2;

//}

//

//// functor

//class Increment

//{

// int num;

//public:

// Increment(int num) : num(num) {}

//

// int operator()(int arr\_num)

// {

// return arr\_num + num;

// }

//};

//

//class Decrement

//{

// double percent;

//public:

// Decrement(const double& percent) : percent(percent) {}

//

// double operator()(double arr\_num)

// {

// return arr\_num - arr\_num \* percent / 100;

// }

//};

//

//void main()

//{

// const int size = 8;

// double arr[size] = { 100,200,300,400,500,600,700,800 };

//

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

// {

// cout << arr[i] << " ";

// }cout << endl;

//

// transform(arr, arr + size, arr, Increment(10));

//

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

// {

// cout << arr[i] << " ";

// }cout << endl;

//

// transform(arr, arr + size, arr, doubleIncrement);

//

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

// {

// cout << arr[i] << " ";

// }cout << endl;

//

// transform(arr, arr + size, arr, Decrement(10));

//

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

// {

// cout << arr[i] << " ";

// }cout << endl;

//

//

//}

//bool desc(int a, int b)

//{

// return a > b;

//}

//

//int mult(int a, int b)

//{

// return a \* b;

//}

//

//void main()

//{

// vector<int> v;

//

// srand(time(0));

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

// {

// v.push\_back(i + 1);

// }

// //v.push\_back(7);

//

// for (auto data : v)

// {

// cout << data << " ";

// }cout << endl;

//

// reverse(v.begin(), v.end());

//

// for (auto data : v)

// {

// cout << data << " ";

// }cout << endl;

//

// sort(v.begin(), v.end(), desc);

//

// for (auto data : v)

// {

// cout << data << " ";

// }cout << endl;

//

// cout << "Max : " << \*max\_element(v.begin(), v.end()) << endl;

// cout << "Min : " << \*min\_element(v.begin(), v.end()) << endl;

//

// int sum = accumulate(v.begin(), v.end(), 0);

// cout << "Sum : " << sum << endl;

//

// int product = accumulate(v.begin(), v.end(), 1, mult);

// cout << "Product : " << product << endl;

//

// int c = count(v.begin(), v.end(), 7);

// cout << "Count : " << c << endl;

//

// auto data = find(v.begin(), v.end(), 7);

// cout << \*data << endl;

//

// sort(v.begin(), v.end());

// bool result = binary\_search(v.begin(), v.end(), 11);

// cout << "Result : " << boolalpha << result << endl;

//}

//

//void print(int i)

//{

// cout << "Data : " << i << endl;

//}

//

//void main()

//{

// vector<int> v;

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

// {

// v.push\_back(rand() % 15);

// }

//

// //for\_each(v.begin(), v.end(), print);

//

// // Print() funksiyasini yazmamaq ucun (eger programda artiq islenmeyecekse)

// for\_each(v.begin(), v.end(), [](int i) {

// cout << "Salam : " << i << endl;

// });

//

// sort(v.begin(), v.end(), [](int a, int b)->bool {

// return a > b;

// });

//

// cout << endl;

// for\_each(v.begin(), v.end(), [](int i) {

// cout << "Salam : " << i << endl;

// });

//

// auto p = find\_if(v.begin(), v.end(), [](int data)->bool {

// return data < 10;

// });

//

// cout << \*p << endl;

//}

class Student

{

string name;

string surname;

double score;

int age;

int hasScholarship;

public:

Student(const string& name, const string& surname, const double& score, const int& age, const int& hasScholarship)

:name(name), surname(surname), score(score), age(age), hasScholarship(hasScholarship){}

string GetName() const

{

return name;

}

string GetSurname() const

{

return surname;

}

double GetScore() const

{

return score;

}

int GetAge() const

{

return age;

}

int GetScholarShip() const

{

return hasScholarship;

}

};

void dataWriteToFileApp(const string& filename, const Student& s)

{

ofstream fout(filename, ios::app);

if (fout.is\_open())

{

fout << "Name : ";

fout << s.GetName();

fout << " Surname : ";

fout << s.GetSurname();

fout << " Age : ";

fout << s.GetAge();

fout << " Score : ";

fout << s.GetScore();

fout << " Scholarship : ";

fout << s.GetScholarShip();

fout << "\n";

}

else

{

cout << filename << " does not exist!" << endl;

}

fout.close();

}

void main()

{

vector<Student> s;

s.push\_back(Student("John", "Johnlu", 95, 14,99));

s.push\_back(Student("Aysel", "Aliyeva", 87, 24, 55));

s.push\_back(Student("Elsen", "Elsenli", 94, 34, 234));

s.push\_back(Student("Ayshen", "Memmedova", 86, 45, 345));

//for\_each(s.begin(), s.end(), [](Student& s) {

// if (s.GetScore() > 90)

// {

// //s.name = "Best";

// cout << s.GetName() << " = " << s.GetScore() << endl;

// }

//});

cout << endl;

for\_each(s.begin(), s.end(), [](Student s) {

if (s.GetAge() < 20)

{

dataWriteToFileApp("young.txt", s);

}

});

for\_each(s.begin(), s.end(), [](Student s) {

if (s.GetScholarShip() > 100)

{

dataWriteToFileApp("richStudents.txt", s);

}

});

/\*int c = count\_if(s.begin(), s.end(), [](Student s) {

return s.GetScore() > 90;

});

cout << "90+ count : " << c << endl;

int c2 = count\_if(s.begin(), s.end(), [](Student s) {

return s.GetName()[0] == 'E';

});

cout << "Name start with E count : " << c2 << endl;\*/

}