Introduction

One useful data point in detecting fraud is the account history of a customer.
For an account, we receive notification of purchases and, sometimes, reports of fraud.
Typically, a prior report of fraud for an account would increase the perceived risk of fraud on future transactions.
Similarly, a history of non-fraudulent purchases for an account would decrease the risk of fraud. A credit card holder has 90 days to report any fraudulent transactions with the card. So if an account has purchases over 90 days old and no reports of fraud, we assume that these older purchases were not-fraudulent.

Problem Description

The purpose of this programming problem is to determine the status of a customer account history at the time a new purchase is made.
The input is a sequence of customer account events, in chronological order. Each event has three fields, all of which are of string type
`<DATE>,<CUSTOMER_ACCOUNT_ID>,<EVENT_TYPE>`

For example:

```
2015-01-01,joe@signifyd.com,PURCHASE
2015-02-01,fraudster@fraud.com,FRAUD_REPORT
```

There are two event types:

1. `PURCHASE` - indicates a purchase by this customer account on the specified date.
2. `FRAUD_REPORT` - indicates we received a report of fraud associated with this customer account. The specified date is that date that we received the report, not the date the fraud was committed.

For each event`PURCHASE`, we are interested in a summary of the customer account history based on prior events. The summary consists of the date of the summary, the customer account id, and a status.

There are four possible values for the status of the customer account history:

1. `NO_HISTORY` - there are no prior events for this customer account
2. `FRAUD_HISTORY` - we have at least one event `FRAUD_REPORT`for this customer account.
3. `GOOD_HISTORY`- customer account has no `FRAUD_REPORT`s and at least one prior that `PURCHASE`is more than 90 days old.

4. `UNCONFIRMED_HISTORY` - customer account has no `FRAUD_REPORT`s and at least one prior `PURCHASE` but no `PURCHASE`s over 90 days old.

For accounts with `FRAUD_HISTORY`, `GOOD_HISTORY`, and `UNCONFIRMED_HISTORY`, the status also contains a count of relevant events.

- `FRAUD_HISTORY` - count of `FRAUD_REPORT`s
- `GOOD_HISTORY` - count of priors `PURCHASE` over 90 days old
- `UNCONFIRMED_HISTORY`- count of prior purchases.

The output is expected to be in the same order as the input.

Sample Input and Output

For the following input:

```
2015-01-01,joe@signifyd.com,PURCHASE
2015-02-01,fraudster@fraud.com,FRAUD_REPORT
2015-02-03,fraudster@fraud.com,FRAUD_REPORT
2015-02-10,joe@signifyd.com,PURCHASE
2015-02-14,fraudster@fraud.com,PURCHASE
2015-03-15,joe@signifyd.com,PURCHASE
2015-05-01,joe@signifyd.com,PURCHASE
2015-10-01,joe@signifyd.com,PURCHASE
```

The following output is expected.

```
2015-01-01,joe@signifyd.com,NO_HISTORY
2015-02-10,joe@signifyd.com,UNCONFIRMED_HISTORY:1
2015-02-14,fraudster@fraud.com,FRAUD_HISTORY:2
2015-03-15,joe@signifyd.com,UNCONFIRMED_HISTORY:2
2015-05-01,joe@signifyd.com,GOOD_HISTORY:1
2015-10-01,joe@signifyd.com,GOOD_HISTORY:4
```

FAQ

1. Does input come in as a file?
   Each event is for an individual customer, so they can come at different times. Whether to design code to handle individual events or a set of events is your call to make.

2. How many input records should I expect?
   There is no specific limit, however, you should assume that there is sufficient RAM available that you will not use disk or other external storage.

3. How should I interpret multiple events with the same customer account id and the same date?
   For the purposes of this problem, assume that there will be at most one event per customer account id per day.

4. Are the input events in any order?
   Yes, the input events are in chronological order (ascending).

5. How many output records will my program produce?
   One output record for each `PURCHASE` event in the input. For each `PURCHASE` event, we want to know the customer account history (based on prior events) at the time of the `PURCHASE` event.

6. How should I handle invalid input records?
   Don't write code to handle invalid input records. Assume that all input records will be valid, i.e. contain 3 fields, the first with a valid date in the format 'YYYY-MM-DD', the second with a non-empty customer id string, and the third with either `PURCHASE` or `FRAUD_REPORT`.