1. Map

a. Khai báo:

map<kiểu dữ liệu, kiểu dữ liệu> tên\_map;

int main ()

{

map<string,int> mp = {

{ "alpha", 20 },

{ "beta", 20 },

{ "alpha", 10 },

{ "gamma", 30 },

{ "alpha", 10 },

};

for (auto x: mp) {

cout << x.first << ": " << x.second << endl;

}

cout<< mp.size() <<endl;

return 0;

}

Kq:

alpha: 20

beta: 20

gamma: 30

3

**b. Tìm phần tử trong map C++ bằng hàm find**

**Hàm count** là một hàm thành viên trong class std:map, có tác dụng đếm số lần xuất hiện của phần tử trong map C++ thông qua khóa của nó.

Chúng ta sử dụng hàm count trong C++ với cú pháp sau đây:

mp.count(key);

#include <iostream>

#include <map>

using namespace std;

int main() {

map<char,int> mp;

mp['a']=100;

mp['b']=200;

mp['c']=300;

//Đếm số lần xuất hiện của phần tử tồn tại trong map

cout << mp.count('b') <<endl;

//Đếm số lần xuất hiện của phần tử không tồn tại trong map

cout << mp.count('f') <<endl;

return 0;

}

Kq:

1

0

**Hàm equal\_range** là một hàm thành viên trong class std:map, có tác dụng tìm phạm vi của tất cả các phần tử có khóa giống với khóa chỉ định trong map.

Chúng ta sử dụng hàm equal\_range trong C++ với cú pháp sau đây:

mp.find(key);

#include <iostream>

#include <map>

using namespace std;

//Tạo hàm xuất map

void dump(map<char,int>& mp)

{

for (auto x: mp) {

cout << x.first << ":" << x.second << " ";

}

cout << endl;

}

int main() {

map<char,int> mp;

mp.insert(make\_pair('a', 1));

mp.insert(make\_pair('b', 2));

mp.insert(make\_pair('c', 3));

mp.insert(make\_pair('d', 1));

mp.insert(make\_pair('e', 2));

dump(mp);

//Tìm phần tử có khóa bằng 'c' trong map

auto ret = mp.equal\_range('c');

//Xóa phần tử vừa tìm thấy

mp.erase (ret.first,ret.second);

dump(mp);

return 0;

}

Kq:

a:1 b:2 c:3 d:1 e:2

a:1 b:2 d:1 e:2

**Hàm lower\_bound** là một hàm thành viên trong class std::map, có tác dụng tìm vị trí **phần tử đầu tiên** trong map có khóa lớn hơn hoặc bằng với khóa chỉ định.

Chúng ta sử dụng hàm lower\_bound trong C++ với cú pháp sau đây:

mp.lower\_bound(key);

Hàm lower\_bound() sẽ trả về trình lặp trỏ đến vị trí **phần tử đầu tiên** có khóa lớn hơn hoặc bằng với khóa chỉ định. Và nếu không tìm thấy, hàm sẽ trả về trình lặp trỏ đến vị trí cuối cùng trong map.

#include <iostream>

#include <map>

using namespace std;

//Tạo hàm xuất map

void dump(map<char,int>& mp)

{

for (auto x: mp) {

cout << x.first << ":" << x.second << " ";

}

cout << endl;

}

int main() {

map<char,int> mp;

mp.insert(make\_pair('a', 1));

mp.insert(make\_pair('b', 2));

mp.insert(make\_pair('c', 3));

mp.insert(make\_pair('d', 1));

mp.insert(make\_pair('e', 2));

mp.insert(make\_pair('f', 3));

dump(mp);

/\*Tìm vị trí phần tử đầu tiên có khóa lớn hơn hoặc bằng 'b' trong map\*/

auto itr1 = mp.lower\_bound('b'); // itr1 trỏ đến b:2

//Tìm vị trí phần tử đầu tiên có khóa lớn hơn 'e' trong map

auto itr2 = mp.upper\_bound('e'); // itr2 trỏ đến f:3

//Xóa các phần tử trong phạm vi [itr1, itr2)

mp.erase (itr1, itr2);

dump(mp);

return 0;

}

Kết quả, các phần tử trong phạm vi từ b:2 đến trước f:3 đã bị xóa đi.

a:1 b:2 c:3 d:1 e:2 f:3

a:1 f:3

**Hàm upper\_bound** là một hàm thành viên trong class std::map, có tác dụng tìm vị trí **phần tử đầu tiên** trong map có khóa lớn hơn khóa chỉ định.

Chúng ta sử dụng hàm upper\_bound trong C++ với cú pháp sau đây:

**mp.upper\_bound(key);**

#include <iostream>

#include <map>

using namespace std;

int main() {

map<char,int> mp;

mp['a']=100;

mp['b']=200;

mp['c']=300;

mp['d']=300;

mp['e']=300;

mp['f']=300;

/\*Duyệt map mp\*/

for (auto x: mp) {

cout << x.first << ":" << x.second << " ";

}

cout << endl; //a:1 b:2 c:3 d:1 e:2

//Tìm phần tử có khóa bằng 'b' trong map

auto itr1 = mp.lower\_bound('b'); // itr1 trỏ đến b:200

//Tìm vị trí phần tử đầu tiên có khóa lớn hơn 'e' trong map

auto itr2 = mp.upper\_bound('e'); // itr2 trỏ đến f:300

//In các phần tử trong phạm vi (itr1, itr2)

for (auto it=itr1; it!=itr2; ++it)

cout << (\*it).first << ":" << (\*it).second << ' ';

return 0;

}

Kq:

a:100 b:200 c:300 d:300 e:300 f:300

b:200 c:300 d:300 e:300

**BÀI 1- 1224: Hệ thống đăng kí**

###### **Mô tả**

Một dịch vụ e-mail mới “Berlandesk” sẽ được mở ra ở Berland trong tương lai gần. Người quản trị muốn dự án sớm đi vào hoạt động, do đó họ yêu cầu bạn giúp đỡ. Bạn được họ đề nghị thực hiện phần giao thức của đăng kí hệ thống. Hệ thống này làm việc theo nguyên tắc sau đây.

Mỗi khi có người dùng mới muốn đăng ký, anh ta gửi tới hệ thống một yêu cầu với tên của mình. Nếu như tên này không tồn tại trong cơ sở dữ liệu của hệ thống thì nó được đưa vào cơ sở dữ liệu và người dùng sẽ nhận được phúc đáp là OK, xác nhận đăng ký thành công. Nếu tên đã tồn tại trong cơ sở dữ liệu thì hệ thống sẽ lập một tên người dùng mới và tên mới đó được gửi cho người sử dụng và được đưa vào cơ sở dữ liệu. Tên mới được hình thành bởi các nguyên tắc sau. Số (bắt đầu từ 1) được viết ngay sau tên (tên1, tên2, ...). Số này cần nhận giá trị nhỏ nhất i sao cho têni chưa tồn tại trong cơ sở dữ liệu.

###### **Dữ liệu vào**

Dòng đầu tiên chứa số nguyên n (1 ≤ n ≤ 105). Tiếp theo có n dòng chứa các yêu cầu tới hệ thống. Mỗi yêu cầu ghi trên một dòng là một tên không rỗng và bao gồm không quá 32 kí tự (chỉ gồm các chữ cái Latin in thường).

###### **Dữ liệu ra**

Ghi ra n dòng, với mỗi yêu cầu ghi ra phúc đáp là OK nếu yêu cầu đăng kí là thành công hoặc ghi ra tên mới nếu tên yêu cầu đã tồn tại trong cơ sở dữ liệu của hệ thống.

###### **Ví dụ dữ liệu vào**

4

abacaba

acaba

abacaba

acab

###### **Ví dụ dữ liệu ra**

OK

OK

abacaba1

OK

###### **Gợi ý**

Ví dụ dữ liệu vào:

6

first

first

second

second

third

second

Ví dụ dữ liệu ra:

OK

first1

OK

second1

OK

second2

##### BÀI 2- 1225: Người chiến thắng

###### **Mô tả**

Người chiến thắng trong trò chơi bài “Berlogging” phổ biến ở Berland được xác định theo các quy tắc sau đây. Nếu kết thúc trò chơi chỉ có một người chơi với số điểm tối đa thì anh ta là người chiến thắng. Việc xác định người chiến thắng sẽ trở nên khó khăn hơn nếu có nhiều người chơi có cùng số điểm tối đa. Trong mỗi vòng chơi, một người chơi sẽ được hoặc mất một số điểm nào đó. Thông tin về của mỗi vòng chơi có dạng “name score”, trong đó name là tên của người chơi và score là số điểm đạt được ở vòng này (score là một số nguyên). Nếu số điểm âm thì có nghĩa là người chơi bị mất điểm ở vòng này. Vì vậy khi kết thúc trò chơi, nếu hai hoặc nhiều người chơi có cùng số điểm tối đa (gọi số điểm tối đa là m) thì người chiến thắng là một trong số những người này mà có được ít nhất m điểm đầu tiên. Ban đầu mỗi người chơi có 0 điểm. Khi kết thúc trò chơi có ít nhất một người chơi có số điểm dương.

###### **Dữ liệu vào**

Dòng đầu tiên chứa số nguyên n (1  ≤  n  ≤  105) là số vòng chơi. Tiếp theo có n vòng, mỗi dòng chứa thông tin của một vòng theo thứ tự thời gian theo định dạng “name score”, ở đó name là một xâu các chữ cái Latin in thường với chiều dài từ 1 đến 32 và score là một số nguyên nằm trong đoạn từ -104 đến 104.

###### **Dữ liệu ra**

Ghi ra tên của người chơi chiến thắng.

###### **Ví dụ dữ liệu vào**

3

andrew 5

mike 6

mike -1

###### **Ví dụ dữ liệu ra**

andrew

##### BÀI 3- 1254: Tham quan

###### **Mô tả**

Điều kiện làm việc cho phép trong khoảng thời gian n ngày liên tục nhân viên có thể đăng ký nghỉ phép hoặc nghỉ bù. Steve quyết định tận dụng khả năng hiếm có này để nghỉ ngơi liên tục tất cả k ngày mà mình được quyền. Sau khi cân nhắc, Steve đăng ký tham gia một chuyến du lịch. Mỗi ngày công ty du lịch sẽ đưa khách tới tham quan một thành phố. Trong bảng giới thiệu chương trình tham quan mỗi thành phố được ghi tương ứng với một số nguyên dương không vượt quá 109, ngày thứ i sẽ tham thành phố ai. Các thành phố khác nhau tương ứng với các số nguyên khác nhau. Một thành phố có thể được tới nhiều lần. Steve có thể tham gia chuyến du lịch bắt đầu từ ngày tùy chọn và dĩ nhiên, muốn thăm được càng nhiều thành phố khác nhau càng tốt.

Cho n, k và ai (1 ≤ k ≤ n ≤ 105). Hãy xác định ngày bắt đầu kỳ nghỉ để nghỉ đủ k ngày và số lượng các thành phố Steve có thể tới thăm là lớn nhất. Ngày đầu tiên trong chương trình của Công ty được đánh số là 1. Nếu tồn tại nhiều cách lựa chọn thì đưa ra ngày sớm nhất.

###### **Dữ liệu vào**

Dòng đầu tiên chứa 2 số nguyên n và k. Dòng thứ hai chứa n số nguyên a1, a2, ..., an.

###### **Dữ liệu ra**

Đưa ra một số nguyên là ngày được chọn để bắt đầu tham quan.

###### **Ví dụ dữ liệu vào**

7 3

1 2 1 3 1 2 1

###### **Ví dụ dữ liệu ra**

2

##### BÀI 4- 1229: Xâu con thú vị

###### **Mô tả**

An gán một số cho mỗi chữ cái Latin in thường để thể hiện mức độ yêu thích của mình với chữ cái đó (anh ta gán số âm cho các chữ cái mà anh ta không thích).

Bình thích các xâu có độ dài lớn hơn 1 mà chữ cái bắt đầu và kết thúc giống nhau.

Ngoài ra, An và Bình có một xâu s. Bây giờ họ đang tính xem có bao nhiêu xâu con t của xâu s mà Bình thích (tức là xâu con t có độ dài lớn hơn 1 và có chữ cái bắt đầu, kết thúc giống nhau) và tổng giá trị của tất cả các chữ cái (được gán bởi An), trừ chữ cái đầu tiên và cuối cùng là bằng 0. Bạn hãy giúp An và Bình tính số lượng xâu con t như vậy.

###### **Dữ liệu vào**

Dòng đầu tiên chứa 26 số nguyên xa, xb, …, xz (-105 ≤ xi ≤ 105) là các giá trị được gán cho các kí tự a, b, c, …, z tương ứng. Dòng thứ hai chứa một xâu s (1 ≤ độ dài xâu s ≤ 105) bao gồm các chữ cái Latin in thường.

###### **Dữ liệu ra**

Ghi ra câu trả lời của bài toán.

###### **Ví dụ dữ liệu vào**

1 1 -1 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 1 8 1 1 1 1 1 1

xabcab

###### **Ví dụ dữ liệu ra**

2

###### **Gợi ý**

Ví dụ dữ liệu vào:

1 1 -1 1 1 1 1 1 1 1 1 1 1 1 1 7 1 1 1 8 1 1 1 1 1 1

aaa

Ví dụ dữ liệu ra:

2

Giải thích:

* Trong ví dụ thứ nhất, các xâu con thỏa mãn yêu cầu là "abca" và "bcab";
* Trong ví dụ thứ hai, các xâu con thỏa mãn yêu cầu là "aa" và "aa" ("aa" xuất hiện 2 lần).

**Bài 1. Tài khoản (6 điểm)**

An có một phần mềm máy tính, người dùng muốn sử dụng được phần mềm này thì phải đăng ký tài khoản. Tuy nhiên với số lượng người dùng ngày càng nhiều, dẫn tới việc có thể người dùng đăng ký tài khoản trùng nhau (tài khoản người đăng ký sau trùng với tài khoản người đã đăng ký trước đó). Để xử lý việc này An nghĩ ra một ý tưởng như sau:

* Nếu tên tài khoản người dùng đăng ký chưa được lưu trong hệ thống thì người dùng sẽ được đăng ký với tên tài khoản đó;
* Nếu tên tài khoản người dùng muốn đăng ký trùng với tên tài khoản đã được đăng ký trước đó thì hệ thống sẽ tự động thêm một số nguyên dương nhỏ nhất vào sau tên tài khoản đó sao cho tên tài khoản đó chưa được đăng ký.

Ví dụ giả sử trong hệ thống đã tồn tại tên “tinhoc” mà người dùng tiếp theo đăng ký trùng tên thì sẽ được lưu tên tài khoản là “tinhoc1”, trong trường hợp có thêm người dùng đăng ký trùng tên “tinhoc” thì hệ thống sẽ lưu tiếp theo là “tinhoc2”, “tinhoc3”...

Bạn hãy giúp An lập trình để thực hiện ý tưởng của mình.

**Dữ liệu:** Vào từ tệp văn bản acco.inp dòng đầu tiên chứa số nguyên dương n (1 ≤ n ≤ 10⁴) là số lượng tên tài khoản mà người dùng muốn đăng ký. Dòng thứ i trong n dòng tiếp theo chứa một xâu kí tự s\_i (1 ≤ i ≤ n) chỉ gồm các chữ cái tiếng Anh thường ‘a’...’z’ và có độ dài không quá 10 kí tự là tên các tài khoản muốn đăng ký.

**Kết quả:** Ghi ra tệp văn bản acco.out gồm n dòng là kết quả các tài khoản của người dùng được lưu trong hệ thống tương ứng với dữ liệu vào.

**Ràng buộc:**

* Có 20% số test ứng với 20% số điểm của bài thỏa mãn: n ≤ 10²;
* 20% số test khác ứng với 20% số điểm của bài thỏa mãn: n ≤ 10³;
* 60% số test còn lại ứng với 60% số điểm của bài không có thêm ràng buộc nào.

