# 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

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

# Điểm thưởng: $5\% \rightarrow 10\%$ (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 kíp được thì gửi mail xin phép thực hành bù trước 1 ngày qua mail <a href="hoalt@soict.hust.edu.vn">hoalt@soict.hust.edu.vn</a>, 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

Problem: Bank Transaction	
Problem: Analyze sales order of an e-commerce company.	14
Table of Figure	
Table of Figure	
Figure 1 Code bài 1	
Figure 2 b1- Input 1	
Figure 3 b1- output 1	
Figure 4 b1- Input 2	
Figure 5 b1- Output 2	10
Figure 6 b1-Input 3	10
Figure 7 b1-Output 3	11
Figure 8 Code bài 2	
Figure 9 b2-input 1	18
Figure 10 b2-output 1	18
Figure 11 b2-input 2	
Figure 12 b2-output 2	19
Figure 13 b2-input 3	20
Figure 14 b2-output 3	
Figure 15 b2-input 4	
Figure 16 b2- output 4	

# Báo cáo tuần 7

## Problem: Bank Transaction

The data about bank transactions consists of a sequence of transactions: the information of each transaction has the following format:

<from\_account> <to\_account> <money> <time\_point> <atm>

#### In which:

- <from\_account>: the account from which money is transferred (which is a string of length from 6 to 20 )
- <to\_account>: the account which receives money in the transaction (which is a string of length from 6 to 20)
- <money>: amount of money transferred in the transaction (which is an integer from 1 to 10000)
- <time\_point>: the time point at which the transaction is performed, it is a string under the format HH:MM:SS (hour: minute: second)
- <atm>: the code of the ATM where the transaction is taken (a string of length from 3 to 10)

Example: T00112233445 T001234002 2000 08:36:25 BIDV (at the ATM BIDV, account T00112233445 transfers 2000\$ to account T001234002 at time point 08:36:25 (08 hour, 36 minutes, 25 seconds)

A *transaction cycle* of length *k* starting from account *a*1 is defined to be a sequence of distinct account *a*1, *a*2, ..., *ak* in which there are transactions from account *a*1 to *a*2, from *a*2 to *a*3, ..., from *ak* to *a*1.

Write a program that process the following queries:

- ?number\_transactions: compute the total number of transactions of the data
- ?total\_money\_transaction: compute the total amount of money of transactions
- ?list\_sorted\_accounts: compute the sequence of bank accounts (including sending and receiving accounts) appearing in the transaction (sorted in an increasing (alphabetical) order)
- ?total\_money\_transaction\_from <account>: compute the total amount of money transferred from the account <account>
- ?inspect\_cycle <account> k : return 1 if there is a *transaction cycle* of length k, starting from <account>, and return 0, otherwise

#### Input (stdin)

The input consists of 2 blocks of information: the data block and the query block

- The data block consists of lines:
- o Each line contains the information about a transaction described above
- o The data is terminated by a line containing #
- The query block consists of lines:
- o Each line is a query described above
- o The query block is terminated by a line containing #

#### **Output (stdout)**

• Print to stdout (in each line) the result of each query described above

#### **Example**

#### Input

T000010010 T000010020 1000 10:20:30 ATM1
T000010010 T000010030 2000 10:02:30 ATM2
T000010010 T000010040 1500 09:23:30 ATM1
T000010020 T000010030 3000 08:20:31 ATM1
T000010030 T000010010 4000 12:40:00 ATM2
T000010040 T000010010 2000 10:30:00 ATM1
T000010020 T000010040 3000 08:20:31 ATM1

T000010040 T000010030 2000 11:30:00 ATM1

T000010040 T000010030 1000 18:30:00 ATM1

#

?number\_transactions

?total\_money\_transaction

?list\_sorted\_accounts

?total\_money\_transaction\_from T000010010

?inspect\_cycle T000010010 3

#

#### **Output**

9

19500

T000010010 T000010020 T000010030 T000010040

4500

1

#### Figure 1 Code bài 1

#### Source code

```
2
3
     #include <iostream>
     #include<sstream>
     #include<vector>
6
     #include<set>
    #include<map>
8
     #include<unordered_map>
9
     using namespace std;
10
11
    class Giaodich_81 // Định nghĩa lớp Giaodich_81
12
13
    public:
14
        string from_81; // Khai báo biến chuỗi from_81
15
        string to_81; // Khai báo biến chuỗi to_81
16
        long money_81; // Khai báo biến dài money_81
17
         string date_81; // Khai báo biến chuỗi date_81
18
         string atm_81; // Khai báo biến chuỗi atm_81
19
        Giaodich_81(string from_81, string to_81, long money_81, string date_81, string atm_81) {
20
             this->atm_81 = atm_81; // Gán giá trị cho atm_81
21
             this->from_81 = from_81; // Gán giá trị cho from_81
22
             this->to_81 = to_81; // Gán giá trị cho to_81
23
             this->money_81 = money_81; // Gán giá trị cho money_81
24
             this->date_81 = date_81; // Gán giá trị cho date_81
25
26
27
     vector<Giaodich_81> listGiaodich_81; // Khai báo vector chứa các đối tượng Giaodich_81
     set<string> listAccount_81; // Khai báo set chứa các tài khoản
29
     unordered_map<string, long> totalmoney_81; // Khai báo unordered_map chứa tổng số tiền
30 int dem_81 = 0; // Khai báo biến đếm
```

```
32
     map<string, bool> visited_81; // Khai báo map chứa trạng thái đã thăm của các tài khoản
33
     void find_cycle_from_81(string targetAccount_81, string account2_81, int n_81, int i_81) {
34
35
         if (i_81 == n_81) {
36
             if (account2_81 == targetAccount_81) {
37
                 dem_81 = 1;//nếu mục tiêu được tìm thấy và thỏa mãn số bước thì dem_81 bằng 1
38
                 return;
39
40
             else {
41
                 return;
42
43
44
         for (const Giaodich_81& gd_81 : listGiaodich_81) {//duyệt các đỉnh khác trong đồ thị giao dịch để tìm đ
45
46
             if (gd_81.from_81 == account2_81 && !visited_81[gd_81.to_81]) {
47
                 visited_81[gd_81.to_81] = true;//đánh dấu đã thăm
48
                 find_cycle_from_81(targetAccount_81, gd_81.to_81, n_81, i_81 + 1);//duyệt tiếp bằng DFS
49
                 visited_81[gd_81.to_81] = false;//xóa dấu đã thăm
50
51
             else {
52
53
54
             if (dem_81 == 1) {//dem_81 =1 thì dừng lại để tiết kiệm tài nguyên
55
56
             }
57
58
         return;
59
60
     int main()
61
62
         ios_base::sync_with_stdio(0); // Tắt đồng bộ giữa C và C++
63
```

```
int main()
60
61
62
         ios_base::sync_with_stdio(0); // Tắt đồng bộ giữa C và C++
63
64
         string s_81; // Khai báo biến chuỗi s_81
65
66
         string s1_81 = "?number_transactions"; // Khai báo chuỗi s1_81
67
         string s2_81 = "?total_money_transaction"; // Khai báo chuỗi s2_81
68
         string s3_81 = "?list_sorted_accounts"; // Khai báo chuỗi s3_81
69
         string s4_81 = "?total_money_transaction_from"; // Khai báo chuỗi s4_81
70
         long totalMoney_81 = 0; // Khai báo biến tổng số tiền
71
         long numberTransaction_81 = 0; // Khai báo biến số giao dịch
72
         while (getline(cin, s_81)) {//nhập dữ liệu
73
             if (s_81 == "#") {//gap \# thi divng}
74
                 break;
75
             stringstream ss_81(s_81); // Khai báo đối tượng stringstream
76
77
             string from_81; // Khai báo biến chuỗi from_81
78
             string to_81; // Khai báo biến chuỗi to_81
79
             long money_81; // Khai báo biến dài money_81
80
             string date_81; // Khai báo biến chuỗi date_81
81
             string atm_81; // Khai báo biến chuỗi atm_81
82
             ss_81 >> from_81 >> to_81 >> money_81 >> date_81 >> atm_81; // Đọc dữ liệu từ ss_81
83
             listGiaodich_81.push_back(Giaodich_81(from_81, to_81, money_81, date_81, atm_81)); // Thêm đối tượn
84
             listAccount_81.insert(from_81); // Thêm from_81 vào listAccount_81
85
             listAccount_81.insert(to_81); // Thêm to_81 vào listAccount_81
86
             totalMoney_81 = totalMoney_81 + money_81; // Cập nhật totalMoney_81
87
             numberTransaction_81++; // Tăng numberTransaction_81
88
             visited_81.insert(make_pair(from_81, false)); // Thêm cặp (from_81, false) vào visited_81
89
             visited_81.insert(make_pair(to_81, false)); // Thêm cặp (to_81, false) vào visited_81
90
             totalmoney_81[from_81] += money_81; // Cập nhật totalmoney_81
91
```

```
92
  93
                                                                  while (getline(cin, s_81)) {//nhập dữ liệu
 94
                                                                                               if (s_81 == "#") {\frac{1}{2}} / {\frac{1}{2}}
 95
                                                                                                                          break;
  96
  97
 98
 99
                                                                                                if (s_81 == s1_81) {
 100
                                                                                                                            cout << numberTransaction_81 << "\n"; // In ra số giao dịch</pre>
 101
102
                                                                                               else if (s_81 == s2_81) {
                                                                                                                          cout << totalMoney_81 << "\n"; // In ra tổng số tiền</pre>
103
104
  105
                                                                                               else if (s_81 == s3_81) {
 106
                                                                                                                            for (string account_81 : listAccount_81) {
107
                                                                                                                                                       cout << account_81 << " "; // In ra danh sách tài khoản
 108
 109
                                                                                                                          cout << "\n";
 110
```

```
111
             else if (s_81.find(s4_81) != -1) {
112
                 stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
113
                 string s4_81, s5_81;
114
                 ss2_81 >> s4_81 >> s5_81; // Đọc dữ liệu từ ss2_81
115
116
                 cout << totalmoney_81[s5_81] << "\n"; // In ra tổng số tiền giao dịch từ s5_81</pre>
117
             }
118
             else {
119
                 stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
120
                 string s4_81, s5_81;
121
                 int n_81;
122
                 ss2_81 >> s4_81 >> s5_81 >> n_81; // Đọc dữ liệu từ ss2_81
123
                 find_cycle_from_81(s5_81, s5_81, n_81, 0); // Tim chu trình từ s5_81
124
                 cout << ((dem_81 == 1) ? 1 : 0) << "\n"; // In ra kết quả
125
                 dem_81 = 0; // Đặt lại dem_81
126
127
128
         return 0; // Kết thúc chương trình
129 }
```

#### Test:

Figure 2 b1- Input 1

```
T000000008 T000000007 8045 18:43:42 atm1
                                                                                                   Ê
T000000009 T000000010 10542 11:11:13 atm1
T000000001 T000000009 1259 00:50:46 atm3
T000000004 T000000006 5138 11:40:18 atm3
T000000010 T000000008 1585 12:44:07 atm1
T000000004 T000000005 6119 22:43:19 atm4
T000000001 T000000008 10830 18:11:32 atm1
T000000003 T000000010 9402 00:00:26 atm5
T000000005 T000000006 8201 01:01:11 atm4
T000000002 T000000009 3534 06:07:50 atm3
?total_money_transaction_from T000000008
?total_money_transaction_from T000000003
?total_money_transaction_from T000000004
?total_money_transaction_from T0000000009
?total_money_transaction_from T000000005
?total_money_transaction_from T000000008
?total_money_transaction_from T000000008
?total_money_transaction_from T000000005
```

Figure 3 b1- output 1

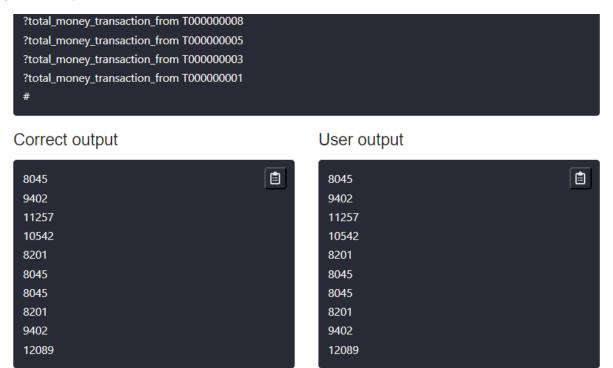


Figure 4 b1- Input 2

```
Ê
T000000002 T000000004 1995 06:10:04 atm5
T000000002 T000000005 3764 04:22:45 atm2
T000000001 T000000003 1772 12:32:13 atm1
T000000005 T000000005 3099 19:07:28 atm1
T000000004 T000000001 4922 00:32:30 atm6
T000000001 T000000007 3068 00:46:25 atm2
T000000003 T000000006 3962 20:46:36 atm2
T000000005 T000000006 5264 10:14:09 atm4
T000000005 T000000002 4285 11:59:50 atm1
T000000004 T000000001 9966 04:23:31 atm2
T000000001 T000000007 4197 10:54:00 atm3
T000000001 T000000002 9538 13:07:47 atm5
T000000006 T000000007 9052 19:52:46 atm1
T000000002 T000000004 9102 12:28:39 atm1
T000000005 T000000005 1139 19:38:12 atm1
?inspect_cycle T000000001 5
?inspect_cycle T000000005 5
```

Figure 5 b1- Output 2



Figure 6 b1-Input 3

```
Ê
T000000001 T000000003 4844 20:38:45 atm3
T000000003 T000000005 9559 00:16:18 atm3
T000000009 T000000005 3127 21:47:51 atm6
T000000004 T000000007 3236 06:24:57 atm4
T000000006 T000000006 4513 06:35:14 atm1
T000000009 T000000008 5475 03:10:57 atm4
T000000008 T000000003 9495 19:12:08 atm5
T000000009 T000000007 8821 11:25:19 atm3
T000000009 T000000008 2800 23:09:57 atm2
T000000003 T000000008 4178 19:10:37 atm6
T000000008 T000000002 3238 02:02:22 atm2
T000000010 T000000006 6892 06:47:35 atm2
T000000005 T000000007 7859 12:12:52 atm4
T000000003 T000000005 1971 09:07:20 atm2
T000000003 T000000001 2479 01:51:46 atm6
T000000005 T000000007 5003 17:51:42 atm5
T000000003 T000000006 4089 13:17:40 atm6
T000000005 T000000010 9562 18:22:09 atm4
```

```
T000000004 T000000009 8915 00:36:58 atm5
T000000001 T000000006 3670 19:46:22 atm2
#
?inspect_cycle T000000003 4
?inspect_cycle T000000002 2
?inspect_cycle T000000002 5
?inspect_cycle T000000010 4
?inspect_cycle T000000008 2
?inspect_cycle T000000003 5
?inspect_cycle T000000009 4
#
```

## Correct output



## User output



#### Code:

```
//Mai Minh Hoàng
//20215381
#include <iostream>
#include<sstream>
#include<vector>
#include<set>
#include<map>
#include<unordered_map>
using namespace std;
class Giaodich_81 // Định nghĩa lớp Giaodich_81
public:
   string from_81; // Khai báo biến chuỗi from_81
    string to_81; // Khai báo biến chuỗi to_81
    long money_81; // Khai báo biến dài money_81
    string date_81; // Khai báo biến chuỗi date_81
    string atm_81; // Khai báo biến chuỗi atm_81
    Giaodich_81(string from_81, string to_81, long money_81, string date_81, string
atm_81) {
        this->atm_81 = atm_81; // Gán giá trị cho atm_81
        this->from_81 = from_81; // Gán giá trị cho from_81
        this->to_81 = to_81; // Gán giá trị cho to_81
        this->money_81 = money_81; // Gán giá trị cho money_81
        this->date_81 = date_81; // Gán giá tri cho date_81
    }
```

```
};
vector<Giaodich_81> listGiaodich_81; // Khai báo vector chứa các đối tượng
set<string> listAccount_81; // Khai báo set chứa các tài khoản
unordered_map<string, long> totalmoney_81; // Khai báo unordered_map chứa tổng số
tiền
int dem_81 = 0; // Khai báo biến đếm
map<string, bool> visited_81; // Khai báo map chứa trạng thái đã thăm của các tài
khoản
void find_cycle_from_81(string targetAccount_81, string account2_81, int n_81, int
i_81) {
    //tìm chu trình bắt đầu từ tài khoản tùy ý
    if (i_81 == n_81) {
        if (account2_81 == targetAccount_81) {
            dem_81 = 1;//nếu mục tiêu được tìm thấy và thỏa mãn số bước thì dem_81
bằng 1
            return;
        }
        else {
            return;
    for (const Giaodich_81& gd_81 : listGiaodich_81) {//duyệt các đỉnh khác trong đồ
thị giao dịch để tìm đỉnh tiếp theo
        if (gd_81.from_81 == account2_81 && !visited_81[gd_81.to_81]) {
            visited_81[gd_81.to_81] = true;//đánh dấu đã thăm
            find_cycle_from_81(targetAccount_81, qd_81.to_81, n_81, i_81 +
1);//duyêt tiếp bằng DFS
            visited_81[gd_81.to_81] = false;//xóa dấu đã thăm
        }
        else {
            continue;
        }
        if (dem_81 == 1) {//dem_81 =1 thì dừng lại để tiết kiệm tài nguyên
            return;
    }
    return;
}
int main()
{
    ios_base::sync_with_stdio(0); // Tắt đồng bô giữa C và C++
    string s_81; // Khai báo biến chuỗi s_81
    string s1_81 = "?number_transactions"; // Khai báo chuỗi s1_81
    string s2_81 = "?total_money_transaction"; // Khai báo chuỗi s2_81
string s3_81 = "?list_sorted_accounts"; // Khai báo chuỗi s3_81
    string s4_81 = "?total_money_transaction_from"; // Khai báo chuôi s4_81
    long totalMoney_81 = 0; // Khai báo biến tổng số tiền
    long numberTransaction_81 = 0; // Khai báo biến số giao dịch
    while (getline(cin, s_81)) {//nhập dữ liệu
        if (s_81 == "#") {//găp # thì dừng}
            break:
        stringstream ss_81(s_81); // Khai báo đối tượng stringstream
        string from_81; // Khai báo biến chuỗi from_81
        string to_81; // Khai báo biến chuỗi to_81
```

```
long money_81; // Khai báo biến dài money_81
        string date_81; // Khai báo biến chuỗi date_81
        string atm_81; // Khai báo biến chuỗi atm_81
        ss_81 >> from_81 >> to_81 >> money_81 >> date_81 >> atm_81; // Đọc dữ liệu
từ ss_81
        listGiaodich_81.push_back(Giaodich_81(from_81, to_81, money_81, date_81,
atm_81)); // Thêm đối tượng Giaodich_81 vào listGiaodich_81
        listAccount_81.insert(from_81); // Thêm from_81 vào listAccount_81
        listAccount_81.insert(to_81); // Thêm to_81 vào listAccount_81
        totalMoney_81 = totalMoney_81 + money_81; // Câp nhật totalMoney_81
        numberTransaction_81++; // Tăng numberTransaction_81
        visited_81.insert(make_pair(from_81, false)); // Thêm căp (from_81, false)
vào visited_81
        visited_81.insert(make_pair(to_81, false)); // Thêm cặp (to_81, false) vào
visited_81
        totalmoney_81[from_81] += money_81; // Câp nhật totalmoney_81
    while (getline(cin, s_81)) {//nhập dữ liệu
        if (s_81 == "#") {//gặp # thì dừng}
            break;
        }
        if (s_81 == s1_81) {
            cout << numberTransaction_81 << "\n"; // In ra số giao dich</pre>
        else if (s_81 == s2_81) {
            cout << totalMoney_81 << "\n"; // In ra tổng số tiền</pre>
        else if (s_81 == s3_81) {
            for (string account_81 : listAccount_81) {
                cout << account_81 << " "; // In ra danh sách tài khoản</pre>
            cout << "\n";
        }
        else if (s_81.find(s4_81) != -1) {
            stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
            string s4_81, s5_81;
            ss2_81 >> s4_81 >> s5_81; // Đoc dữ liêu từ ss2_81
            cout << totalmoney_81[s5_81] << "\n"; // In ra tổng số tiền giao dịch từ
s5_81
        }
        else {
            stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
            string s4_81, s5_81;
            int n_81;
            ss2_81 >> s4_81 >> s5_81 >> n_81; // Đọc dữ liệu từ ss2_81
            find_cycle_from_81(s5_81, s5_81, n_81, 0); // Tim chu trình từ s5_81
            cout << ((dem_81 == 1) ? 1 : 0) << "\n"; // In ra kết quả</pre>
            dem_81 = 0; // Đặt lại dem_81
        }
    }
    return 0; // Kết thúc chương trình
}
```

# Problem: Analyze sales order of an e-commerce company

Data about sales in an e-commerce company (the e-commerce company has several shops) consists a sequence of lines, each line (represents an order) has the following information:

<CustomerID> <ProductID> <Price> <ShopID> <TimePoint>

in which the customer <CustomerID> buys a product <ProductID> with price <Price> at the shop <ShopID> at the time-point <TimePoint>

- <CustomerID>: string of length from 3 to 10
- < ProductID>: string of length from 3 to 10
- <Price>: a positive integer from 1 to 1000
- <ShopID>: string of length from 3 to 10
- <TimePoint>: string representing time-point with the format HH:MM:SS (for example, 09:45:20 means the time-point 9 hour 45 minutes 20 seconds)

Perform a sequence of queries of following types:

- ?total\_number\_orders: return the total number of orders
- ?total\_revenue: return the total revenue the e-commerce company gets
- ?revenue of shop <ShopID>: return the total revenue the shop <ShopID> gets
- ?total\_consume\_of\_customer\_shop <CustomerID> <ShopID>: return the total revenue the shop <ShopID> sells products to customer <CustomerID>
- ?total\_revenue\_in\_period <from\_time> <to\_time>: return the total revenue the e-commerce gets of the period from <from\_time> to <to\_time> (inclusive)

#### Input

The input consists of two blocks of data:

The first block is the operational data, which is a sequence of lines (number of lines can be upto 100000), each line contains the information of a submission with above format

The first block is terminated with a line containing the character #

The second block is the query block, which is a sequence of lines (number of lines can be upto 100000), each line is a query described above

The second block is terminated with a line containing the character #

### **Output**

Write in each line, the result of the corresponding query

### **Example**

#### Input

C001 P001 10 SHOP001 10:30:10

C001 P002 30 SHOP001 12:30:10

C003 P001 40 SHOP002 10:15:20

C001 P001 80 SHOP002 08:40:10

C002 P001 130 SHOP001 10:30:10

C002 P001 160 SHOP003 11:30:20

#

?total\_number\_orders

?total\_revenue

?revenue\_of\_shop SHOP001

?total\_consume\_of\_customer\_shop C001 SHOP001

?total\_revenue\_in\_period 10:00:00 18:40:45

#

#### **Output**

6

450

170

40

370

Figure 8 Code bài 2

Source code

```
3 #include <iostream>
4 #include<set>
5 #include<sstream>
6 #include<unordered_map>
   using namespace std;
9 long countOrder_81 = 0; // Khai báo biến đếm số lượng đơn hàng
10 long sumTotalMoney_81 = 0; // Khai báo biến tổng số tiến
11 unordered_map<string, long> totalmoney_81; // Khai báo unordered_map chứa tổng số tiến theo ShopId
12 unordered_map<string, long> totalmoney_product_81; // Khai báo unordered_map chứa tổng số tiền theo CustomId và ShopId
13 long time_order_81[86400]; // Khai báo mảng chứa số tiền theo thời gian
14
15 int main()
16 {
         ios_base::sync_with_stdio(θ); // Tắt đồng bộ
18
19
         string s_81; // Khai báo biến chuỗi s_81
20
         while (getline(cin, s_81)) {
21
            if (s_81 == "#") {
                 break;
23
            stringstream ss_81(s_81); // Khai báo đối tượng stringstream
             string CustomId_81; // Khai báo biến chuỗi CustomId_81
25
26
            string Product_81; // Khai báo biến chuỗi Product_81
            long money_81; // Khai báo biến dài money_81
27
28
            string ShopId_81; // Khai báo biến chuỗi ShopId_81
             string Time_81; // Khai báo biến chuỗi Time_81
29
             ss_81 >> CustomId_81 >> Product_81 >> money_81 >> ShopId_81 >> Time_81; // Đọc dữ liệu từ ss_81
30
32
             countOrder_81++; // Tăng countOrder_81
33
             sumTotalMoney_81 += money_81; // Cập nhật sumTotalMoney_81
             totalmoney_81[ShopId_81] += money_81; // Cập nhật totalmoney_81
34
```

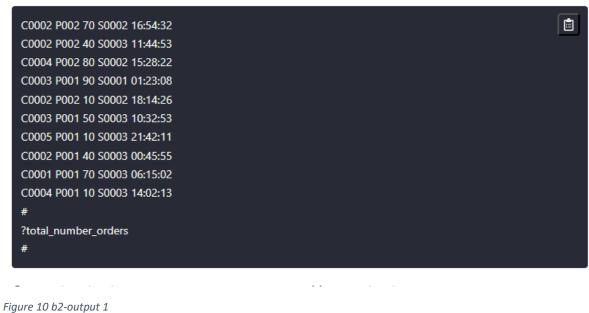
```
totalmoney_product_81[CustomId_81 + " " + ShopId_81] += money_81; // Câp nhật totalmoney_product_81
int h_81 = stoi(Time_81.substr(0, 2)); // Chuyển đổi giờ thành số nguyên
int m_81 = stoi(Time_81.substr(3, 5)); // Chuyển đổi phút thành số nguyên
int sss_81 = stoi(Time_81.substr(6)); // Chuyển đổi giây thành số nguyên
time_order_81[h_81 * 60 * 60 + m_81 * 60 + sss_81] += money_81; // Cập nhật time_order_81

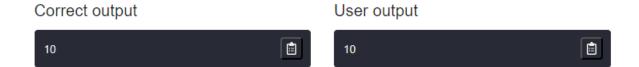
for (int i_81 = 0; i_81 <= 86400; i_81++) {
    time_order_81[i_81] += time_order_81[i_81 - 1]; // Cập nhật time_order_81
}
```

```
while (getline(cin, s_81)) {
            if (s_81 == "#") {
                break;
            if (s_81[7] == 'r') {
49
                if (s_81.length() > 15) {
50
                    stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
51
                    string a_81, b_81, c_81;
52
                    ss2_81 >> a_81 >> b_81 >> c_81; // Đọc dữ liệu từ ss2_81
53
                    int h_81 = stoi(b_81.substr(0, 2)); // Chuyển đổi giờ thành số nguyên
54
                    int m_81 = stoi(b_81.substr(3, 5)); // Chuyển đổi phút thành số nguyên
                    int sss_81 = stoi(b_81.substr(6)); // Chuyển đổi giây thành số nguyên
56
                    int timee_81 = h_81 * 60 * 60 + m_81 * 60 + sss_81; // Tinh toán thời gian
57
                    int h1_81 = stoi(c_81.substr(0, 2)); // Chuyển đổi giờ thành số nguyên
58
                    int m1_81 = stoi(c_81.substr(3, 5)); // Chuyển đổi phút thành số nguyên
                    int sss1_81 = stoi(c_81.substr(6)); // Chuyển đổi giây thành số nguyên
59
60
                    int timee1_81 = h1_81 * 60 * 60 + m1_81 * 60 + sss1_81; // Tinh toan thời gian
61
                    cout << time_order_81[timee1_81] - time_order_81[timee_81 - 1] << "\n"; // In ra tổng số tiền trong khoảng thời gian
                else {
                    cout << sumTotalMoney_81 << "\n"; // In ra tổng số tiền
66
68
70
                if (s_81[7] == 'n') {
                    cout << countOrder_81 << "\n"; // In ra số lượng đơn hàng
                else if (s_81[9] == 'o') {
                    stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
75
                    string a_81, b_81;
                    ss2_81 >> a_81 >> b_81; // Đọc dữ liệu từ ss2_81
                    cout << totalmoney_81[b_81] << "\n"; // In ra tổng số tiền theo ShopId</pre>
                 else {
 80
                     stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
```

```
79 else {
80 stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
81 string a_81, b_81, c_81;
82 ss2_81 >> a_81 >> b_81 >> c_81; // Đạc dữ liệu từ ss2_81
83 cout << totalmoney_product_81[b_81 + " " + c_81] << "\n"; // In ra tổng số tiến theo CustomId và ShopId
84 }
85
86 }
87 }
```

#### Test:





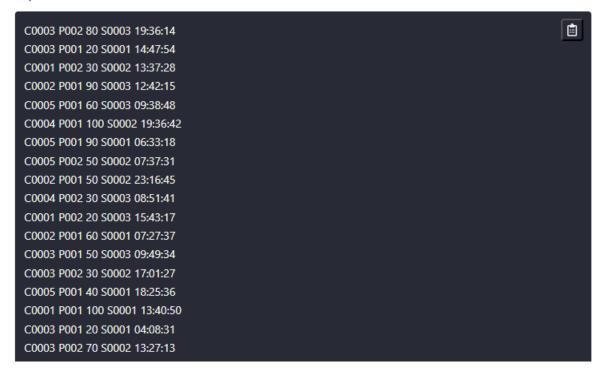


Figure 12 b2-output 2



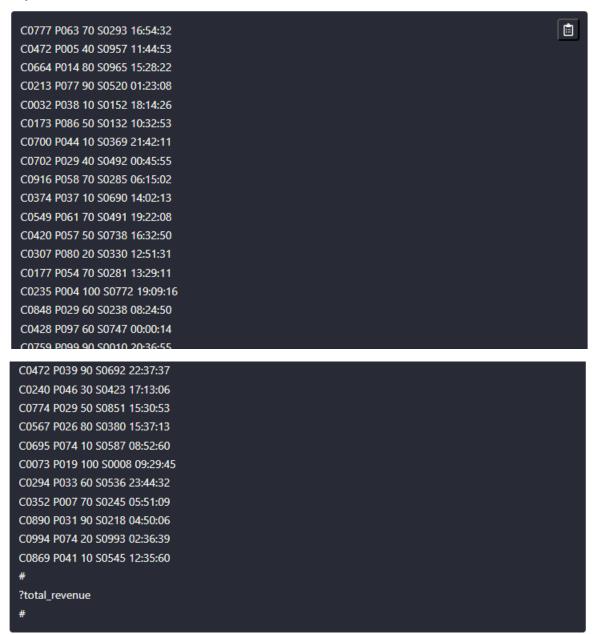
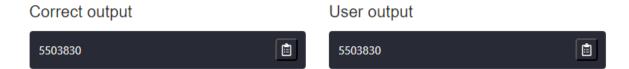
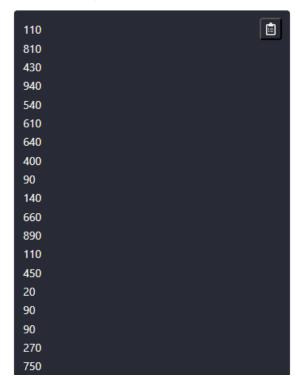


Figure 14 b2-output 3

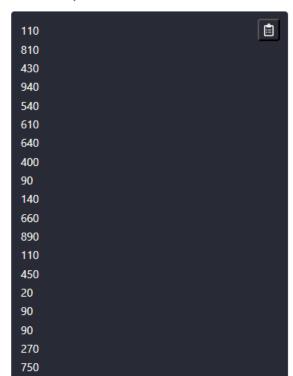




## Correct output



#### User output



#### Code:

```
//Mai Minh Hoàng
//20215381
#include <iostream>
#include<set>
#include<sstream>
#include<unordered_map>
using namespace std;
long countOrder_81 = 0; // Khai báo biến đếm số lượng đơn hàng
long sumTotalMoney_81 = 0; // Khai báo biến tổng số tiền
unordered_map<string, long> totalmoney_81; // Khai báo unordered_map chứa tổng số
tiền theo ShopId
unordered_map<string, long> totalmoney_product_81; // Khai báo unordered_map chứa
tống số tiền theo CustomId và ShopId
long time_order_81[86400]; // Khai báo mảng chứa số tiền theo thời gian
int main()
    ios_base::sync_with_stdio(0); // Tắt đồng bô
    string s_81; // Khai báo biến chuỗi s_81
    while (getline(cin, s_81)) {
        if (s_81 == "#") {
            break:
        }
        stringstream ss_81(s_81); // Khai báo đối tượng stringstream
        string CustomId_81; // Khai báo biến chuỗi CustomId_81
        string Product_81; // Khai báo biến chuỗi Product_81
```

```
long money_81; // Khai báo biến dài money_81
        string ShopId_81; // Khai báo biến chuỗi ShopId_81
        string Time_81; // Khai báo biến chuỗi Time_81
        ss_81 >> CustomId_81 >> Product_81 >> money_81 >> ShopId_81 >> Time_81; //
Đọc dữ liệu từ ss_81
        countOrder_81++; // Tăng countOrder_81
        sumTotalMoney_81 += money_81; // Câp nhật sumTotalMoney_81
        totalmoney_81[ShopId_81] += money_81; // Cập nhật totalmoney_81
        totalmoney_product_81[CustomId_81 + " " + ShopId_81] += money_81; // Câp
nhật totalmoney_product_81
        int h_81 = stoi(Time_81.substr(0, 2)); // Chuyển đổi giờ thành số nguyên
        int m_81 = stoi(Time_81.substr(3, 5)); // Chuyến đối phút thành số nguyên
        int sss_81 = stoi(Time_81.substr(6)); // Chuyển đổi giây thành số nguyên
        time_order_81[h_81 * 60 * 60 + m_81 * 60 + sss_81] += money_81; // Câp nhật
time_order_81
   for (int i_81 = 0; i_81 <= 86400; i_81++) {</pre>
        time_order_81[i_81] += time_order_81[i_81 - 1]; // Câp nhật time_order_81
    while (getline(cin, s_81)) {
        if (s_81 == "#") {
            break;
        if (s_81[7] == 'r') {
            if (s_81.length() > 15) {
                stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
                string a_81, b_81, c_81;
                ss2_81 >> a_81 >> b_81 >> c_81; // Đọc dữ liệu từ ss2_81
                int h_81 = stoi(b_81.substr(0, 2)); // Chuyển đổi giờ thành số
nguyên
                int m_81 = stoi(b_81.substr(3, 5)); // Chuyển đổi phút thành số
nguyên
                int sss_81 = stoi(b_81.substr(6)); // Chuyển đổi giây thành số
nguyên
                int timee_81 = h_81 * 60 * 60 + m_81 * 60 + sss_81; // Tính toán
thời gian
                int h1_81 = stoi(c_81.substr(0, 2)); // Chuyển đổi giờ thành số
nguyên
                int m1_81 = stoi(c_81.substr(3, 5)); // Chuyển đổi phút thành số
nguyên
                int sss1_81 = stoi(c_81.substr(6)); // Chuyển đổi giây thành số
nguyên
                int timee1_81 = h1_81 * 60 * 60 + m1_81 * 60 + sss1_81; // Tính toán
thời gian
                cout << time_order_81[timee1_81] - time_order_81[timee_81 - 1] <</pre>
"\n"; // In ra tổng số tiền trong khoảng thời gian
            else {
                cout << sumTotalMoney_81 << "\n"; // In ra tổng số tiền</pre>
        }
        else
            if (s_81[7] == 'n') {
                cout << countOrder_81 << "\n"; // In ra số lượng đơn hàng</pre>
            else if (s_81[9] == 'o') {
                stringstream ss2_81(s_81); // Khai báo đối tượng stringstream
```