#include<iostream>

#include<iomanip>

#include<list>

#include<algorithm>

#include<numeric>

using namespace std;

//

//bool desc(int a, int b) {

// return a > b;

//}

//

//

//void main() {

//

// list<int>mylist;

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

// {

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

// }

// cout << "unsorted" << endl;

// for (auto d : mylist) {

// cout << d << " ";

// }

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

// mylist.sort(desc);

// for (auto d : mylist) {

// cout << d << " ";

// }

//

//}

//

//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;

// }

//}

//

////Functor

//

////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 (size\_t i = 0; i < text.length(); i++)

// {

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

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

// }

// else {

// empty += text[i];

// }

// }

// return empty;

// }

//};

//

//#include<string>

//

//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;

//};

//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;

// }

//

//};

//

//

//

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

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

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

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

//

//void main() {

//

// list<Kitty>kitties;

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

// kitties.push\_back(Kitty("Mestan", "Aslan123", 10));

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

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

//

// printKitties(kitties);

// cout << endl;

//

// //kitties.sort(byAge("Desc"));

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

// //kitties.sort(byAgeDesc);

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

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

// //kitties.remove\_if(nickGreaterThan7);

//

// printKitties(kitties);

//

//

//

//}

//Functor vs Predicate

//int increment(int data) {

// return data + 1;

// }

//int doubleIncrement(int data) {

// return data \* 2;

//}

//

//class increment {

// int num;

//public:

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

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

// return arr\_num + num;

// }

//};

//

//

//class decrease {

// double percent;

//public:

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

// {

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

// } cout << endl;

//

// transform(arr, arr + size, arr, decrease(50));

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

// {

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

// } cout << endl;

//

//

//}

#include<vector>

bool Desc(int a, int b) {

return a > b;

}

int mult(int a, int b) {

return a \* b;

}

#include<time.h>

//void main() {

//vector<int>v;

//srand(time(0));

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

//{

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

// //v.push\_back(rand()%10+1);

//}

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

//for (auto data : v) {

// cout << data << " ";

//}cout << endl;

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

/\*sort(v.begin(), v.end(), Desc);

for (auto data : v) {

cout << data << " ";

}cout << endl;\*/

/\*cout << "maximum : " << \*max\_element(v.begin(), v.begin()+v.size()/2) << endl;

cout << "minimum : " << \*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 << endl;\*/

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

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

/\*auto data = find(v.begin(), v.end(), 555);

try

{

cout << \*data << endl;

}

catch (...)

{

cout << "Not found" << endl;

}\*/

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

bool result = binary\_search(v.begin(), v.end(), 3);

cout << "Result : " << result << endl;\*/

//}

//void print(int i) {

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

// }

//

//void main() {

//

// vector<int>v;

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

// {

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

// }

//

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

//

//

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

// // cout << "Salam : " << i\*10/3 << endl;

// // });

//

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

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

// });

//

// cout << endl;

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

// return a > b;

// });

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

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

// });\*/

//

//

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

// // return data < 10;

// // });

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

//

//}

class Student {

public:

string name;

string surname;

double score;

Student(const string&name,const string&surname,

const double&score)

:name(name),surname(surname),score(score)

{

}

string GetName()const {

return name;

}

string GetSurname()const {

return surname;

}

double GetScore()const {

return score;

}

};

void main() {

vector<Student>s;

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

s.push\_back(Student("Aysel", "Eliyeva", 75));

s.push\_back(Student("Ayshen", "Mammadova", 98));

s.push\_back(Student("Elshen", "Eliyev", 100));

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

if (s.GetScore() > 85) {

//s.name = "best";

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

}

});

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

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

});\*/

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

return s.GetScore() > 95;

});

cout << "Best : " << count << endl;\*/

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

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

});

cout << "Best : " << count << endl;

}