

CÁCH ĐÁNH GIÁ ĐIỂM THỰC HÀNH

HỌC PHẦN: IT3150 – Project 1- 2023.1

I. Quy định, yêu cầu:

- Tài liệu và nội dung thực hành chấm điểm trên hệ thống:
<https://lab.soict.hust.edu.vn/>
- Bài tập trên lớp chấm điểm tự động (các bài không chấm trên hệ thống làm vào máy tính → làm báo cáo thực hành – Theo mẫu).
- Hạn nộp báo cáo trên Teams (Bài tập trên lớp + Bài tập về nhà): 1 tuần.

II. Đánh giá điểm thực hành

- Chuyên cần (đúng giờ, nghiêm túc trong giờ học) - Điểm danh trên Teams: 10%
- Báo cáo thực hành (bài tập trên lớp + Về nhà) theo mẫu nộp trên Teams: 40%
- Trắc nghiệm – Form trên Teams: 10%
- Kiểm tra thực hành: 40%. (Tiết 2,3 buổi thực hành thứ 5).

Điểm thưởng: 5% → 10% (Cho Mục 1,2 điểm TB từ 9-10).

Tham gia thực hành đúng giờ đầy đủ theo thời khóa biểu (nếu có lý do không đi thực hành đúng giờ được thì gửi mail xin phép thực hành bù trước 1 ngày qua mail hoalt@soict.hust.edu.vn, Tiêu đề: đăng ký học bù – IT3040 – MaLopTH.

Các kíp có thể bù:

TT	Thời gian, địa điểm, Tuần học	Mã nhóm	Mã lớp
1			
2			
3			
4			
5			
6			
7			

Nếu nghỉ không có lý do 3 buổi, không thực hành bù thì điểm chuyên cần, báo cáo và BTVN coi như 0 điểm thực hành.

Contents

Bài 1: Phân tích việc gửi mã của một cuộc thi lập trình.....	3
Bài 2: Phân tích dữ liệu công dân.....	13

Table of Figure

Figure 1 b1- Code	5
Figure 2 b1- Input 1.....	7
Figure 3 b1- Output 1.....	7
Figure 4 b1- Input 2.....	8
Figure 5 b1- Output 2.....	8
Figure 6 b1- Input 3.....	9
Figure 7 b1- Output 3.....	9
Figure 8 b1- Input 4.....	10
Figure 9 b1- Output 4.....	10
Figure 10 b2- Code	14
Figure 11 b2- Input 1.....	17
Figure 12 b2- Output 1.....	17
Figure 13 b2- Input 2.....	18
Figure 14 b2- Output 2.....	18
Figure 15 b2- Input 3.....	19
Figure 16 b2- Output 3.....	20
Figure 17 b2- Input 4.....	20
Figure 18 b2- Output 4.....	21

Báo cáo tuần 8

Bài 1: Phân tích việc gửi mã của một cuộc thi lập trình

Dữ liệu gửi dự thi lập trình gồm một dãy các dòng, mỗi dòng có các thông tin sau:

<UserID> <ProblemID> <TimePoint> <Status> <Point>

trong đó người dùng <UserID> gửi mã của mình để giải quyết vấn đề <ProblemID> tại thời điểm <TimePoint> và nhận trạng thái <Trạng thái> và điểm <Điểm>

- <UserID>: chuỗi có độ dài từ 3 đến 10
- <ProblemID>: chuỗi có định dạng Pxy trong đó x, y là các chữ số 0,1,...,9 (ví dụ P03, P10)
- <TimePoint>: chuỗi biểu thị thời điểm có định dạng HH:MM:SS (ví dụ: 09:45:20 nghĩa là thời điểm 9 giờ 45 phút 20 giây)
- <Status>: chuỗi có hai trường hợp (ERR, OK)
- <Point>: số nguyên từ {0, 1, 2, ..., 10}

Người dùng có thể gửi mã để giải quyết từng vấn đề nhiều lần. Điểm mà người dùng nhận được cho một bài toán là điểm tối đa trong số các bài gửi cho bài toán đó.

Thực hiện một chuỗi các truy vấn thuộc các loại sau:

- ?total_number_submissions: trả về số lượng bài dự thi
- ?number_error_submission: trả về số lượng bài gửi có trạng thái ERR
- ?number_error_submission_of_user <UserID>: trả về số lần gửi có trạng thái ERR của người dùng <UserID>
- ?total_point_of_user <UserID>: trả về tổng điểm của người dùng <UserID>
- ?number_submission_Period <from_time_point> <to_time_point>: trả về số lần gửi trong khoảng thời gian từ <from_time_point> đến <to_time_point> (đã bao gồm)

Figure 1 b1- Code

Source code

```

1  //Mai Minh Hoàng
2  //20215381
3  #include <iostream>
4  #include<set>
5  #include<sstream>
6  #include<unordered_map>
7  #include<vector>
8
9  using namespace std;
10
11 long countSubmit_81 = 0; // Biến đếm tổng số lượng bài nộp
12 long sumtotalError_81 = 0; // Biến đếm tổng số lỗi
13 unordered_map<string, long> totalError_81; // Bản đồ lưu tổng số lỗi theo người dùng
14 unordered_map<string, int > totalUser_Problem_point_81; // Bản đồ lưu tổng điểm theo người dùng và vấn đề
15 unordered_map<string, long> total_point_81; // Bản đồ lưu tổng điểm theo người dùng
16 long time_submit_81[86400]; // Mảng lưu thời gian nộp
17
18 int main()
19 {
20     ios_base::sync_with_stdio(0); //tắt đồng bộ
21     string s;
22     while (getline(cin, s)) { //nhập đầu vào
23         if (s == "#") { //gặp # thì dừng
24             break;
25         }
26         stringstream ss(s); // phân tách mỗi dòng
27         string UserId;
28         string Problem;
29         long point;
30         string Status;
31         string Time;
32         ss >> UserId >> Problem >> Time >> Status >> point;
33
34         countSubmit_81++; // Tăng biến đếm tổng số lượng bài nộp
35         if (Status == "ERR") {

```

Mai Minh Hoàng – 20215381

```
36         sumtotalError_81++; // Tăng biến đếm tổng số lỗi
37         totalError_81[UserId]++; // Tăng số lỗi của người dùng
38     }
39     totalError_81[Status] += point; // Cập nhật điểm lỗi
40     if (point > totalUser_Problem_point_81[UserId + Problem]) {
41         total_point_81[UserId] += point - totalUser_Problem_point_81[UserId + Problem]; // Cập nhật tổng điểm
42         totalUser_Problem_point_81[UserId + Problem] = point; // Cập nhật điểm của người dùng và vấn đề
43     }
44     //tách giờ phút giây từ Time
45     int h = stoi(Time.substr(0, 2));
46     int m = stoi(Time.substr(3, 5));
47     int sss = stoi(Time.substr(6));
48     time_submit_81[h * 60 * 60 + m * 60 + sss] += 1; // Cập nhật thời gian nộp
49 }
50 for (int i = 0; i <= 86400; i++) { //mỗi thời điểm sẽ có tổng submit bằng chính nó cộng với tất cả thời gian trước đó
51     time_submit_81[i] += time_submit_81[i - 1]; // Cập nhật thời gian nộp
52 }
53 while (getline(cin, s)) { //nhập tiếp
54     if (s == "#") { //gặp # thì dừng
55         break;
56     }
57     if (s[8] == 's') { // nếu chuỗi dạng number_submission_period
58
59         stringstream ss(s);
60         string a, b, c;
61         ss >> a >> b >> c; //tách ra a b c trong xâu
62         int h = stoi(b.substr(0, 2)); //tách ra giờ
63         int m = stoi(b.substr(3, 5)); //phút
64         int sss = stoi(b.substr(6)); //giây
65         int timee = h * 60 * 60 + m * 60 + sss; //tương tự trên nhưng là thời điểm sau
66         int h1 = stoi(c.substr(0, 2));
67         int m1 = stoi(c.substr(3, 5));
68         int sss1 = stoi(c.substr(6));
69         int timee1 = h1 * 60 * 60 + m1 * 60 + sss1;
70
71         cout << time_submit_81[timee1] - time_submit_81[timee - 1] << "\n"; // In ra thời gian nộp
72     }
```

```
73     }
74     else
75     {
76         if (s[7] == 'n') { //nếu chuỗi dạng ?total_number_submissions
77             cout << countSubmit_81 << "\n"; // In ra tổng số lượng bài nộp
78         }
79         else if (s[8] == 'e') {
80             if (s == "?number_error_submission") { //nếu chuỗi là ?number_error_submission
81                 cout << sumtotalError_81 << "\n"; // In ra tổng số lỗi
82             }
83             else { //nếu chuỗi dạng ?number_error_submission_of_user
84                 stringstream ss(s);
85                 string a, b;
86                 ss >> a >> b;
87                 cout << totalError_81[b] << "\n"; // In ra tổng số lỗi theo người dùng
88             }
89         }
90     }
91     else { //nếu chuỗi dạng ?total_point_of_user
92         stringstream ss(s);
93         string a, b;
94         ss >> a >> b;
95         cout << total_point_81[b] << "\n"; // In ra tổng điểm theo người dùng
96     }
97 }
98 }
99 }
```

Test:

Figure 2 b1- Input 1

Input

```
U0003 P01 20:46:11 OK 9
U0001 P01 05:12:30 OK 6
U0001 P01 09:27:28 ERR 0
U0001 P02 09:28:55 ERR 0
U0003 P02 07:34:50 OK 4
U0003 P02 15:12:51 OK 6
U0003 P02 03:39:08 OK 8
U0003 P01 07:44:03 OK 3
U0001 P01 16:21:13 OK 6
U0001 P02 15:24:40 OK 6
#
?total_number_submissions
#
```

Figure 3 b1- Output 1

Correct output

10



User output

10



Figure 4 b1- Input 2

Input

```
U0002 P01 20:46:11 OK 9
U0003 P01 05:12:30 OK 6
U0004 P01 09:27:28 ERR 0
U0001 P02 09:28:55 ERR 0
U0003 P02 07:34:50 OK 4
U0001 P02 15:12:51 OK 6
U0001 P02 03:39:08 OK 8
U0002 P01 07:44:03 OK 3
U0003 P01 16:21:13 OK 6
U0004 P02 15:24:40 OK 6
U0003 P01 07:53:00 ERR 0
U0002 P02 02:40:29 OK 4
U0005 P02 20:32:52 ERR 0
U0002 P01 18:22:43 OK 5
U0001 P01 20:16:56 OK 8
U0003 P02 15:33:53 ERR 0
U0003 P01 05:18:29 ERR 0
U0004 P02 01:09:05 ERR 0
U0004 P02 17:18:43 ERR 0
```

Figure 5 b1- Output 2

```
U0004 P02 01:09:05 ERR 0
U0004 P02 17:18:43 ERR 0
U0005 P01 00:05:58 OK 5
#
?total_number_submissions
#
```

Correct output

20



User output

20



Figure 6 b1- Input 3

Input

```
U0003 P02 21:06:37 OK 5
U0001 P01 22:03:44 OK 9
U0002 P02 19:52:51 ERR 0
U0003 P01 04:11:34 OK 6
U0002 P02 18:48:29 OK 9
U0003 P01 01:51:56 ERR 0
U0001 P01 01:03:42 OK 5
U0001 P01 08:26:37 ERR 0
U0001 P02 04:55:30 OK 5
U0002 P01 05:47:39 OK 3
#
?number_submission_period 00:20:04 16:16:07
?number_submission_period 03:43:46 21:00:49
?number_submission_period 01:03:32 21:34:17
?number_submission_period 09:34:25 20:23:59
?number_submission_period 01:57:18 05:16:47
?number_submission_period 04:28:12 22:22:20
?number_submission_period 05:26:37 09:34:43
?number_submission_period 13:34:46 13:57:30
```

Figure 7 b1- Output 3

```
?number_submission_period 05:26:37 09:34:43
?number_submission_period 13:34:46 13:57:30
?number_submission_period 08:23:03 11:34:29
?number_submission_period 07:03:51 16:12:14
#
```

Correct output

```
6
6
9
2
2
7
2
0
1
1
```

User output

```
6
6
9
2
2
7
2
0
1
1
```

Figure 8 b1- Input 4

Input

```
U0003 P01 20:46:11 OK 9
U0001 P01 05:12:30 OK 6
U0001 P01 09:27:28 ERR 0
U0001 P02 09:28:55 ERR 0
U0003 P02 07:34:50 OK 4
U0003 P02 15:12:51 OK 6
U0003 P02 03:39:08 OK 8
U0003 P01 07:44:03 OK 3
U0001 P01 16:21:13 OK 6
U0001 P02 15:24:40 OK 6
#
?total_point_of_user U0002
?total_point_of_user U0002
?total_point_of_user U0002
?total_point_of_user U0001
?total_point_of_user U0001
?total_point_of_user U0001
?total_point_of_user U0001
?total_point_of_user U0002
```

Figure 9 b1- Output 4

```
?total_point_of_user U0001
?total_point_of_user U0001
?total_point_of_user U0001
?total_point_of_user U0002
?total_point_of_user U0003
?total_point_of_user U0001
#
```

Correct output

```
0
0
0
12
12
12
12
12
0
17
12
```

User output

```
0
0
0
12
12
12
12
12
0
17
12
```

Code:

```

//Mai Minh Hoàng
//20215381
#include <iostream>
#include<set>
#include<sstream>
#include<unordered_map>
#include<vector>

using namespace std;

long countSubmit_81 = 0; // Biến đếm tổng số lượng bài nộp
long sumtotalError_81 = 0; // Biến đếm tổng số lỗi
unordered_map<string, long> totalError_81; // Bản đồ lưu tổng số lỗi theo người dùng
unordered_map<string, int > totalUser_Problem_point_81; // Bản đồ lưu tổng điểm theo
người dùng và vấn đề
unordered_map<string, long> total_point_81; // Bản đồ lưu tổng điểm theo người dùng
long time_submit_81[86400]; // Mảng lưu thời gian nộp

int main()
{
    ios_base::sync_with_stdio(0); //tắt đồng bộ
    string s;
    while (getline(cin, s)) { //nhập đầu vào
        if (s == "#") { //gặp # thì dừng
            break;
        }
        stringstream ss(s); // phân tách mỗi dòng
        string UserId;
        string Problem;
        long point;
        string Status;
        string Time;
        ss >> UserId >> Problem >> Time >> Status >> point;

        countSubmit_81++; // Tăng biến đếm tổng số lượng bài nộp
        if (Status == "ERR") {
            sumtotalError_81++; // Tăng biến đếm tổng số lỗi
            totalError_81[UserId]++; // Tăng số lỗi của người dùng
        }
        totalError_81[Status] += point; // Cập nhật điểm lỗi
        if (point > totalUser_Problem_point_81[UserId + Problem]) {
            total_point_81[UserId] += point -
totalUser_Problem_point_81[UserId + Problem]; // Cập nhật tổng điểm
            totalUser_Problem_point_81[UserId + Problem] = point; // Cập
nhập điểm của người dùng và vấn đề
        }
        //tách giờ phút giây từ Time
        int h = stoi(Time.substr(0, 2));
        int m = stoi(Time.substr(3, 5));
        int sss = stoi(Time.substr(6));
        time_submit_81[h * 60 * 60 + m * 60 + sss] += 1; // Cập nhật thời gian
nộp
    }
    for (int i = 0; i <= 86400; i++) { //mỗi thời điểm sẽ có tổng submit bằng
chính nó cộng với tất cả thời gian trước đó
        time_submit_81[i] += time_submit_81[i - 1]; // Cập nhật thời gian nộp
    }
}

```

```

while (getline(cin, s)) { //nhập tiếp
    if (s == "#") { //gặp # thì dừng
        break;
    }
    if (s[8] == 's') { // nếu chuỗi dạng number_submission_period

        stringstream ss(s);
        string a, b, c;
        ss >> a >> b >> c; //tách ra a b c trong xâu
        int h = stoi(b.substr(0, 2)); //tách ra giờ
        int m = stoi(b.substr(3, 5)); //phút
        int sss = stoi(b.substr(6)); //giây
        int timee = h * 60 * 60 + m * 60 + sss; //tương tự trên nhưng là
thời điểm sau

        int h1 = stoi(c.substr(0, 2));
        int m1 = stoi(c.substr(3, 5));
        int sss1 = stoi(c.substr(6));
        int timee1 = h1 * 60 * 60 + m1 * 60 + sss1;

        cout << time_submit_81[timee1] - time_submit_81[timee - 1] <<
"\n"; // In ra thời gian nộp

    }
    else
        if (s[7] == 'n') { //nếu chuỗi dạng ?total_number_submissions
            cout << countSubmit_81 << "\n"; // In ra tổng số lượng bài
nộp

        }
        else if (s[8] == 'e') {
            if (s == "?number_error_submission") { //nếu chuỗi là
?number_error_submission

                cout << sumtotalError_81 << "\n"; // In ra tổng số
lỗi

            }
            else { //nếu chuỗi dạng ?number_error_submission_of_user
                stringstream ss(s);
                string a, b;
                ss >> a >> b;
                cout << totalError_81[b] << "\n"; // In ra tổng số
lỗi theo người dùng

            }

        }
        else { //nếu chuỗi dạng ?total_point_of_user
            stringstream ss(s);
            string a, b;
            ss >> a >> b;
            cout << total_point_81[b] << "\n"; // In ra tổng điểm theo
người dùng

        }
    }
}

```

Bài 2: Phân tích dữ liệu công dân

Cho một DataBase về công dân, thực hiện các truy vấn trên DataBase này.

Đầu vào

Đầu vào bao gồm hai khối: khối đầu tiên là DataBase và khối thứ hai là danh sách các truy vấn. Hai khối cách nhau bằng một dòng chứa ký tự *.

1. Khối đầu tiên (DataBase về công dân) gồm các dòng (số dòng có thể lên tới 100000), mỗi dòng là thông tin về một người và có dạng:

`<code> <date_of_birth> <father_code> <mother_code> <is_alive> <region_code>`

trong đó:

- `<code>`: mã của người là chuỗi có độ dài 7
- `<date_of_birth>`: ngày sinh của người đó và có định dạng YYYY-MM-DD (ví dụ 1980-02-birth>), `<date_of_birth>` là trước 3000-12-31
- `<father_code>` và `<mother_code>` là mã của cha và mẹ: chúng cũng là chuỗi có độ dài 7. Nếu mã là 0000000 thì người hiện tại không có thông tin về bố hoặc mẹ
- `<is_alive>`: ký tự có 2 giá trị: 'Y' nghĩa là người đó vẫn còn sống, và 'N' nghĩa là người hiện tại đã chết.
- `<region_code>`: mã vùng nơi người đó sinh sống

2. Khối thứ hai là danh sách các truy vấn (số lượng truy vấn có thể lên tới 100000) trên DataBase bao gồm các lệnh sau:

- `NUMBER_PEOPLE`: trả về số lượng người (số dòng của DataBase)
- `NUMBER_PEOPLE_BORN_AT <date>`: trả về số người có ngày sinh bằng `<date>`
- `MOST_ALIVE_ANCESTOR <code>`: tìm tổ tiên cao nhất (xa nhất về khoảng cách thế hệ) của người `<code>` nhất định. Trả về khoảng cách thế hệ giữa tổ tiên được tìm thấy và người đã cho
- `NUMBER_PEOPLE_BORN_BETWEEN <from_date> <to_date>`: tính số người có ngày sinh trong khoảng từ `<from_date>` đến `<to_date>` (`<from_date>` và `<to_date>` có dạng YYYY-MM-DD, `<to_date>` là trước đó 3000-12-31)
- `MAX_UNRELATED_PEOPLE`: tìm một tập con gồm những người trong đó có hai người bất kỳ trong tập con đó không có cha/mẹ-con và kích thước của tập con đó là lớn nhất. Trả về kích thước của tập hợp con được tìm thấy.

Khối thứ hai được kết thúc bằng một dòng chứa ***.

Đầu ra

- Mỗi dòng trình bày kết quả của truy vấn tương ứng (được mô tả ở trên).

Figure 10 b2- Code

Source code

```

1  //Mai Minh Hoàng
2  //20215381
3  #include <list>
4  #include <iostream>
5  #include <set>
6  #include <sstream>
7  #include <unordered_map>
8  #include <vector>
9  #include <map>
10 using namespace std;
11
12 long countPeople_81 = 0; // Biến đếm tổng số người
13 long sumtotalPeople_81 = 0; // Biến đếm tổng số người
14 map<string, long> totalPeople_81; // Bản đồ lưu tổng số người theo ngày sinh
15 unordered_map<string, int> notAlone_81; // Bản đồ kiểm tra người không một mình
16 unordered_map<string, int> maxAcestor_81; // Bản đồ lưu số tổ tiên tối đa của mỗi người
17 unordered_map<string, int> peopleBirth_81; // Bản đồ lưu số người theo ngày sinh
18 long countAlone_81 = 0; // Biến đếm số người một mình
19 long countGroup_81 = 0; // Biến đếm số nhóm
20
21 int main()
22 {
23     ios_base::sync_with_stdio(0);
24     string s;
25
26     // Tạo bản đồ cho tất cả các ngày từ năm 1900 đến 2030
27     for (int i = 1900; i < 2030; i++) {
28         for (int j = 1; j <= 12; j++) {
29             for (int k = 1; k <= 31; k++) {
30                 string year = to_string(i);
31                 string month = (j < 10) ? "0" + to_string(j) : to_string(j);
32                 string day = (k < 10) ? "0" + to_string(k) : to_string(k);
33
34                 string date = year + "-" + month + "-" + day;
35                 totalPeople_81[date] = 0; // Khởi tạo số người sinh vào ngày đó là 0

```

```

36         }
37     }
38 }
39
40 // Đọc dữ liệu từ đầu vào
41 while (getline(cin, s)) {
42     if (s == "*") { // gặp * thì dừng
43         break;
44     }
45     stringstream ss(s); // phân tách đầu vào
46     string UserId;
47     string Birth;
48     string fCode;
49     string mCode;
50     string isAlive;
51     string rCode;
52     ss >> UserId >> Birth >> fCode >> mCode >> isAlive >> rCode;
53
54     countPeople_81++; // Tăng biến đếm tổng số người
55     if ((fCode != "00000000") && (mCode != "00000000")) {
56         maxAcestor_81[UserId] = (maxAcestor_81[fCode] > maxAcestor_81[mCode] ? maxAcestor_81[fCode] + 1 : maxAcestor_81[mCode] + 1);
57         countGroup_81++; // Tăng biến đếm số nhóm
58         notAlone_81[UserId] = 1; // Đánh dấu người này không một mình
59         notAlone_81[fCode] = 1; // Đánh dấu cha/mẹ của người này không một mình
60         notAlone_81[mCode] = 1; // Đánh dấu cha/mẹ của người này không một mình
61     }
62     else {
63         notAlone_81[UserId] = 0; // Đánh dấu người này một mình
64     }
65
66     peopleBirth_81[Birth] += 1; // Tăng số người sinh vào ngày Birth
67     totalPeople_81[Birth] += 1; // Tăng tổng số người sinh vào ngày Birth
68 }
69 int previousValue = 0;
70 for (auto& p : totalPeople_81) { // cộng dồn ngày sau bằng tổng tất cả người sinh các ngày trước đó
71     int currentValue = p.second;
72     p.second = previousValue + currentValue; // Cập nhật tổng số người sinh đến ngày hiện tại

```

Mai Minh Hoàng – 20215381

```
73     previousValue = p.second; //gán lại để tới vòng lặp tiếp
74
75 }
76 for (auto& p : notAlone_81) { //duyệt qua notAlone_81
77     if (p.second == 0) {
78         countAlone_81++; // Đếm số người một mình
79     }
80 }
81 while (getline(cin, s)) {
82     if (s == "***") { //gặp *** thì dừng
83         break;
84     }
85     if (s[7] == 'P') { //chuỗi là NUMBER_PEOPLE_BORN_BETWEEN <from_date> <to_date>
86         if (s.length() > 34) {
87             stringstream ss(s);
88             string a, b, c;
89             ss >> a >> b >> c;
90
91             cout << totalPeople_81[c] - totalPeople_81[b] + peopleBirth_81[b] << "\n"; // Trả về số người sinh
92         }
93         else if (s.length() == 13) { //chuỗi là NUMBER_PEOPLE
94             cout << countPeople_81 << "\n"; // Trả về tổng số người
95         }
96         else { //chuỗi là NUMBER_PEOPLE_BORN_AT <date>
97             stringstream ss(s);
98             string a, b;
99             ss >> a >> b;
100             cout << peopleBirth_81[b] << "\n"; // Trả về số người sinh vào ngày <b>
101         }
102     }
103 }
104 else
105     if (s[0] == 'M' && s[1] == '0') { //chuỗi là MOST_ALIVE_ANCESTOR <code>
106         stringstream ss(s);
107         string a, b;
108         ss >> a >> b;
109         cout << maxAcestor_81[b] << "\n"; // Trả về số tổ tiên tối đa của người b
110     }
111     else { //chuỗi là MAX_UNRELATED_PEOPLE
112         // Trả về tổng số người một mình và số nhóm chính là kết quả để 2 người bất kỳ trong nhóm không phải
113         cout << countAlone_81 + countGroup_81 << "\n";
114     }
115
116 }
117 }
118 }
```

Test:

Figure 11 b2- Input 1

Input

```
0000001 1920-09-29 0000000 0000000 Y 00005
0000002 1920-04-19 0000000 0000000 Y 00001
0000003 1940-05-14 0000001 0000002 Y 00004
0000004 1948-12-20 0000001 0000002 Y 00002
0000005 1920-06-19 0000000 0000000 Y 00003
0000006 1920-05-16 0000000 0000000 Y 00001
0000007 1940-11-06 0000005 0000006 Y 00004
0000008 1944-04-18 0000005 0000006 Y 00004
0000009 1920-11-08 0000000 0000000 Y 00003
0000010 1920-11-18 0000000 0000000 Y 00003
*
NUMBER_PEOPLE
***
```

Figure 12 b2- Output 1

Correct output

10



User output

10



Figure 13 b2- Input 2

Input

```
0000001 1920-05-09 0000000 0000000 Y 00005
0000002 1920-10-21 0000000 0000000 Y 00001
0000003 1943-01-25 0000001 0000002 Y 00004
0000004 1941-05-16 0000001 0000002 Y 00004
0000005 1920-08-27 0000000 0000000 Y 00002
0000006 1920-05-10 0000000 0000000 Y 00002
0000007 1943-11-06 0000005 0000006 Y 00001
0000008 1945-07-14 0000005 0000006 Y 00004
0000009 1920-06-19 0000000 0000000 Y 00001
0000010 1920-03-02 0000000 0000000 Y 00001
0000011 1947-03-29 0000009 0000010 Y 00001
0000012 1947-02-12 0000009 0000010 Y 00001
0000013 1972-05-28 0000008 0000012 Y 00004
0000014 1967-05-22 0000008 0000012 Y 00001
0000015 1969-03-26 0000007 0000012 Y 00004
0000016 1971-03-09 0000007 0000012 Y 00001
0000017 1963-10-13 0000003 0000012 Y 00001
0000018 1971-03-20 0000003 0000012 Y 00002
```

Figure 14 b2- Output 2

```
0000018 1971-03-20 0000003 0000012 Y 00002
0000019 1966-10-12 0000008 0000012 Y 00004
0000020 1965-04-16 0000008 0000012 Y 00002
*
MAX_UNRELATED_PEOPLE
***
```

Correct output

14

User output

14

Figure 15 b2- Input 3

Input

```

0000001 1920-03-17 0000000 0000000 Y 00003
0000002 1920-05-26 0000000 0000000 Y 00001
0000003 1944-05-23 0000001 0000002 Y 00003
0000004 1941-04-14 0000001 0000002 Y 00004
0000005 1920-11-07 0000000 0000000 Y 00004
0000006 1920-09-30 0000000 0000000 Y 00003
0000007 1941-01-03 0000005 0000006 Y 00003
0000008 1942-12-21 0000005 0000006 Y 00001
0000009 1920-12-02 0000000 0000000 Y 00002
0000010 1920-06-17 0000000 0000000 N 00005
0000011 1949-10-17 0000009 0000010 Y 00001
0000012 1945-05-20 0000009 0000010 Y 00001
0000013 1966-07-15 0000007 0000011 Y 00002
0000014 1965-05-01 0000007 0000011 Y 00003
0000015 1970-05-05 0000004 0000012 Y 00002
0000016 1967-05-17 0000004 0000012 Y 00004
0000017 1961-05-20 0000004 0000007 Y 00001
0000018 1966-08-13 0000004 0000007 Y 00001

```

```

0000096 1930-06-19 0000000 0000000 Y 00002
0000097 1930-12-02 0000000 0000000 Y 00003
0000098 1930-12-01 0000000 0000000 Y 00001
0000099 1930-09-28 0000000 0000000 Y 00003
0000100 1930-05-24 0000000 0000000 Y 00004
*
NUMBER_PEOPLE_BORN_BETWEEN 1924-11-02 1945-09-09
NUMBER_PEOPLE_BORN_BETWEEN 1903-03-26 1910-05-19
NUMBER_PEOPLE_BORN_BETWEEN 2003-06-05 2016-03-09
NUMBER_PEOPLE_BORN_BETWEEN 1942-07-11 1983-05-15
NUMBER_PEOPLE_BORN_BETWEEN 1930-05-20 1987-08-12
NUMBER_PEOPLE_BORN_BETWEEN 1928-01-24 1964-12-21
NUMBER_PEOPLE_BORN_BETWEEN 1905-10-01 2012-01-30
NUMBER_PEOPLE_BORN_BETWEEN 2017-04-18 2019-08-17
NUMBER_PEOPLE_BORN_BETWEEN 1981-06-28 2001-11-10
NUMBER_PEOPLE_BORN_BETWEEN 1959-07-14 1986-05-23
***

```

Figure 16 b2- Output 3

Correct output

```
75
0
0
12
55
78
100
0
10
8
```

User output

```
75
0
0
12
55
78
100
0
10
8
```

Figure 17 b2- Input 4

Input

```
0000001 1920-01-18 0000000 0000000 Y 00003
0000002 1920-02-11 0000000 0000000 Y 00005
0000003 1948-08-04 0000001 0000002 Y 00001
0000004 1946-09-25 0000001 0000002 Y 00001
0000005 1920-10-18 0000000 0000000 Y 00002
0000006 1920-01-07 0000000 0000000 Y 00002
0000007 1947-09-05 0000005 0000006 Y 00004
0000008 1943-08-27 0000005 0000006 Y 00002
0000009 1920-04-15 0000000 0000000 Y 00003
0000010 1920-07-14 0000000 0000000 Y 00002
*
NUMBER_PEOPLE_BORN_AT 1959-12-24
NUMBER_PEOPLE_BORN_AT 1997-02-28
NUMBER_PEOPLE_BORN_AT 1999-07-02
NUMBER_PEOPLE_BORN_AT 1936-03-10
NUMBER_PEOPLE_BORN_AT 1910-11-04
NUMBER_PEOPLE_BORN_AT 1941-05-03
NUMBER_PEOPLE_BORN_AT 1982-01-17
NUMBER_PEOPLE_BORN_AT 1992-03-22
```

Figure 18 b2- Output 4

```
NUMBER_PEOPLE_BORN_AT 1956-03-03
NUMBER_PEOPLE_BORN_AT 1998-10-02
***
```

Correct output

```
0
0
0
0
0
0
0
0
0
0
0
0
```

User output

```
0
0
0
0
0
0
0
0
0
0
0
0
```

Code:

```
//Mai Minh Hoàng
//20215381
#include <list>
#include <iostream>
#include<set>
#include<sstream>
#include<unordered_map>
#include<vector>
#include<map>
using namespace std;

long countPeople_81 = 0; // Biến đếm tổng số người
long sumtotalPeople_81 = 0; // Biến đếm tổng số người
map<string, long> totalPeople_81; // Bản đồ lưu tổng số người theo ngày sinh
unordered_map<string, int> notAlone_81; // Bản đồ kiểm tra người không một mình
unordered_map<string, int> maxAcestor_81; // Bản đồ lưu số tổ tiên tối đa của mỗi người
unordered_map<string, int> peopleBirth_81; // Bản đồ lưu số người theo ngày sinh
long countAlone_81 = 0; // Biến đếm số người một mình
long countGroup_81 = 0; // Biến đếm số nhóm

int main()
{
    ios_base::sync_with_stdio(0);
    string s;

    // Tạo bản đồ cho tất cả các ngày từ năm 1900 đến 2030
    for (int i = 1900; i < 2030; i++) {
        for (int j = 1; j <= 12; j++) {
            for (int k = 1; k <= 31; k++) {
                string year = to_string(i);
```

```

        to_string(j);
        string month = (j < 10) ? "0" + to_string(j) :
        string day = (k < 10) ? "0" + to_string(k) : to_string(k);
        string date = year + "-" + month + "-" + day;
        totalPeople_81[date] = 0; // Khởi tạo số người sinh vào
ngày đó là 0
    }
}

// Đọc dữ liệu từ đầu vào
while (getline(cin, s)) {
    if (s == "*") { // gặp * thì dừng
        break;
    }
    stringstream ss(s); // phân tách đầu vào
    string UserId;
    string Birth;
    string fCode;
    string mCode;
    string isAlive;
    string rCode;
    ss >> UserId >> Birth >> fCode >> mCode >> isAlive >> rCode;

    countPeople_81++; // Tăng biến đếm tổng số người
    if ((fCode != "0000000") && (mCode != "0000000")) {
        maxAcestor_81[UserId] = (maxAcestor_81[fCode] >
maxAcestor_81[mCode] ? maxAcestor_81[fCode] + 1 : maxAcestor_81[mCode] + 1);
        countGroup_81++; // Tăng biến đếm số nhóm
        notAlone_81[UserId] = 1; // Đánh dấu người này không một mình
        notAlone_81[fCode] = 1; // Đánh dấu cha/mẹ của người này không
một mình
        notAlone_81[mCode] = 1; // Đánh dấu cha/mẹ của người này không
một mình
    }
    else {
        notAlone_81[UserId] = 0; // Đánh dấu người này một mình
    }

    peopleBirth_81[Birth] += 1; // Tăng số người sinh vào ngày Birth
    totalPeople_81[Birth] += 1; // Tăng tổng số người sinh vào ngày Birth
}
int previousValue = 0;
for (auto& p : totalPeople_81) { // cộng dồn ngày sau bằng tổng tất cả người
sinh các ngày trước đó
    int currentValue = p.second;
    p.second = previousValue + currentValue; // Cập nhật tổng số người sinh
đến ngày hiện tại
    previousValue = p.second; // gán lại để tới vòng lặp tiếp
}
for (auto& p : notAlone_81) { // duyệt qua notAlone_81
    if (p.second == 0) {
        countAlone_81++; // Đếm số người một mình
    }
}
while (getline(cin, s)) {
    if (s == "***") { // gặp *** thì dừng
        break;
    }
}

```

```

    }
    if (s[7] == 'P') { //chuỗi là NUMBER_PEOPLE_BORN_BETWEEN <from_date>
<to_date>
        if (s.length() > 34) {
            stringstream ss(s);
            string a, b, c;
            ss >> a >> b >> c;

            cout << totalPeople_81[c] - totalPeople_81[b] +
peopleBirth_81[b] << "\n"; // Trả về số người sinh trong khoảng thời gian
        }
        else if (s.length() == 13) { //chuỗi là NUMBER_PEOPLE
            cout << countPeople_81 << "\n"; // Trả về tổng số người
        }
        else { //chuỗi là NUMBER_PEOPLE_BORN_AT <date>
            stringstream ss(s);
            string a, b;
            ss >> a >> b;
            cout << peopleBirth_81[b] << "\n"; // Trả về số người sinh
vào ngày <b>
        }
    }
    else
        if (s[0] == 'M' && s[1] == 'O') { //chuỗi là MOST_ALIVE_ANCESTOR
<code>
            stringstream ss(s);
            string a, b;
            ss >> a >> b;
            cout << maxAcestor_81[b] << "\n"; // Trả về số tổ tiên tối
đa của người b
        }
        else { //chuỗi là MAX_UNRELATED_PEOPLE
            // Trả về tổng số người một mình và số nhóm chính là kết
quả để 2 người bất kỳ trong nhóm không phải vợ chồng/ anh chị em
            cout << countAlone_81 + countGroup_81 << "\n";
        }
    }
}

```