Thank you for your interest in becoming a developer for Solium Capital. We like to make sure you write code as part of the interview process, because that's what you'll be doing if we hire you. Please take a look at the problem below and submit the solution to Dropbox at the link below:

https://www.dropbox.com/request/a1S4VE0YtN3XFfmUjko9

If the link does not work, if you would like to confirm receipt of the submission, or if you have any questions, please contact dev-careers@solium.com. We typically respond within a business day, so follow up if you do not hear from us.

We respect that people have other commitments, but in general we like to see solutions within 5 business days.

Please read and understand the following before reading the problem as this section describes general requirements. If you have any questions, don't hesitate to send an email to dev-careers@solium.com.

This is your chance to prove that you can be one of our developers by impressing us with your creativity and polish. You may use the language of your choice, so long as it can be run on Windows, Linux, or OS X. Since our application is written in Java, there is a slight bias toward it, but we have been known to hire programmers who have never written a line of Java.

In order for your application to be considered complete, we require the following:

- All files – source, test, and documentation – must be stored within a single archive
- There must be a one-step build (if applicable to the language) – 'ant build' is fine
- There must be a way to run it with a single command (elaborated below)

The correctness of your submission will be determined using an automated test suite, so your program **must** read input from **stdin** and write output to **stdout**. Command-line arguments or GUI applications will not be compatible; any submission where these are the only means of I/O will not be considered. To be clear, your program should not require any action on the part of the user, nor should it output anything other than the answer. In an UNIX-like operating system, we should be able to run your program with a command similar to this:

```
java -jar solution.jar < input.def
```

We encourage the use of tools such as gradle to build Java programs and TestNG for testing them.

Finally, please note that while we provide sample input and output, we expect your solution to be general and work for **whatever legal input** we choose to give it.

Good Luck.

Stock Option Vesting

As part of a robust compensation package, employees are given the option to buy stock (a "stock option") at a favorable set price (the "grant" or "strike" price) after a certain period of time has passed (the "vesting period"). The date that the options are granted is known as the "grant date," and the date the options are available is known as the "vest date."

Given a list of records outlining vest dates and amounts, calculate the total gain an employee has at a given date and market price.

Note: For any required rounding, use "round half up" behavior – round towards "nearest neighbor" unless both neighbors are equidistant, in which case round up.

Input: The input consists of a record count N followed by N comma delimited rows. Each row will have 5 fields. The first field will be the word "VEST". The second field is an arbitrary string representing an individual employee. The third will be the **vest date** in YYYYMMDD format. The fourth is the amount of units that are vesting. The fifth and final field is the **grant price** for those options it is a decimal number round to two places – currency is ignored in this case. There is one final input line that consists of a date in YYYYMMDD format and a **market price** for the stock as at that date.

Output: Output will be a two-field comma delimited row for each employee (sorted by employee identifier). The first field will be the employee identifier. The second field will be the total cash gain available for that employee – ignore currency and display as a decimal number rounded to 2 places. Total gain per vested option is calculated as **market price – grant price**. Remember that different rows can have different total gains. Any row with a negative value can be ignored. If an employee does not have anything vested by the date given, they should still appear in the output with a total gain of **0.00**. If an employee has a negative net gain (these options would be called "under water"), they should still appear in the output with a total gain of **0.00**.

Example:

Input:

```
5
VEST,001B,20120101,1000,0.45
VEST,002B,20120101,1500,0.45
VEST,002B,20130101,1000,0.50
VEST,001B,20130101,1500,0.50
VEST,003B,20130101,1000,0.50
20140101,1.00
```

Output:

```
001B,1300.00
002B,1325.00
003B,500.00
```

If the date were changed to 20120615:

```
001B,550.00
002B,825.00
003B,0.00
```

BONUS 1: For exceptional performance, occasionally companies will use a multiplier against the number of granted units as an extra bonus. For example, if an executive has her team hit certain revenue targets, she might get a 50% bonus, for a 1.5 multiplier. So instead of 1000 units, she will receive 1500. A new record type will be added. It is a comma delimited record where the first field is always "PERF", the second field is the employee identifier, the third field is the date the bonus takes effect, and the last field is the multiplier. For simplification purposes, this multiplier **applies to all available units on or before the day of the bonus**.

Example:
Input:

```
5
VEST,001B,20120102,1000,0.45
VEST,002B,20120102,1000,0.45
VEST,003B,20120102,1000,0.45
PERF,001B,20130102,1.5
PERF,002B,20130102,1.5
20140101,1.00
```

Output:

```
001B,825.00
002B,825.00
003B,550.00
```

If final date was changed to 20130101:

```
001B,550.00
002B,550.00
003B,550.00
```

BONUS 2: After options have vested, to realize value, employees can sell their options on the market (they also need to buy them, but that will not be taken into account for this problem). This will add an additional record type that will consist of 5 comma-delimited fields. The first field is always "SALE". The second field is the employee identifier, the third field is the date of the sale, the fourth field is the amount sold, and the fifth field is the **market price** that the grants were sold at. Note that sold units are no longer considered available for performance calculations.

For the output to this question, we will be adding a third field to each line, which is total gain through sale.

Input:

```
5
VEST,001B,20120102,1000,0.45
SALE,001B,20120402,500,1.00
VEST,002B,20120102,1000,0.45
PERF,001B,20130102,1.5
PERF,002B,20130102,1.5
20140101,1.00
```

Output:

```
001B,412.50,275.00
002B,825.00,0.00
```

If final date was changed to 20130101:

```
001B,275.00,275.00
002B,550.00,0.00
```